«ТҰРАН» УНИВЕРСИТЕТІ

Д.Р. КУАНДЫКОВА, И.Т. УТЕПБЕРГЕНОВ,
А.Д. ХОМОНЕНКО

АҚПАРАТТЫҚ ЖҮЙЕЛЕРДЕГІ ДЕРЕКТЕР ҚОРЫНЫҢ ТЕОРИЯСЫ МЕН ТӘЖІРИБЕСІ

Оқу құралы

Алматы
2017

УДК 004.65 (075.8)
ББК 32.81 я 73
К 81

Пікір жазғандар:

Мамырбаев О.Ж. – Ph.D докторы, ассоц. профессор, Ақпараттық және есептеуіш технологиялар институты директорының орынбасары

Букашев А.А. – т.ғ.к., доцент, М.Тынышбаев атындағы Қазақ көлік және коммуникациялар академиясы

Нурмаганбетова А.Т. – т.ғ.к., доцент, Тұран университеті

К81 Куандыкова Д.Р., Утепбергенов И.Т., Хомоненко А.Д.

Ақпараттық жүйелердегі деректер қоры. Оқулық. – Алматы: «Тұран» университеті, 2017 – 256 бет.

ISBN 978-601-214-325-6

Оқу құралында деректер қоры және ақпараттық жүйелер, деректер қорын басқару жүйелері түсініктері, дербес және үлестірімді деректер қоры құрылымы қарастырылған. Деректердің реляциялық үлгісі және реляциялық деректер қорын жобалау сипаттамасы берілген. Клиенттің интерфейстік бөлігін құру технологиясы, дербес және үлестірімді деректер қорын C++ Builder деректерді өңдеудің біріктірілген жүйесі құралдары арқылы, сонымен қатар SQL Server серверлік бөлігі арқылы құру талданады. Web-қосымша құрылымы, HTML және XML тілдері, CSS стилі каскадты кестелері, HTTP хаттамасы бойынша негізгі мәліметтер келтірілген. Деректер қорымен жұмыс жасау мысалдары мен сипаттамалары берілген. EL тілі, JSTL кітапханасы, пайдаланушы әрекеті және тегтер кітапханасы сипатталған.

УДК 004.65 (075.8)
ББК 32.81 я 73

ISBN978-601-214-325-6

Мазмұны

ҚЫСҚАРҒАН СӨЗДЕР:

ДҚА – деректер қоры администраторы
ДҚ – деректер қоры
ДБ – деректер банкі
ЕЖ – есептеу жүйесі
АЖ – ақпараттық жүйелер
КДҚ – корпоративті ДҚ
КК – компьютер-клиент
КС – компьютер-сервер
ЖДҚ – жергілікті ДҚ
ЖЕЖ – жергілікті есептеу желісі
ОЖ – операциялық жүйе
ДДҚ – дербес ДҚ
ДҚБЖ – деректер қорын басқару жүйесі
ТФДҚБЖ – толық функционалдық ДҚБЖ
ДРҮ – деректердің реляциялық үлгісі
ЖБ – жұмыс бекеті
ДС – деректер сөздігі
ОДҚ – орталық ДҚ
API – Application Programming Interface
ATMI – Application Transaction Monitor Interface
CGI – Common Gateway Interface
DBS – DataBase Server
DDL – Data Definition Language
DML – Data Manipulation Language
JDBC – Java Database Connectivity
ODBC – Open Database Connectivity
QBE – Query By Example
RDA – Remote Data Access
RM – Resource Manager
SQL – Structured Query Language
TM – Transaction Manager
TPM – Transaction Processing Monitor
HTTP – HyperText Transfer Protocol
HTML – HyperText Markup Language
URI – Uniform Resource Identifie
URL – Uniform Resource Locator
WEB – World Wide Web
ISAPI – Internet Server API
IDC/HTX – Internet Database Connector HTML Extension
ASP – Active Server Pages
XML – eXtensible Markup Language
DTD – Document Type Definition
CSS – Cascading Style Sheets
JVM – Java Virtual Machine
MVC – Model-view-controller
JSP – Java Server Pages
JSTL – JavaServer Pages Standard Tag Library
EL – Expression Language

1. ДЕРЕКТЕР ҚОРЫНА КІРІСПЕ

Бұл бөлімде деректер қоры және ақпараттық жүйелер қарастырылады. Деректер қоры және деректер қорын басқару жүйелерінің негізгі ұғымдары сипатталады. Клиент-сервер құрылымы бойынша ақпараттық жүйелерді ұйымдастыру нұсқалары сипатталады. ДҚБЖ жіктеу және олардың негізгі функциялары келтіріледі. Қосымшалар құру және пайдаланушы мен ақпараттық жүйелердің өзара әрекеті қарастырылады.

1.1. Деректер қоры және ақпараттық жүйелер

Көптеген тапсырмалар шешімінің негізі ақпаратты өңдеу. Ақпаратты өңдеуді жеңілдету мақсатында ақпараттық жүйелер (АЖ) құрылады. Автоматтандырылған АЖ дегеніміз техникалық құралдар, жеке жағдайда ЭЕМ қолданылатын АЖ. Көптеген АЖ автоматтандырылған, сондықтан бұдан былай оларды қысқаша АЖ деп қарастырамыз.

Кеңейтілген мағынада, АЖ анықтамасына кез-келген ақпаратты өңдеу жүйелері жатады. АЖ қолдану облысы бойынша өндірістегі, білім беру саласындағы, денсаулық сақтау саласындағы, ғылымдағы, әскери бөлімдегі, әлеуметтік саладағы, сауда саласындағы және т.б. салалардағы АЖ деп бөлуге болады. Атқаратын қызметі бойынша АЖ келесі топтарға жіктеледі: басқарушы, ақпараттық-анықтамалық, шешім қабылдауды қолдайтын.

Кей жағдайда, АЖ тар мағынада, қайсібір қолданбалы есепті шешу үшін қажетті аппараттық-бағдарламалық құралдар жиынтығы ретінде қолданылатынын ескере кеткеніміз жөн. Мысалы, мекемеде келесі міндеттер жүктелген ақпараттық жүйелер болуы мүмкін: кадрлар мен материалдық-техникалық құралдарды есепке алу, жеткізуші мен тапсырыс берушіні есепке алу, бухгалтерлік есеп және т.с.с.

Деректер банкі АЖ бір түрі, онда орталықтандырылған сақтау бір немесе бірнеше деректер қорынан біріктіріліп, өңделген ақпаратты жинақтау функциялары жүзеге асқан.

Жалпы жағдайда деректер банкі (ДБ) келесі құрамдас бөліктерден тұрады: деректер қоры (бірнеше қорлар), деректер қорын басқару жүйелері, деректер сөздігі, администратор, есептеуіш жүйелер және қызмет көрсетушілер. Аталған құрамдас бөліктерді және олармен байланысты негізгі ұғымдарды қарастырайық.

Деректер қоры (ДҚ) есептеу жүйелері жадында сақталатын және қарастырылып отырған саланың объектілерінің жағдайы мен өзара байланысын бейнелейтін деректердің арнайы түрде ұйымдасқан жиынтығы болып саналады.

Деректер қорында сақталатын логикалық құрылымды ***деректерді ұсыну үлгісі*** деп атайды. Деректерді ұсынудың (деректер үлгілері) негізгі үлгілеріне мыналар жатады: иерархиялық, желілік, реляциялық, постреляциялық, көпөлшемді және объектілі-бағытталған (2-бөлімді қараңыз).

Деректер қорын басқару жүйесі (ДҚБЖ) – көптеген пайдаланушылар АЖ деректер қорын бірлесіп құруына, жүзеге асыруына және қолдануына арналған тілдік және бағдарламалық құралдар кешені. Әдетте ДҚБЖ қолданылатын деректер үлгілері бойынша жіктейді. Мысалы, деректердің реляциялық үлгісіне негізделген ДҚБЖ реляциялық ДҚБЖ деп аталады.

Ең алғашқы ДҚБЖ келесі жүйелер жатады: IMS (IBM, 1968 ж.), IDMS (Cullinet, 1971 ж.), ADABAS (Software AG, 1969 ж.) және ИНЭС (ВНИАЖ И АН СССР, 1976 ж.). Қазіргі заманғы АЖ деректер қорын басқару жүйелері мыңнан астам.

Қосымша дегеніміз қолданбалы есептер үшін ақпаратты өңдеуді автоматтандыруды жүзеге асыратын бағдарлама немесе бағдарламалар кешені. Біз ДҚ қолданатын АЖ

қосымшаларын қарастырамыз. Қосымша ДҚБЖ ортасында немесе ортадан тыс құрылуы мүмкін – бағдарламалау жүйелері көмегімен, ДҚ қатынас құратын құралдарды пайдаланатын АЖ көмегімен, мысалы, Delphi немесе C++ Builder. ДҚБЖ ортасында құрылған қосымшаны көбінесе *ДҚБЖ қосымшалары* деп атайды, ал ДҚБЖ тыс қосымшаларды – *сыртқы* қосымшалар деп атайды.

Деректер қорымен жұмыс жасау үшін көбінесе ДҚБЖ құралдары жеткілікті және АЖ-де құру үшін бағдарламалауды қажет ететін қосымшаны қолдану керек емес. Қосымшаны ДҚ-мен маманданбаған пайдаланушылар жұмыс жасауын оңтайландыру үшін немесе ДҚБЖ интерфейсі пайдаланушыны қанағаттандырмаған жағдайда ғана құрады.

Деректер сөздігі (ДС) ДБ-ның ішкі жүйелері болып саналады; олар деректер құрылымдары, ДҚ файлдарының бір-бірімен өзара әрекеті, деректер типі және оларды бейнелеу форматы, деректердің пайдаланушыға қатыстылығы, қорғаныс коды және қатынас құруға шектеу және т.с.с ақпараттарды орталақтандырылған түрде сақтауға арналған.

Функционалды түрде ДС барлық ДБ-да болады, бірақ осы функцияларды атқаратын құрамдас бөліктер басқаша аталуы да мүмкін. Көбінесе ДС функциялары ДҚБЖ көмегімен жүзеге асады және жүйенің негізгі мәзірінен шақырылады немесе утилиталар көмегімен жүзеге асады.

Деректер қоры администраторы (ДҚА) дегеніміз ДҚ құру талаптарына, оны жобалау, құру, тиімді қолдану және сүйемелдеу үшін жауап беретін тұлға немесе қызметкерлер тобы. ДҚ пайдалану кезінде әдетте ДҚА ақпараттық жүйелердің жұмысын қадағалайды, оларды рұқсат етілмеген қатынас құрулардан қорғайды, ДҚ-ғы ақпараттардың артықтығын, қайшылықсыздығын, сақталуын және нақтылығын бақылайды. Жалғызпайдаланушылық ақпараттық жүйелер үшін ДҚА функциялары әдетте ДҚ қосымшасымен жұмыс жасайтын тұлғаларға жүктеледі.

Есептеу желісінде ДҚА, қағидаға сәйкес, *желі администраторымен* бірлесіп әрекет етеді. Желі администраторының міндеті: желідегі аппараттық-бағдарламалық құралдардың жұмыс істеуін бақылау, желіні қайта құру, құрылғылар істен шыққаннан кейін бағдарламалық қамтамасыз етілуді қалпына келтіру, профилактикалық іс-шаралар жүргізу және қатынас құруды шектеуді қамтамасыз ету.

Есептеу жүйесі (ЕЖ) дегеніміз қолданушының ақпаратты қабылдау, өңдеу және жөнелту процестерін автоматтандыруды қамтамасыз ететін өзара байланысты бірлесе жұмыс жасайтын ЭЕМ немесе процессорлар жиынтығы. ДБ негізгі қызметі деректерді сақтау және өңдеу болғандықтан, ЕЖ қолданылатын АЖ-дің орталық процессорының (ОП) лайықты қуаттылығы болуымен қатар, тікелей қатынас құруы үшін жедел және сыртқы жадынын қажетті көлемі болуы қажет.

Қызмет көрсетуші техникалық және бағдарламалық құралдарды жұмысқа қабілетті жағдайда ұстауды сүйемелдеу функциясын атқарады. Ол жоспарға сәйкес, сонымен қатар, қажетті жағдайда профилактикалық, регламенттік, қалпына келтіру жұмыстарын жүзеге асырады.

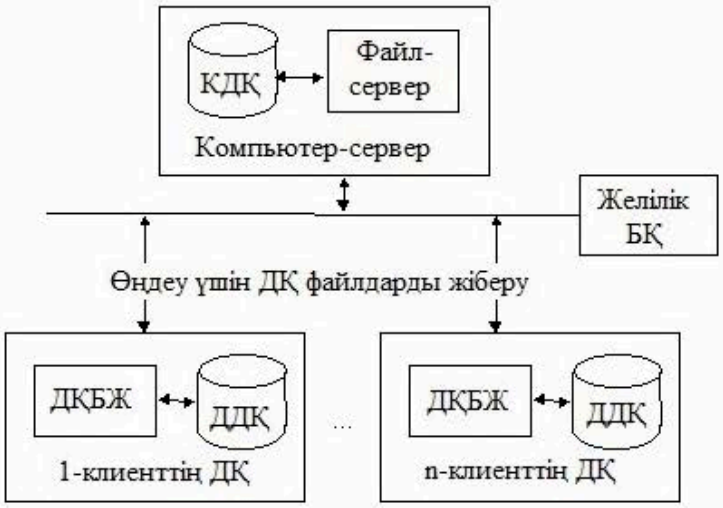
1.2. Ақпараттық жүйелер құрылымы

Ақпараттық жүйелердің (АЖ) жұмыс жасау тиімділігі көбінесе оның құрылымына тәуелді. Қазіргі уақытта клиент-сервер құрылымының келешегі бар. Кең таралған нұсқада корпоративті деректер қорын (КДҚ) және дербес деректер қорын (ДДҚ) қамтитын үлестірімді деректер қоры мен компьютерлік желінің болуын қажет етеді. КДҚ компьютер-серверде орналасқан, ДДҚ корпоративті ДҚ клиенттері болып табылатын бөлімше қызметкерлерінің компьютерлерінде орналасқан.

Компьютерлік желідегі нақты бір ресурстың сервері дегеніміз осы ресурсты басқаратын компьютер (бағдарлама), клиент дегеніміз осы ресурсты пайдаланатын компьютер (бағдарлама). Компьютерлік желінің ресурсы қызметін деректер қоры, файлдық жүйелер, баспаға шығару қызметі, поштальық қызметтер атқара алады. Сервер типі ол басқаратын ресурс түрімен анықталады. Мысалы, егер басқарушы ресурс деректер қоры болса, онда сәйкес деректер қоры сервері деп аталады.

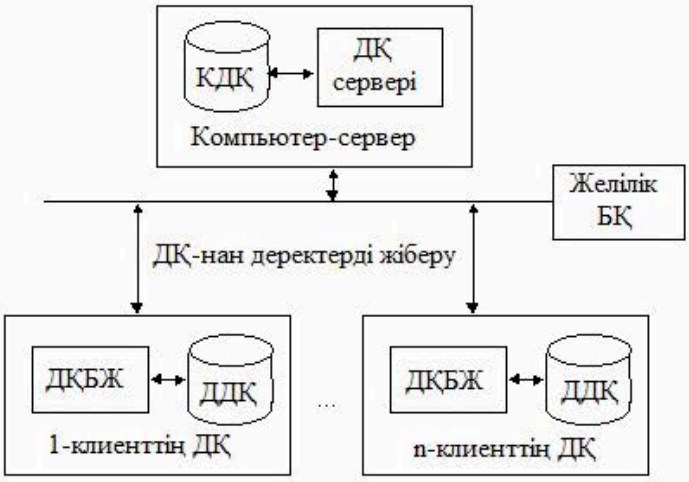
Клиент-сервер архитектурасы бойынша ақпараттық жүйелерді ұйымдастырудың ***құндылығы***: пайдаланушының дербес ақпаратпен жұмыс жасау кезінде жалпы корпоративтік ақпаратты ұжымдық пайдалана алуы, қызмет көрсетуі және ақпаратты *орталықтандырылған сақтаудың* тиімді үйлесімі. Клиент-сервер құрылымы әртүрлі нұсқада жүзеге асады.

Тарихи тұрғыда алдымен ***файл-сервер*** қолданылатын үлестірімді АЖ пайда болды. Мұндай АЖ пайдаланушы сұранысы бойынша деректер қоры файлы дербес компьютерлерге (ДК) жөнелтіледі, онда олар өңделеді. Бұл нұсқаның ***кемшілігі*** өңделетін деректерді жөнелтудің жоғары өнімділігі. Сонымен қатар, көбінесе артық ақпараттар жөнелтіледі: пайдаланушыға деректер қорынан қанша жазба қажет екенінен тәуелсіз, деректер қоры файлы толығымен жіберіледі(1.1-сурет).



1.1-сурет. Файл-серверлі АЖ құрылымы

Клиент-сервер архитектурасы бойынша тұрғызылған, деректер қоры серверін қолданатын үлестірімді АЖ құрылымы 1.2-суретте көрсетілген. Мұндай архитектурада деректер қоры сервері деректерді өңдеудің негізгі мөлшерінің орындалуын қамтамасыз етеді. Пайдаланушымен немесе қосымшамен құрылған сұраныстар ДҚ серверіне SQL тіліндегі нұсқаулар жиыны түрінде түседі. Деректер қоры сервері қажетті деректерді іздеу және табуды жүзеге асырады, содан соң олар пайдаланушы компьютеріне жөнелтіледі. Бұл тәсілдің ***артықшылығы:*** алдыңғы тәсілмен салыстырғанда жөнелтілетін деректер көлемінің аздығы.



1.2-сурет. Деректер қоры серверімен АЖ құрылымы

Клиент-сервер архитектурасы бойынша үлестірімді ДҚ тұрғызудың негізгі нұсқалары 4 бөлімде қарастырылған.

АЖ дербес ДҚ-мен және олармен жұмыс жасайтын қосымшаларды құру және басқару үшін Microsoft фирмасының Access және Visual FoxPro, Borland фирмасының Paradox сияқты ДҚБЖ қолданады.

Корпоративті ДҚ Microsoft SQL Server немесе Oracle Server сияқты ДҚ серверінің басқаруымен құрылады, сүйемелденеді және жұмыс жасайды.

Мекеменің көлемі және қойылған тапсырманың ерекшелігіне байланысты ақпараттық жүйелердің төмендегідей үйлесімі болуы мүмкін:

- бірлескен немесе дербес қорларды қамтитын компьютер-сервер;
- компьютер-сервер және ДДҚ-мен дербес компьютерлер;
- бірнеше компьютер-серверлер және ДДҚ-мен дербес компьютерлер

Клиент-сервер қолдану архитектурасы кәсіпорынның ақпараттық жүйелерін біртіндеп өсіруге мүмкіндік береді, алдымен, кәсіпорынның өсуі бойынша; екіншіден, ақпараттық жүйелердің өзінің өсуі бойынша.

Жалпы ДҚ-ын бірлескен ДҚ және дербес ДҚ-на жіктеу ДҚ жобалау қиыншылығын орталықтандырылған нұсқаға қарағанда азайтуға мүмкіндік береді, яғни жобалау кезінде кететін қателіктерді азайтады және жобалау құнын төмендетеді.

Ақпараттық жүйелерде ДҚ қолданудың басты ***құндылығы*** деректермен қамтамасыз етудің қолданбалы бағдарламалардан тәуелсіздігі. Бұл пайдаланушыларға физикалық деңгейде деректерді ұсыну мәселелерімен айналыспауға мүмкіндік береді: деректерді жадыда орналастыру, оларға қатынас құру және т.б.

Мұндай тәуелсіздік ДҚ-на логикалық (пайдаланушылық) және физикалық деңгейде деректердің көпдеңгейлі ұсынылуы ДҚБЖ сүйемелдеуі нәтижесінде жүзеге асады. ДҚБЖ және логикалық деңгейде деректерді ұсынуды қамтамасыз ету арқасында ДҚ концептуальды (ұғымдық) үлгілері оның ЭЕМ жадысында физикалық деңгейде орналасуынан айырмашылығын көрсетті.

1.3. Деректер қорын басқару жүйелері

Бұл бөлімде ДҚБЖ жіктелуі және олардың негізгі функциялары қарастырылады. Негізгі жіктелу белгілері ретінде келесілерді қолдануға болады: бағдарламалар түрлері, пайдалану өзгешелігі, деректер үлгісі. Аталған белгілер ДҚБЖ-ны мақсатты таңдау және өңделетін ақпараттық жүйелерді тиімділігі пайдалануға елеулі ықпал етеді.

ДҚБЖ жіктеу. Жалпы жағдайда ДҚБЖ дегеніміз ДҚ құру, жұмыс жасау және пайдалану процесстерін сүйемелдейтін кез-келген бағдарламалық өнім.

Қазіргі заманғы қандай бағдарламалар ДҚ-на қатысты және олар қандай деңгейде деректер қорымен байланысты екендігін қарастырайық.

ДҚБЖ-не жататын бағдарламалардың негізгі түрлері мыналар:

- толық функционалдық ДҚБЖ;
- ДҚ серверлері;
- ДҚ клиенттері;
- ДҚ-мен жұмыс жасау бағдарламаларын құру құралдары.

Толық функционалдық ДҚБЖ (ТФДҚБЖ) дегеніміз алдымен үлкен машиналар үшін, содан соң мини-машиналар және ДЭЕМ үшін пайда болған дәстүрлі ДҚБЖ. Барлық ДҚБЖ арасында заманауи ТФДҚБЖ неғұрлым жиі кездеседі және өздерінің мүмкіндіктері бойынша қуатты. ТФДҚБЖ-ға келесі бағдарламалық кешендер жатады: Clarion Database Developer, DataEase, DataFlex, dBase IV, Microsoft Access, Microsoft FoxPro және Paradox R:BASE.

Әдетте ТФДҚБЖ дамыған интерфейске ие, ол командалар мәзірі арқылы ДҚ-мен негізгі әрекеттерді орындауға мүмкіндік береді: құру және кесте құрылымын өзгерту, деректерді енгізу, сұраныстарды қалыптастыру, есеп беруді жүзеге асыру, оларды баспаға шығару және т.с.с. Сұраныстар құру және есеп беру үшін бағдарламалау міндетті емес, QBE (Query By Example – үлгі бойынша сұраныстарды қалыптастыру, 3.8 бөлімді қараңыз) тілін қолдану тиімді. Көптеген ТФДҚБЖ бағдарламалау құралдары үшін кәсіби бағдарлама құрушылар қызметін пайдаланады.

Қайсібір жүйелер көмекші және қосымша құралдар ретінде ДҚ сызбаларын жобалау немесе CASE-ішкі жүйелерді пайдаланады. Басқа ДҚ-на немесе SQL-серверлері деректеріне қатынас құруды жүзеге асыру үшін толық функционалдық ДҚБЖ факультативті модульдарды қолданады.

ДҚ серверлері желілерде деректерді өңдеу орталықтарын ұйымдастыруға арналған. ДҚ бұл тобы қазіргі уақытта азырақ, бірақ олардың саны біртіндеп өсіп келеді. ДҚ серверлері функциялары әдетте SQL операторлары көмегімен басқа бағдарламалармен сұралатын деректер қорын басқару арқылы жүзеге асады.

ДҚ серверлері мысалдары келесі бағдарламалар: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress).

ДҚ серверлері үшін ***клиенттік бағдарламалар*** ретінде жалпы жағдайда, әртүрлі бағдарламалар қолданылады: ТФДҚБЖ, электрондық кестелер, мәтіндік процессорлар, электрондық почта бағдарламалары және т.б. Сонымен қатар "клиент-сервер" жұбының элементтері бір немесе әртүрлі бағдарламалық қамтамасыз етуді өндірушілердің өнімі болуы мүмкін.

Клиенттік және серверлік бөлімдер бір фирмамен орындалған жағдайда атқаратын қызметтерді бөлу олардың арасында, әрине, рационалды жүзеге асады. Басқа жағдайларда, әдетте, деректерге қатынас құруды "кез-келген бағамен" жүзеге асыру мақсаты көзделеді. Мұндай байланысқа толық функционалдық ДҚБЖ біреуі сервер ролін атқаратын, ал екінші ДҚБЖ (басқа өндірушінің) – клиент ролін атқаратын жағдайы мысал болады. SQL Server (Microsoft) ДҚ сервері үшін клиенттік (фронтальды) бағдарламалар ретінде көптеген ДҚБЖ қарастырылады, олар: dBASE IV, Blyth Software, Paradox, DataEase, Focus, 1-2-3, MDBS III, Revelation және басқа.

ДҚ-мен жұмыс жасау бағдарламаларын құру құралдары келесі әртүрлі бағдарламалар үшін қолданылады:

- клиенттік бағдарламалар;
- ДҚ серверлері және олардың жекелеген құрамдас бөліктері;

- пайдаланушының қосымшалары.

Бірінші және екінші түрлі бағдарламалар неғұрлым аз, себебі олар негізінен жүйелік бағдарлама құрушыларға арналған. Үшінші түрдегі бағдарламалық кешендер анағұрлым көп, бірақ толық функционалдық ДҚБЖ-мен салыстырғанда аз.

Пайдаланушының қосымшаларын құру құралдарына бағдарламалау жүйелері жатады, мысалы Clipper, әртүрлі бағдарламалау тілдері үшін көптүрлі бағдарламалар кітапханасы, сонымен қатар құруды автоматтандыру дестелері бар (сонымен қатар клиент-сервер типті жүйелер). Неғұрлым кең таралғандар ретінде келесі құрал-саймандық жүйелерді атауға болады: Delphi және Power Builder (Borland), Visual Basic (Microsoft), SILVERRUN (Computer Advisers Inc.), S-Designor (SDP және Powersoft) және ERwin (LogicWorks).

Аталған құралдардан басқа, деректерді басқару және ДҚ-на қызмет көрсетуді ұйымдастыру үшін әртүрлі қосымша құралдар қолданылады, мысалға, транзакциялар мониторы (4.2 бөлімін қараңыз).

ДҚБЖ-ін пайдалану *сипаттамасы бойынша дербес* және *көппайдаланушы* деп жіктейді.

Дербес ДҚБЖ әдетте дербес ДҚ-мен және олармен жұмыс жасайтын арзан қосымшалармен жұмыс жасауға мүмкіндік береді. Дербес ДҚБЖ немесе олардың көмегімен құрылған қосымшалар көбінесе көппайдаланушылық ДҚБЖ-інің клиенттік бөлімдері ролін атқара алады. *Дербес* ДҚБЖ жататындар, мысалы, Visual FoxPro, Paradox, Clipper, dBase, Access және басқалар.

Көппайдаланушы ДҚБЖ ДҚ серверін және клиенттік бөлімді қамтиды және, қағидаға сәйкес, әртекті есептеуіш ортада қолданыла алады (әртүрлі типті ЭЕМ-мен және операциялық жүйелермен). Көппайдаланушылық ДҚБЖ-не мыналар жатады, мысалы, Oracle және Informix ДҚБЖ.

Қолданатын үлгілері бойынша ДҚБЖ (ДҚ-ның да) деректерін иерархиялық, желілік, реляциялық, объектілі-бағытталған және басқа типтерге жіктейді. Қайсібір ДҚБЖ бір уақытта бірнеше деректер үлгісін сүйемелдей алады.

Пайдаланушы тұрғысынан алғанда ДҚБЖ сақтау, өзгерістер енгізу (толтыру, өңдеу және өшіру) және ақпаратты өңдеу, сонымен қатар әртүрлі шығыс құжаттарын өңдеу және алу ***функцияларын*** жүзеге асырады.

Деректер қорында сақталатын ақпараттармен жұмыс жасау үшін ДҚБЖ бағдарламалар және пайдаланушыларға келесі екі бағдарламалау *тілдерін* ұсынады:

- деректерді сипаттау тілдері – жоғары деңгейлі процедуралық емес декларативтік типті тіл, логикалық құрылымды деректерді сипаттау үшін арналған;
- деректермен іс-әрекеттер жасау тілі – деректермен жұмыс жасау үшін негізгі әрекеттерді: енгізу, іс-әрекеттер жасау және деректерді сұраныс бойынша таңдау операцияларын орындауды қамтамасыз ететін құрылымдар жынтығы.

Аталған тілдер әртүрлі ДҚБЖ үшін өзгеше болуы мүмкін. Неғұрлым кең таралған екі стандартты тілдер: QBE (Query By Example) – үлгі бойынша сұраныстар тілі және SQL (Structured Query Language) – сұраныстар тілі. QBE көбінесе деректермен *іс-әрекеттер жасау* тілінің қасиеттеріне ие, SQL екі типтің – *сипаттау және* деректермен *іс-әрекеттер жасау* қасиеттерін қамтыған.

Жоғарыда аталған ДҚБЖ функциялары, өз кезегінде, неғұрлым төменгі деңгейдегі, ***төменгі деңгейлі*** деп аталатын келесі негізгі функцияларды қолданады:

- сыртқы жадыда деректерді басқару;
- жедел жадыда буферлерді басқару;
- транзакциялармен басқару;
- ДҚ өзгерістерді қадағалау журналы ;
- тұтастықты және ДҚ қауіпсіздігін қамтамасыз ету.

Аталған функциялардың заманауи ДҚБЖ-да жүзеге асуы ерекшеліктері мен қажеттіліктеріне қысқаша сипаттама берейік.

Сыртқы жадыда деректерді басқаруды жүзеге асыру әртүрлі жүйелерде, ресурстарды басқару деңгейінде және деректерді басқару алгоритмдерінің логикасы бойынша өзгеше болуы мүмкін. Көбінесе деректерді басқару әдістері мен алгоритмдері ДҚБЖ-ның "сыртқы жұмысы" болып табылады, және пайдаланушыға тікелей қатысы жоқ. Бұл функцияның сапалы түрде жүзеге асуы АЖ-мен жұмыс жасау тиімділігіне анағұрлым көп әсер етеді, мысалы, үлкен ДҚ, күрделі сұраныстармен, деректерді өңдеудің үлкен көлемімен жұмыс жасау кезінде.

Деректерді буферлеу қажеттілігі және жедел жадыда *буферлерді басқару* функцияларының жүзеге асуы, жедел жадының көлемінің сыртқы жадыға қарағанда аздығымен негізделеді.

Буферлер сыртқы және ішкі жадылар арасындағы ақпарат алмасуды жеделдетуге арналған жедел жадының облысы. Буферлерде уақытша ДҚ бөліктері, ДҚБЖ-не қатынас құру кезінде қолданылатын деректер сақталады.

Транзакциялар механизмі ДҚБЖ-де қордағы деректер тұтастығын қамтамасыз ету үшін қолданылады. ***Транзакция*** дегеніміз ДҚ-ғы деректемен орындалатын қайсібір бөлінбейтін операторлар тізбегі, олар басынан аяғына дейін ДҚБЖ-мен қадағаланады. Егер қандай да бір себептермен (құрылғының тоқтап қалуы немесе істен шығуы, бағдарламалық қамтамасыз етудегі қателіктер) транзакция аяқталмай қалса, онда ол жойылады.

Транзакцияға тән негізгі үш қасиеттер:

- атомарлық (транзакцияға кіретін барлық операциялар орындалады немесе бірде-біреуі орындалмайды);
- сериялық (бір мезетте орындалатын транзакциялардың өзара әсері жоқ);
- төзімділік (тіпті жүйелердің күйреуі транзакциялардың айқындалған нәтижелерінің жоғалуына әкелмейді).

Транзакцияға банк жүйелеріндегі бір шоттан басқа бір шотқа ақша аудару мысал бола алады. Бұл жерде, ең болмағанда, екі қадамдық үдеріс қажет. Алдымен бір шоттан ақшаны алады, содан соң оны басқа шотқа салады. Егер ең болмағанда, бір әрекет ойдағыдай орындалмаса, онда операция нәтижесі дұрыс емес болады және шоттар арасындағы теңгерім бұзылады.

Транзакцияларды бақылау жалғызпайдаланушылық және көппайдаланушылық ДҚБЖ үшін маңызды, онда транзакциялар параллель орындалады. Соңғы жағдайда транзакциялар сериялығы туралы айтылады. С*ериялық* дегеніміз орындалу жоспары қосынды нәтиженің олардың тізбектей орындалуының нәтижесіне эквивалент болатындай құрылғанын айтамыз.

Параллель орындалу кезінде транзакциялар қоспасында шиеленістер болуы мүмкін, олардың шешімі ДҚБЖ функциясы болып табылады. Мұндай жағдайлар кездескен кезде әдетте бір немесе бірнеше транзакциялармен жүргізілген өзгерістерді болдырмау жолымен «шегінулер» жүзеге асады.

ДҚ өзгерістерді қадағалау журналы (өзгерістер журналы) ДҚБЖ үшін құрылғының тоқтап қалуы немесе істен шығуы, бағдарламалық қамтамасыз етудегі қателіктер кезінде қордағы деректерді сақтау сенімділігін қамтамасыз ету үшін орындалады.

ДҚБЖ журналы – бұл ерекше ДҚ немесе негізгі ДҚ бөлімі, ол пайдаланушыға қолжетімді емес және деректер қорына енгізілген барлық өзгерістерді жазу үшін қолданылады. Әртүрлі ДҚБЖ журналға әртүрлі деңгейлерге сәйкес өзгерістерді жазып отыруы мүмкін: сыртқы жады беттерінің түрлендірулер операцияларының сыртқы кішігірім өзгерістерінен бастап ДҚ түрлендірулер логикалық операцияларына дейін

(мысалы, жазба қосу, бағананы өшіру, өрістегі шамаларға өзгерістер енгізу) және тіпті транзакциялар.

ДҚ өзгерістер журналын жүргізу функциялары тиімді жүзеге асуы үшін журналдың өзінің жұмыс жасауы кезінде жоғары деңгейлі сақтау сенімділігін қамтамасыз ету және сүйемелдеу қажет. Кей жағдайда осы үшін жүйелерде журналдың бірнеше көшірмесі сақталады.

Тұтастықты қамтамасыз ету ДҚ-мен ойдағыдай жұмыс жасау үшін қажетті шарт, әсіресе желіде ДҚ пайдалану жағдайы үшін.

ДҚ тұтастығы дегеніміз онда зерттеу облысын толық, қайшылықсыз және адекватты бейнелейтін ақпараттар сақталғанын білдіретін деректер қорының қасиеті. ДҚ тұтастығын сүйемелдеу тұтастығын тексеруді және деректер қорында қайшылықтар табылғанда оны қалпына келтіруді қамтиды. ДҚ тұтастығы ДҚ-ның *тұтастыққа шектеу қою* шарттары көмегімен сипатталады, ол шарттарды қорда сақталатын деректер қанағаттандыруы керек. Мұндай шарттарға келесі түрдегі мысалдар келтіруге болады: объектілер атрибуттарының мүмкін мәндер аралығына шектеу қою, ДҚ-да сақталатын мәліметтер немесе реляциялық ДҚ кестелеріндегі қайталанатын жазбалардың болмауы.

Қауіпсіздікті қамтамасыз ету ДҚБЖ-ның қолданбалы бағдарламалары мен деректерін шифрлау, құпия сөзбен қорғау, деректер қоры мен оның жекелеген элементтеріне (кестелерге, формаларға, есеп берулерге және т.б.) қатынас құруға шектеу қою деңгейін сүйемелдеу арқылы жүзеге асады.

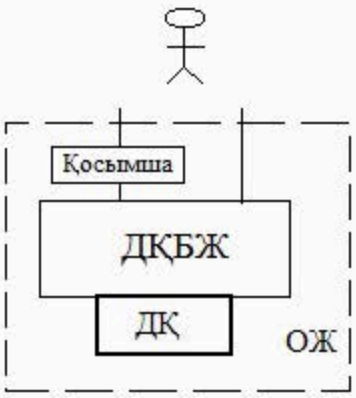
1.4. Жергілікті ақпараттық жүйелер

Ақпараттық жүйелердің функционалдық бөлімдері бір немесе бірнеше компьютерлерде орналасуы мүмкін. Бір ДК-дегі АЖ құру нұсқаларын қарастырайық. Сәйкес АЖ әдетте жергілікті немесе жалғызпайдаланушылық деп аталады (дегенмен бір компьютерде кезектесіп бірнеше пайдаланушы жұмыс жасауы мүмкін). АЖ ұйымдастырудың неғұрлым күрделі нұсқалары 4-бөлімде қарастырылады.

Жергілікті АЖ-мен бір компьютерде қайсыбір операциялық жүйелерде (ОЖ) жұмыс жасауды ұйымдастыру келесі бағдарламалық құралдарды пайдалану арқылы жүзеге асуы мүмкін:

- "толық" ДҚБЖ;
- ДҚБЖ қосымшасы және "қысқартылған" ядро;
- тәуелсіз қосымша.

Алғашқы әдіс әдетте барлық ДҚБЖ компьютердің дискілік жадыда орналасқан жағдайында пайдаланылады және ол қосымшаны түзету үшін жиі қолданылады (1.3-сурет).

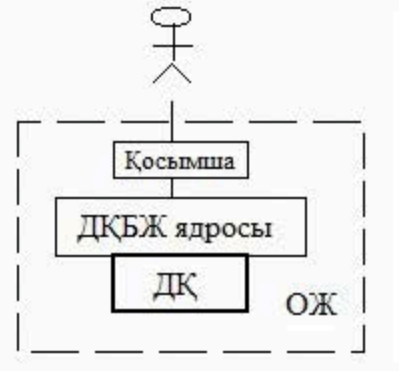


1.3-сурет. Қосымшаны қолдану және ДҚБЖ

Пайдаланушының ДҚ-мен өзара әрекеті пайдаланушының (терминалдық) ДҚБЖ интерфейсі арқылы немесе қосымша көмегімен тікелей жүзеге асады. Қосымшалар *талдаулар* режимінде орындалады (1.5-бөлімін қараңыз).

Сызбаның негізгі ***құндылығы*** – өңдеу қарапайымдылығы және ДҚ мен қосымшаларды дамыған сәйкес құралдармен өңдеу, сервистік құралдармен сүйемелдеу. Бұл сызбаның ***кемшілігі*** ДҚБЖ бағдарламаларын дискілік жадыда сақтау шығындары болып табылады. ДҚБЖ (1.4) ядромен қосымшалар келесі мақсаттарға жету үшін қолданылады:

- қатқыл диск және жедел жадыдағы ДҚБЖ-ның алатын көлемін азайту;
- қосымшаның жұмыс жылдамдығын арттыру;
- қосымшаны пайдаланушының түрлендіруінен қорғау (әдетте ядро қосымшаны өңдеу құралдарын қамтымайды).

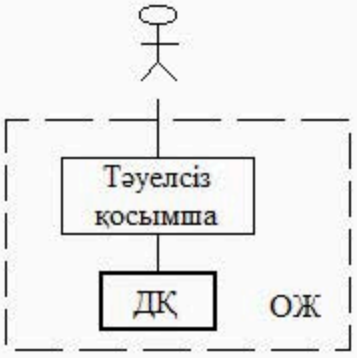


1.4-сурет. Қосымшаны қолдану және ДҚБЖ ядросы

Бұған FoxBase+ жүйелерінің FoxRun модульдерін қолдану мысал бола алады. Заманауи ДҚБЖ арасынан, Microsoft Access Developer's Toolkit қосымша дестесін қамтитын, Microsoft Access атауға болады. Оның көмегімен дискімен тасымалданатын "қысқартылған" (run-time) Microsoft Access нұсқасын құруға болады.

ДҚБЖ ядросын пайдалану ***құндылықтары*** ДҚБЖ-ның толық нұсқасымен салыстырғанда: компьютердің жады ресурстарын аз пайдаланады, қосымшалар жұмысын жылдамдатады және қосымшаларды түрлендірулерден қорғауға мүмкіндік береді. Негізгі ***кемшіліктері:*** ДҚБЖ ядросын сақтау үшін дискілік жадыда көп орын алатыны және қосымшамен жұмыс жасау жылдамдығының төмендігі (қосымшалар *талдаулар* жолымен жүзеге асады).

Үшінші әдісте АЖ ұйымдастырудың негізгі бағдарламасы алдын-ала *компиляцияланады* – орындалатын машиналық командалар тізбегіне айналады. Нәтижесінде орындалуға дайын тәуелсіз, жұмыс жасау үшін ДҚБЖ-ны да, ядроны да қажет етпейтін бағдарлама алынады (1.5). Деректерді сақтау және өңдеу негізгі функцияларының орындалуы тұрғысынан қарағанда мұндай бағдарламаның ДҚБЖ немесе оның ядросын басқару жұмысын атқаратын қосымшалардан айырмашылығы аз екендігін ескере кетейік.



1.5-сурет. Тәуелсіз қосымшаны қолдану

Бұл нұсқаның негізгі ***құндылықтары*** алдыңғы екеуімен салыстырғанда компьютердің сыртқы және жедел жадысын үнемдейді, қосымшаның орындалуын жылдамдатады және қосымшаны түрлендірулерден (дизассемблирования жағдайы мен өз кодын кірістіру және сол сияқтыларды есептемегенде) толықтай қорғайды. Кемшіліктері ретінде қосымшаны өңдеудің қиындығын, ДҚ-мен қызмет атқару кезінде ДҚБЖ-ның стандартты құралдарын қолдану мүмкіндігінің жоқтығын атауға болады.

1.5. Қосымшаларды құру және орындау әдістері

Заманауи ДҚБЖ деректер қорымен қосымшаны өңдеусіз жұмыс жасауға негізделген көптеген тапсырмаларды шешуге мүмкіндік береді. Дегенмен, қосымшалар құруды қажет ететін жағдайлар бар. Мысалы, егер деректерді бұрмалауды автоматтандыру талап етілсе, терминалдық ДҚБЖ интерфейсі жеткіліксіз дамыған болса, немесе ақпараттарды өңдеудің стандартты функциялары пайдаланушыны қанағаттандырмаса. Қосымшаны өңдеу үшін ДҚБЖ-ның бағдарламалық интерфейсі болуы керек, оның негізін сәйкес бағдарламалау тілінің функциялары және/немесе процедуралары қамтиды.

Қолданыстағы ДҚБЖ қосымшаны өңдеудің келесі технологияларын (олардың комбинациясын) сүйемелдейді:

- бағдарламаларды қолмен кодтау (Clipper, FoxPro, Paradox);
- генераторлар көмегімен қосымшалар мәтінін құру (FoxPro-да FoxApp, Paradox-да Personal Programmer);
- визуальды бағдарламалау әдістерімен дайын қосымшаны автоматты түрде іздестіру (Delphi, Access, Paradox for Windows).

Қолмен кодтау кезінде бағдарламалар қосымшасының бағдарлама мәтінін қолмен тереді, содан соң оны орындауға жібереді.

Генераторларды қолдану қосымшаны құруды жеңілдетеді, себебі мұнда бағдарламалар кодын қолмен термей алуға болады. Қосымшалар генереторы қосымшаның негізгі элементтерін (мәзір, экрандық формалар, сұраныстар және т.б.) құруды жеңілдетеді, бірақ көбінесе қолмен кодтауды толықтай алмастыра алмайды. *Қосымшаны визуальды бағдарламалау* құралдары қосымша генераторларын пайдалану мақсатын бұдан әрі дамытты. Қосымшалар сонымен қатар ыңғайлы интегралдық орта көмегімен дайын "құрылыс блоктарынан" тұрады. Қажеттілік кезінде бағдарлама құрушы қосымшаларға өз кодтарын енгізе алады. Интегралдық орта, қағидаға сәйкес, қосымшаны құру, жүзеге асыру және түрлендірулердің қуатты құралдарын ұсынады. Визуальды бағдарламалау құралдарын қолдану, алдыңғы екі әдіспен салыстырғанда, аз

уақыт аралығында неғұрлым сенімді, жағымды және тиімді қосымшалар құруға мүмкіндік береді.

Құрылған қосымшалар әдетте операциялық жүйелердің бір немесе бірнеше файлдарынан тұрады.

Егер қосымшаның негізгі файлы орындалатын файл (мысалы, exe-файл) болса, онда бұл қосымшалар, *тәуелсіз қосымшалар* болады, олар ДҚБЖ ортасынан дербес орындалады. Практикада тәуелсіз қосымшаны алу бағдарламалардың бастапқы мәтіндерін әртүрлі әдістермен компиляциялау жолымен жүзеге асады: мәтінді қолмен теру жолы арқылы, сонымен қатар қосымша генератор немесе визуальды бағдарламалау ортасы көмегімен.

Тәуелсіз қосымшаларды, мысалы, ДҚБЖ FoxPro және визуальды бағдарламалау Delphi арқылы алады. Ескере кетейік, әдетте Delphi құралдар көмегімен тәуелсіз қосымшалар құрмайды, себебі бұл қиын процесс, ол ДҚБЖ ядросы ролін атқаратын BDE деректер қоры процессорын (Borland DataBase Engine) жұмылдырады. Дербес ЭЕМ үшін қосымшаны өңдеу құралдарының алғашқыларының бірі Clipper жүйелері болып табылады, олар "таза компилятор" деп саналады.

Көптеген жағдайларда қосымшалар ДҚБЖ ортасынсыз орындала алмайды. Қосымшаны орындау кезінде алдымен, ДҚБЖ қосымша файлдары құрамын талдайды (жеке жағдайда – бұл бастапқы бағдарламалар мәтіні) және автоматты түрде орындалуы қажетті машиналық командалар тізбегін береді. Басқаша айтқанда, қосымшалар *талдаулар* әдісімен орындалады.

Талдаулар режимі көптеген заманауи ДҚБЖ-да жүзеге асқан, мысалы, Access, Visual FoxPro және Paradox, сонымен қатар FoxBase және FoxPro.

Сонымен қатар, компиляция және талдау арасындағы аралық нұсқаны қолданатын жүйелер бар – олар *псевдокомпиляция* деп аталады. Мұндай жүйелерде бастапқы бағдарлама компиляциялау жолымен өзгертіледі де аралық код (псевдокод) дискіге жазылады. Қайсібір жүйелерде тіпті редактрлеу рұқсат етіледі, бірақ псевдокомпиляцияның негізгі мақсаты – оны талдаулар процесін жылдамдататын бағдарламалау түріне түрлендіру. Мұндай әдіс DOS басқаруымен жұмыс жасайтын ДҚБЖ-да кеңінен қолданылды, мысалы, Foxbase+ және Paradox 4.0/4.5 for DOS.

Windows басқаруымен жұмыс жасайтын ДҚБЖ-да псевдокод қосымшаларды түрлендіруге тыйым салу үшін жиі қолданылады. Бұл жұмыс жасайтын бағдарламаларды кездейсоқ немесе қасақана бұзудан қорғау үшін пайдалы. Мысалы, мұндай әдіс қолданылған Paradox for Windows ДҚБЖ-да құрылған экрандық формалар және есеп беруді редактрлеуге болмайтын сәйкес объектілерге түрлендіруге болады.

Қайсібір ДҚБЖ пайдаланушыға қосымшаны өңдеу нұсқасын таңдау мүмкіндігін береді: ДҚБЖ бағдарламалық коды ретінде немесе тәуелсіз бағдарламалар ретінде.

Тәуелсіз қосымшаны қолдану ***құндылығы*** машиналық бағдарламаларды орындауға кеткен уақыт, талдауларға кеткен уақытқа қарағанда, әдетте аз болып табылады. Ондай қосымшаларды әлсіз машиналарда және "жап-жаңа" жүйелер орнату кезінде қосымшаларды пайдаланушылардың түзетулерінен жауып қою қажет болған жағдайларда қолданған жөн.

Түсіндірмелік қосымшаны қолданудың басты құндылығы оны түрлендірудің оңайлығы болып табылады. Егер дайын бағдарлама жиі өзгерістерге ұшыраса, онда оларды енгізу үшін құрал-саймандық жүйелер қажет, яғни ДҚБЖ немесе ұқсас орта. Түсіндірмелік қосымша үшін мұндай құрал әрқашан қолжетімді, бұл өте тиімді.

Басқа маңызды ***құндылығы*** жүйелер талданған болып табылады, әдетте жақсы ДҚБЖ-лардың деректер тұтастығын бақылау және рұқсат етілмеген қатынас құрулардан қорғау үшін қуатты құралдары бар, бұл компиляциялық типті жүйелерге

тән емес. Соңғыларда аталған функцияларды қолмен бағдарламалауға тура келеді, немесе администраторлардың ар-ұятына қалдырамыз.

Қосымшаны өңдеу үшін құралдарды таңдау кезінде негізгі үш факторды ескеру керек: компьютердің ресурстары, қосымшаның ерекшеліктері (бағдарламалар функцияларын түрлендірулер қажеттілігі, құруға кеткен уақыт, қатынас құруды бақылау қажеттілігі және ақпараттардың тұтастығын сүйемелдеу) және өңдеу мақсаты (шеттетілген бағдарламалық өнім немесе өзінің күнделікті қызметін автоматтандырған жүйелер).

Заманауи компьютері бар және қиын емес қосымшаларды құруды жоспарлаған пайдаланушы үшін *интерпретацияланған* типті ДҚБЖ сай келеді. Ескере кетейік, ондай жүйелер жеткілікті дәрежеде қуатты, жоғары деңгейлі құралдары бар, өңдеу және жүзеге асыру үшін қолайлы, өңдеуді жылдам орындайды және тиімді сүйемелдеу мен қосымша жасау іс-әрекеттерін қамтамасыз етеді.

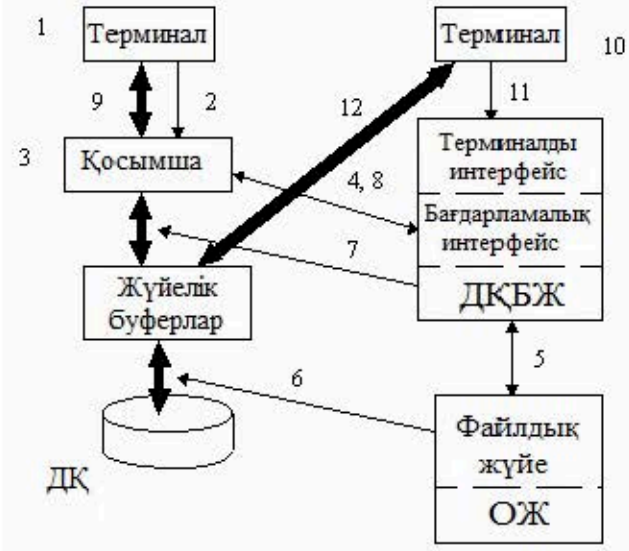
Сипаттамасы төмен компьютерді қолданған кезде қосымшаны өңдеудің тәуелсіз құралдары бар жүйелерді таңдаған дұрыс. Сонымен қатар, қосымшалардағы аздаған өзгерістердің бағдарламалау, компиляция және жүзеге асыру кезеңдерінің циклдік қайталануына апарып соқтыратыны есте болған жөн. Тәуелсіз қосымшаның орындалуы мен қосымша режимінде орындалудың айырмашылығы миллисекунд қана.

1.6. ДҚ жұмыс жасауда деректер алмасу сызбасы

Кез-келген санатты пайдаланушы (ДҚ администраторы, қосымшаны құрушы бағдарлама, әдеттегі пайдаланушы) үшін тапсырманың сауатты шешімі ДҚ-мен жұмыс жасау кезінде ОЖ-де жүзеге асатын есептеу процесі болып табылады. Осы процестің сыртқы механизмдерін АЖ ұйымдастырудың пайдаланушы бір ДК-мен, ДҚБЖ бағдарламаларының "толық" нұсқасымен (1.3) жұмыс жасайтын кездегі неғұрлым жалпы жағдайындағы мысалында ашайық. 1.4 және 1.5-суреттердегі көрсетілген нұсқаларды жеке жағдайлар деп есептеуге болады.

Пайдаланушының деректер қорымен жұмыс жасауы кезінде оның құрамымен келесі негізгі операциялар орындалады: деректерді таңдау, кірістіру немесе қосу, түрлендіру (алмастыру) және өшіру. Жекелеген пайдаланушылар және дербес ДҚБЖ арасында неғұрлым жиі қолданатын *деректерді таңдау* операциясындағы деректермен алмасу процессі қалай жүзеге асатынын қарастырайық. Пайдаланушы мен ДҚ арасында деректермен алмасудың басқа операциялардан айырмашылығы аз.

Сызбалық тұрғыда пайдаланушының ДҚ-мен жұмыс жасау кезіндегі деректермен алмасуын 1.6-суретте көрсетілгендей суреттеуге болады, онда қарапайым бағыттауыштармен басқару бойынша байланыстар, жуандатылған бағыттауыштармен ақпараттар арасындағы байланысар белгіленген.



1.6-сурет. ДҚ жұмыс жасауда деректер алмасу сызбасы

Пайдаланушының ДҚ-мен қосымша көмегімен өзара әрекеті кезеңін келесі негізгі кезеңдерге бөлуге болады.

1. Терминалды пайдаланушы (1) қосымшалармен сұқбаттасу процесінде ДҚ-ның қайсібір деректеріне сұраныста (2) тұжырымдайды.

2. Қосымшалар (3) бағдарламалық деңгейде іс-әрекеттер жасау тілі құралдарымен ДҚБЖ деректеріне сұраныс (4) жасайды.

3. Өздерінің жүйелік басқарушы блоктар және кестелерін пайдаланып, ДҚБЖ көмегімен *деректер сөздігі* талап етілген деректердің орналасқан орнын анықтайды және олар үшін ОЖ-ге қатынас құрады (5).

4. ОЖ файлдық жүйесіне қатынас құру әдістерінің бағдарламалары сыртқы жадыдан ізделінді деректерді оқиды (6) және оларды ДҚБЖ жүйелік буферіне орналастырады.

5. Алынған деректерді талап етілген форматқа сай өзгерте отырып, ДҚБЖ оларды (7) сәйкес бағдарламалар облысына жібереді және қандай да бір жолмен операцияның аяқталғандығы туралы белгі береді (8).

6. Қосымшалар қорынан (3) деректерді таңдау нәтижелері пайдаланушының терминалында (1) бейнеленеді (9).

Пайдаланушының жұмыс жасау кезінде ДҚБЖ (қосымшасыз) сұқбаттасу режимі пайдаланушының ДҚ-мен өзара әрекетін жеңілдетеді. Оны келесі кезеңдерге бөліп көрсетуге болады.

1. Терминалды пайдаланушы (10) сұраныстар тілінде ДҚБЖ тұжырымдайды, мысалы QBE, (11) байланыс бойынша қордан қайсібір деректерді таңдау талап етіледі

2. ДҚБЖ талап етілген деректердің орналасқан орнын анықтайды және оларды алу үшін ОЖ-ге қатынас құрады (5), ол сыртқы жадыдан (6) ізделінді деректерді табады және оларды ДҚБЖ жүйелік буферіне орналастырады.

3. Ақпарат жүйелік буферлерден талап етілген форматқа өзгертіледі (12), содан соң пайдаланушының терминалында (10) бейнеленеді (13).

Ескере кетейік, суреттелген сызба бір пайдаланушының жекелеген ДЭЕМ-де ДҚБЖ қалай жұмыс жасайтынын түсіндіреді.

Егер компьютер және ОЖ көппайдаланушының жұмыс режимін сүйемелдесе, онда мұндай есептеуіш жүйелерде *көппайдаланушылық ДҚБЖ* жұмыс жасау мүмкін. Соңғысы, жалпы жағдайда, бір уақытта тікелей ДҚ-мен немесе қосымшалармен

(олардың әрқайсысы бір немесе бірнеше пайдаланушылардың жұмысын сүйемелдеуі мүмкін) жұмыс жасайтын бірнеше пайдаланушыларға қызмет көрсетуге мүмкіндік береді.

Кей жағдайда есептеуіш жүйелерге "шалғайдағы пайдаланушы" қосылады, олар ЭЕМ-нен қандай да бір қашықтықта орналасқан және онымен бір байланысу ортасы арқылы жалғасқан (ЭЕМ интерфейсі, байланыстың телефон каналы, радиоканал, оптико-талшықты желі және т.б.). Әдетте мұндай пайдаланушы бағдарламалық әдіспен әдеттегі жергілікті пайдаланушы ретінде жұмыс жасайды. ДҚБЖ, қағидаға сәйкес, мұндай ауыстыруды "байқамайды" және сұраныстармен әдеттегідей жұмыс жасайды.

Көппайдаланушылық ДҚБЖ-да әртүрлі операциялар орындау кезінде параллель түрде жоғарыда сипатталған процестерге ұқсас процесстер пайда болады.

Бірнеше ақпарат көздеріне параллель қызмет көрсету кезінде ДҚБЖ сұраныстарды (пайдаланушылар және қосымшадан), өз ресурстарын және ЭЕМ ресурстарын қолданып жоспарлайды.

Көппайдаланушы ДҚБЖ үлкен және орташа ЭЕМ-де жиі қолданылады, онда негізгі режимде ресурстарға топпен қатынас құруға болады.

Дербес ЭЕМ-де пайдаланушы әдетте жалғыз, бірақ әртүрлі баңдарламалармен және бір уақытта (нақтырақ айтқанда, кезектесіп) жұмыс жасайды. Кей жағдайда мұндай бағдарлама ДҚБЖ болады: ДҚБЖ-ның әртүрлі бағдарламалары немесе әртүрлі көшірмелері. Соңғы жағдай, мысалы, Access ДҚБЖ көмегімен әртүрлі деректер қорымен жұмыс жасау кезінде пайда болады.

Пайдаланушының бір уақытта бірнеше бағдарламалармен жұмыс жасау технологиясы Windows операциялық жүйесінде жақсы жүзеге асқан. Мұнда әрбір орындалатын бағдарламаның пайдаланушымен өзара әрекеті терезесі және бағдарламалар арасында ауыстырып қосудың қолайлы құралдары бар. Windows операциялық жүйесінде ДҚБЖ-мен жұмыс жасау кезінде пайдаланушылардың бірнеше сеанстарын сүйемелдеудің қажеттілігі жоқ.

Бақылау сұрақтары және тапсырмалар

1. Ақпараттық жүйелер түсінігі анықтамасын кең және тар мағынада беріңіз.
2. Деректер банкі дегеніміз не және оның құрамдас бөліктері?
3. ДҚБЖ мақсаты қандай?
4. Деректердің негізгі үлгілерін атаңыз.
5. Қосымшаға анықтама беріңіз, ол қандай жағдайларда құрылатындығын атаңыз.
6. Деректер сөздігі қызметін түсіндіріңіз.
7. Деректер қоры администраторы функцияларын атап шығыңыз.
8. Есептеу жүйесі не болып саналады?
9. Клиент-сервер архитектурасын сипаттаңыз және оның нұсқаларының жүзеге асуын атаңыз, артықшылықтары мен кемшіліктерін түсіндіріңіз.
10. Ақпараттық жүйелердің файл-серверлік құрылымын суреттеңіз.
11. Ақпараттық жүйелердің деректер қоры серверінің құрылымын суреттеңіз.
12. ДҚБЖ жататын бағдарламалардың негізгі түрлерін сипаттаңыз.
13. Қолданбалы тапсырмаларды шешу кезінде пайдаланушының деректер қорымен жұмысының негізгі әдістерін атаңыз.
14. Деректер қорымен жұмыс жасау кезінде қосымшаны құру технологиясын түсіндіріңіз.
15. Деректер қорымен жұмыс жасау кезінде қосымшаның орындалу әдістерін сипаттаңыз.

16. Пайдаланушының ДҚ-мен деректер алмасуы үшін қажетті деректерді өңдеудің келесі операцияларын суреттеңіз:

- таңдау,
- кірістіру,
- түрлендіру;
- өшіру.

17. Көппайдаланушылық ДҚБЖ сипаттамасын беріңіз.

2. ДЕРЕКТЕР ҮЛГІСІ ЖӘНЕ ТИПІ

Қорда сақталатын деректер нақты логикалық құрылымға ие – басқаша айтқанда, ДҚБЖ сүйемелдейтін қайсібір ***деректерді ұсыну үлгісі*** (деректер үлгісі) суреттеледі. Классикалық деректер үлгісіне мыналар жатады:

- иерархиялық,
- желілік,
- реляциялық.

Сонымен қатар, соңғы жылдарда келесі деректер үлгілері пайда болды және олар практикаға неғұрлым белсенді енгізілуде:

- постреляциялық,
- көпөлшемді,
- объектілі-бағытталған.

Деректердің басқа үлгілеріне негізделген, қолданыстағы үлгілердің аумағын кеңейтетін барлық мүмкін жүйелер құрылуда. Олардың санатында объектілі-реляциялық, дедуктивті-объектілі-бағытталған, семантикалық, концептуальды және бағытталған үлгілерді атауға болады. Осы модельдердің қайсібірі деректер қоры, білім қоры және бағдарламалау тілдерін интеграциялау үшін қолданылады.

Қайсібір ДҚБЖ бір уақытта бірнеше деректер үлгісін сүйемелдейді. Мысалы, ИНТЕРБАЗА жүйесінде қосымшада деректермен іс-әрекеттер жасау үшін желілік тіл қолданылады, ал пайдаланушы интерфейсі SQL және QBE тілдері арқылы жүзеге асқан.

2.1. Реляциялық үлгі

Деректердің реляциялық үлгісін IBM фирмасының қызметкері Эдгар Кодд ұсынған және ол қатынас (relation) ұғымына негізделген.

Қатынас кортеж деп аталатын элементтер жиыны болып саналады. Деректердің реляциялық үлгісі жан-жақты теориялық негізде келесі бөлімде қарастырылады. Қатынастар көрінісінің көрнекі формасы адам қабылдауы үшін үйреншікті екіөлшемді кесте болып табылады.

Кесте жолдар (жазба) және бағаналардан (колонкалар) тұрады. Кестелердің әрбір жолы бірдей құрылымға ие және өрістерден тұрады. Кестелер жолына кортеждер, ал бағаналарға – атрибутты қатынастар сәйкес келеді.

Кестелер көмегімен деректердің арасындағы байланыстың қарапайым түрін тиімді сипаттауға болады, нақтырақ айтқанда: бір объектіні (құбылыстар, болмыстар, жүйелер және басқалар), кестеде сақталатын және әрқайсысына кестелер жолы немесе жазбасы сәйкес келетін ақпараттарды ішкі объектілер жиынына бөлу. Сонымен қатар, ішкі объектілердің әрқайсысы жазбалар өрісі мәндеріне сәйкес сипатталатын бірдей құрылымға немесе қасиеттерге ие. Мысалы, кесте оқу тобы туралы мәліметтерден тұруы мүмкін, олардың әрқайсысы туралы келесі сипаттамалар белгілі: фамилия, аты және әкесінің аты, жынысы, жасы және білімі. Себебі бір ғана кестелер шеңберінде зерттеу облысының неғұрлым күрделі логикалық құрылым деректерін сипаттай алмаймыз, ол кестелер үшін *байланыстыру* қолданылады.

Сыртқы есте сақтау құрылғыларында деректердің физикалық орналасуы реляциялық қорда қарапайым файлдар көмегімен оңай жүзеге асады.

Деректердің реляциялық үлгісінің ***құндылығы*** оның ЭЕМ-де физикалық жүзеге асуының қарапайымдылығында, түсініктілігінде және ыңғайлылығында. Қарапайымдылығы және түсініктілігі үшін пайдаланушылар оларды кеңінен пайдаланады. Деректерді өңдеу тиімділігі техникалық тұрғыда жүзеге асты.

Реляциялық үлгілердің негізгі ***кемшіліктері*** келесілер: жекелеген жазбаларды сәйкестендірудің стандартты құралдарының болмауы және иерархиялық және желілік байланысты сипаттаудың қиындығы.

ДЭЕМ үшін шетелдік реляциялық ДҚБЖ мысалдары келесілер: dBaseIII Plus және dBase IY (Ashton-Tatem фирмасы), DB2 (IBM), R:BASE (Microrim), FoxPro алғашқы нұсқасы және FoxBase (Fox Software), Paradox және dBASE for Windows (Borland), FoxPro неғұрлым соңғы нұсқасы, Visual FoxPro және Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) және Oracle (Oracle).

Реляциялық типті ДҚБЖ жататын жүйелер: ПАЛЬМА (ИК АН УССР), сонымен қатар HyTech (МИФИ) жүйелері.

Ескере кетейік, реляциялық ДҚБЖ соңғы нұсқалары объектілі-бағытталған жүйелердің қайсібір қасиеттеріне ие. Ондай ДҚБЖ жиі объектілі-реляциялық деп аталады. Мұндай жүйелерге мысал ретінде Oracle 8.x өнімін атауға болады. Oracle 7.x нұсқасына дейінгі жүйелерді "таза" реляциялық деп есептейді.

2.2. Постреляциялық үлгі

Классикалық реляциялық үлгі кесте жазбаларының өрістерінде сақталатын деректердің тұтастығын жобалайды. Бұл ақпараттың кестеде алғашқы қалыпты формада (5.2-бөлім) берілгенін білдіреді. Кейбір жағдайларда бұл шектеу қосымшаның тиімді жүзеге асуына кедергі келтіреді.

Постдеректердің реляциялық үлгісі кесте жазбаларында сақталатын деректердің тұтастығына қойылған шектеуді алып тастайтын кеңейтілген реляциялық үлгі болып саналады. Постдеректердің реляциялық үлгісі көпмәнді өрістерді қолдануға рұқсат береді. Көпмәнді өрістің мәндер жиыны негізгі кестеге енгізілген дербес кесте болып саналады.

2.1-суретте жөнелтпе құжат және тауарлар туралы ақпараттарды салыстыру үшін реляциялық (а) және постреляциялық (б) модельдердің бірдей деректері келтірілген. INVOICES (жөнелтпе құжат) кестесі жөнелтпе құжат (INVNO) нөмірлері және сатып алушы нөмірі (CUSTNO) туралы деректерді қамтиды. INVOICE.ITEMS (жөнелтпе құжат-тауарлар) кестесінде әрбір жөнелтпе құжат туралы деректер болады: жөнелтпе құжат нөмірі (INVNO), тауар атауы (GOODS) және тауар саны (QTY). INVOICES кестесі INVOICE.ITEMS кестесімен INVNO өрісі бойынша байланысқан.

а)

INVOICES

INVNO	CUSTNO
0373	8723
8374	8232
7364	8723

INVOICE.ITEMS

INVNO	GOODS	QTY
0373	Сыр	3
0373	Балық	2
8374	Лимонад	1
8374	Сок	6
8374	Печенье	2
7364	Йогурт	1

б)

INVOICES

INVNO	CUSTNO	GOODS	QTY
0373	8723	Сыр	3
		Балық	2
8374	8232	Лимонад	1
		Сок	6
		Печенье	2
7364	8723	Йогурт	1

2.1-сурет. Деректердің реляциялық және постреляциялық моделі құрылымы

Суреттен реляциялық үлгімен салыстырғанда, постреляциялық үлгілерде деректер неғұрлым тиімді сақталатынын көреміз, ал өңдеу екі кестедегі деректерді байланыстыру операциясы арқылы орындалады. Дәлелдеу үшін 2.1-суретте реляциялық (а) және постреляциялық (б) модельдер үшін SQL тілінде қордың барлық өрістерінен деректерді таңдаудың SELECT операторы мысалға келтіріледі.

а)

```
SELECT
  INVOICES.INVNO, CUSTNO, GOODS, QTY
FROM
  INVOICES, INVOICE.ITEMS
WHERE
  INVOICES.INVNO=INVOICE.ITEMS.INVNO;
```

б)

```
SELECT
  INVNO, CUSTNO, GOODS, QTY
FROM
  INVOICES;
```

2.2-сурет. Реляциялық және постреляциялық модельдер үшін SQL операторы

Өріске енгізуді қамтамасыз етуден басқа постреляциялық үлгі қауымдастырылған көпмәнді өрістерді (көптеген топтар) сүйемелдейді. Қауымдасқан өрістер жиынтығы ***ассоциация*** деп аталады. Сонымен қатар, жолда ассоциацияның бір бағанының алғашқы мәні ассоциацияның барлық басқа бағаналарының алғашқы мәнімен сәйкес келеді. Осылайша барлық екінші бағана мәні және т.б. байланысады.

Кесте жазбаларындағы өрістің ұзындығы және санына тұрақты болу шарты қойылмайды. Бұл деректер және кесте құрылымының икемділігін білдіреді.

Себебі постреляциялық үлгі кестелерде қалыпқа сай емес деректерді сақтауға рұқсат береді, бұдан деректердің тұтастығын және қайшылықсыздығын қамтамасыз ету мәселесі туындайды. Бұл мәселе клиент-серверлік жүйелерде ДҚБЖ-ға сақталатын процедураларға ұқсас механизмдерді қосу арқылы шешіледі.

Өрістердегі мәндерді бақылау функциясын сипаттау үшін процедуралар (корреляциялар коды) құру мүмкіндігі бар, олар деректерге қатынас құрғанға дейін немесе кейін автоматты түрде шақырылады. Корреляциялар коды деректерді оқығаннан кейін бірден оларды өңдегенге дейін орындалады.

Постреляциялық үлгілер ***құндылығы*** бір постреляциялық кестемен байланысты реляциялық кестелер жиынтығының мүмкін көрінісі болып табылады. Бұл

ақпараттарды бейнелеудің жоғары деңгейін қамтамасыз етеді және оны өңдеудің тиімділігін арттырады. Постреляциялық үлгілерлердің ***кемшілігі*** сақталатын деректердің тұтастығын және қайшылықсыздығын қамтамасыз ету мәселелерін шешуде қиындық тудырады.

Біз қарастырған постдеректердің реляциялық үлгісін uniVers ДҚБЖ сүйемелдейді. Постдеректердің реляциялық үлгісіне негізделген басқа ДҚБЖ-ларға Bubba және Dasdb жүйелері де жатады.

2.3. Көпөлшемді үлгі

Деректердің қордағы ұсынылуының көпөлшемді тәсілдемесі реляциялық деректер қорымен бір уақытта пайда болды, бірақ нақты жұмыс жасайтын көпөлшемді ДҚБЖ (КДҚБЖ) осы уақытқа дейін өте аз болды. 90-шы жылдардың ортасынан бастап оларға жаппай қызығушылық артты.

1993 жылы реляциялық тәсілдеменің негізін қалаушылардың бірі Э. Коддтың бағдарламалық мақаласы түрткі болды. Онда OLAP (OnLine Analytical Processing – жедел талдамалық өңдеу) тобы жүйелеріне қойылатын 12 негізгі талаптар тұжырымдалған, олардың маңыздылары көпөлшемді деректердің тұжырымдамалық көрінісі және өңдеу мүмкіндіктерімен байланысты. Көпөлшемді жүйелер ақпаратты шапшаң өңдеу үшін талдау жасайды және шешім қабылдайды.

АЖ концепциясы дамуында келесі екі бағытты баса атауға болады:

- жедел (транзакциялық) өңдеу жүйелері;
- талдау жасаушылық өңдеу (шешім қабылдауды сүйемелдеу жүйелері) жүйелері.

Реляциялық ДҚБЖ ақпаратты *жедел* өңдейтін ақпараттық жүйелер үшін тағайындалған және осы облыста өте тиімді. Талдау жасаушылық өңдеу жүйелерінде олар өздерінің бірнеше қолайсыз және осал қырларын көрсетті. Неғұрлым тиімдісі көпөлшемді ДҚБЖ (КДҚБЖ) болып табылады.

Көпөлшемді ДҚБЖ интербелсенді талдау жасаушылық ақпаратты өңдеуге арналған тар шеңбердегі мамандандырылған ДҚБЖ болып табылады. Осы ДҚБЖ-да қолданылатын негізгі ұғымдарды ашайық: деректердің бірігушілігі, тарихшылдығы және болжамдылығы.

Деректердің *бірігушілігі* ақпараттарды әртүрлі деңгейде жалпылай қарастыруды білдіреді. Ақпараттық жүйелердегі ақпараттарды толық жете суреттеу үшін пайдаланушы оның деңгейінен тәуелді болады: талдау жасаушы, пайдаланушы-оператор, басқарушы, жетекші.

Деректердің *тарихшылдығы* деректердің және олардың өзарабайланысының өзгермейтіндігін жоғары деңгейде қамтамасыз етеді, сонымен қатар деректердің міндетті түрде уақытқа тәуелділігін жобалайды.

Деректердің өзгермейтіндігі оларды жүктеу, сақтау, индекстеу және таңдау мамандандырылған әдістерін өңдеуге қолдануға мүмкіндік береді.

Деректерді уақытша қосу таңдаудың мәні уақыт пен мерзімге тәуелді болатын сұраныстар үшін қажет. Деректерді уақыт бойынша реттеу қажеттілігі деректерді өңдеу және пайдаланушыға ұсыну процесінде ақпараттарды сақтау және оларға қатынас құру механизмдеріне талаптар жүктейді. Сұраныстарды өңдеу уақытын азайту үшін деректер әрқашан неғұрлым жиі сұралатын ретпен сұрыпталғаны дұрыс.

Деректердің *болжамдылығы* болжам жасау функциясын жүзеге асыруды және оларды әртүрлі уақыт аралығында қолдануды тұспалдайды.

Деректер үлгісінің көпөлшемділігі сандық деректерді шолудың көпөлшемділігін емес, ақпараттарды сипаттау кезінде және деректермен іс-әрекеттер жасау операцияларында көпөлшемді логикалық бейнелеуді білдіреді.

Реляциялық үлгімен салыстырғанда деректерді ұйымдастырудың көпөлшемділігі жоғары *көрнекілікке* ие және *ақпараттандырылған*. 2.3-суретте машина сату көлемі туралы бірдей деректердің реляциялық (а) және көпөлшемді (б) көрінісі келтірілген.

а)

Үлгі	Айы	Көлемі
“Жигули”	маусым	12
“Жигули”	шілде	24
“Жигули”	тамыз	5
“Москвич”	маусым	2
“Москвич”	шілде	18
“Волга”	шілде	19

б)

Үлгі	Маусым	Шілде	Тамыз
“Жигули”	12	24	5
“Москвич”	2	18	No
“Волга”	No	19	No

2.3-сурет. Деректердің реляциялық және көпөлшемді бейнеленуі

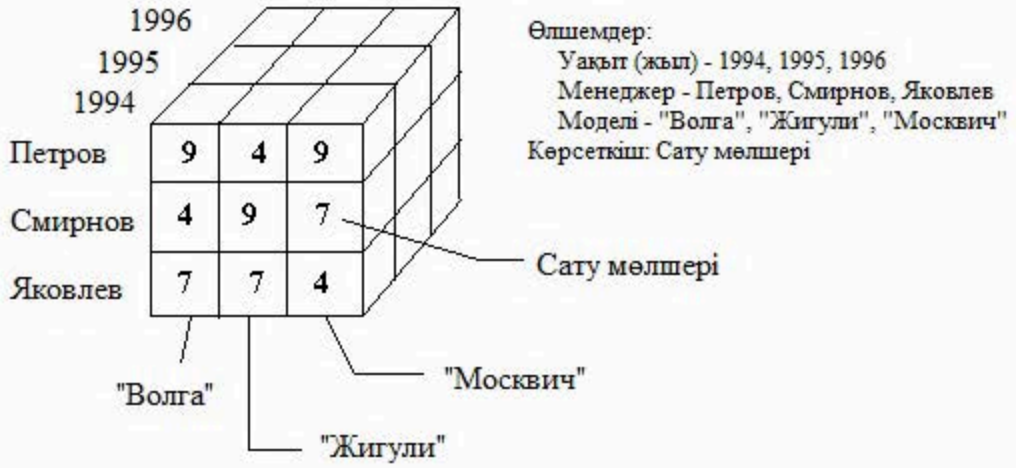
Егер өлшемі екіден артық көпөлшемді үлгілер туралы сөз болса, онда ақпаратты көпөлшемді объектілер (үш-, төрт- және неғұрлым жоғары өлшемді гиперкубтар) түрінде визуальды қарастыру міндетті емес. Пайдаланушыға осы жағдайда екіөлшемді кестелермен немесе графиктермен жұмыс жасаған анағұрлым тиімді. Деректер сонымен қатар көпөлшемді деректер қоймасының әртүрлі деңгейде орындалған “қималарын” (нақтырақ айтқанда, “қиық”) көрсетеді.

Көпөлшемді деректер үлгісінің негізгі ұғымдарын қарастырайық, оларға өлшем және ұяшық жатады.

Өлшем (Dimension) – бұл гиперкуб қабырғаларын құрайтын біртипті деректердің жиыны. Неғұрлым жиі қолданылатын уақыт өлшемдерінің мысалдары: Күндер, Айлар, Тоқсандар және Жылдар. Географиялық өлшемдер ретінде Қалалар, Аудандар, Аймақтар және Елдер кеңінен қолданылады. Көпөлшемді деректер үлгісінде өлшем гиперкуб ұяшықтарындағы нақты мәндерді сәйкестендіру үшін қолданылатын индекс рөлін атқарады.

Ұяшық (Cell) немесе көрсеткіш – бұл мәні бірмәнді анықталатын өріс. Өрістер типі сандық түрде жиі анықталады. Қайсібір ұяшық мәні қалай құрылғанына тәуелді, әдетте ол айнымалы (мәні өзгереді және деректердің сыртқы көзінен жүктелген немесе бағдарламалау кезінде қалыптасқан) немесе формула (мәні электрондық кестедегі формулаға сәйкес есептелген) болуы мүмкін.

2.3 б-суретте ұяшықтың әрбір мәні уақытша өлшем (сату айы) және көлік үлгілері комбинациясымен, сату көлемімен бірмәнді анықталады. Іс жүзінде көбінесе өлшемнің көптеген мәні талап етіледі. Үшөлшемді деректер үлгісі мысалы 2.4-суретте келтірілген.



2.4-сурет. Үшөлшемді үлгілер мысалы

Қолданыстағы КДҚБЖ деректерді ұйымдастырудың екі негізгі нұсқасы (сызба) қолданылады: гиперкубтық және поликубтық.

Поликубтық сызбада ДҚ-да әртүрлі өлшемді және қырларының өлшемі әртүрлі бірнеше гиперкуб анықталуы мүмкін деп жобалайды. Поликубтық ДҚ сүйемелдейтін жүйелерге Oracle Express Server сервері мысал болады.

Гиперкубтық сызбада барлық көрсеткіштер бірдей өлшемдер жиынтығымен анықталған деп жобалайды. Бұл бірнеше гиперкубтар болған кезде ДҚ өлшемдері бірдей дегенді білдіреді. Қайсібір жағдайда ДҚ-да ақпарат артық болуы мүмкін (егер ұяшықтарды толтыру міндеті талап етілсе).

Көпөлшемді деректер үлгісі жағдайында арнайы операциялар қолданылады, оларға жататындар: "қиық" қалыптастыру, "айналу", агрегация және бөлшектеу.

"Қиық" (Slice) бір немесе бірнеше өлшемдерді белгілеу нәтижесінде алынған гиперкуб ішкі жиыны болып саналады. "Қиықтарды" қалыптастыру гиперкубтың барлық мәні іс жүзінде ешқашан бір уақытта қолданылмайтындықтан шектеулер қою үшін орындалады. Мысалы, егер 2.4-суретте гиперкубтың Көлік үлгісі өлшемінің мәніне "Жигули" маркасына шектеу қойса, онда осы маркалы көлікті әртүрлі менеджерлердің жылдар бойынша сатылымының екі өлшемді кестесі алынады.

"Айналу" (Rotate) операциясын деректерді екі өлшемді бейнелеуде қолданады. Оның мағынасы деректердің визуальды бейнелеуіндегі өлшемдер ретінің өзгеруін білдіреді. 2.3 6-суретте көрсетілген екіөлшемді кестелер "айналуы" оның түрінің өзгеруіне апарып соқтырады, X осі бойынша көлік маркасы, ал Y осі бойынша – уақыт.

Егер "айналу" операциясын өлшемдердің кезектесу ретіне өзгерістер енгізу процедурасы ретінде қабылдасақ, онда оны көпөлшемді жағдайға жалпылауға болады. Қарапайым жағдайда, мысалы, бұл екі ерікті өлшемді өзара алмастырулар үшін мүмкін.

"Агрегация" (Drill Up) және "бөлшектеу" (Drill Down) операциясы неғұрлым жалпы жағдайға көшуді және гиперкубтың пайдаланушыға ақпараттарды неғұрлым біртіндеп ұсынуын білдіреді.

"Агрегация" операциясының мағынасын түсіндіру үшін бізде қандай да бір гиперкуб бар деп есептейік. Гиперкубтың 2.4-суретте көрсетілген өлшемдерінен басқа мынадай өлшемдері болсын: Бөлімше, Аймақ, Фирма, Ел. Ескере кетейік, бұл жағдайда гиперкубта өлшемдер арасында қатынастар иерархиясы (төменнен жоғары) бар: Менеджер, Бөлімше, Аймақ, Фирма, Ел.

Сипатталған гиперкубта 1995 жылы "Жигули" және "Волга" көліктерін менеджер Петров қаншалықты табысты сатқандығы анықталған. Онда, иерархия бойынша бір

деңгей жоғары көтеріліп, “агрегация” операциясы көмегімен Петров жұмыс жасайтын бөлімше деңгейінде осы үлгілердің сатылымы қандай екенін анықтауға болады.

Көпөлшемді деректер үлгісінің негізгі ***құндылығы*** деректердің үлкен көлемін уақытпен байланысты өңдеу және ыңғайлылығы мен тиімділігін талдау болып табылады. Ұйымдастыру кезінде деректерді өңдеу реляциялық үлгілер негізінде жүзеге асады.

Көпөлшемді деректер үлгісінің ***кемшілігі*** қарапайым тапсырмаларды шешу кезінде жедел ақпаратты өңдеудің қолайсыздығы.

Көпөлшемді деректер үлгісін сүйемелдейтін жүйелер мысалдары мыналар: Essbase (Arbor Software), Media Multi-matrix (Speedware), Oracle Express Server (Oracle) және Cache (InterSystems). Қайсібір бағдарламалық өнімдер, мысалы Media/MR (Speedware), бір уақытта көпөлшемді және реляциялық ДҚ-мен жұмыс жасайды. Cache ДҚБЖ-сы деректерге қатынас құрудың үш әдісі жүзеге асқан көпөлшемді үлгі болып табылады: тура (көпөлшемді массив тораптары деңгейінде), объектілі және реляциялық.

2.4. Объектілі-бағытталған үлгі

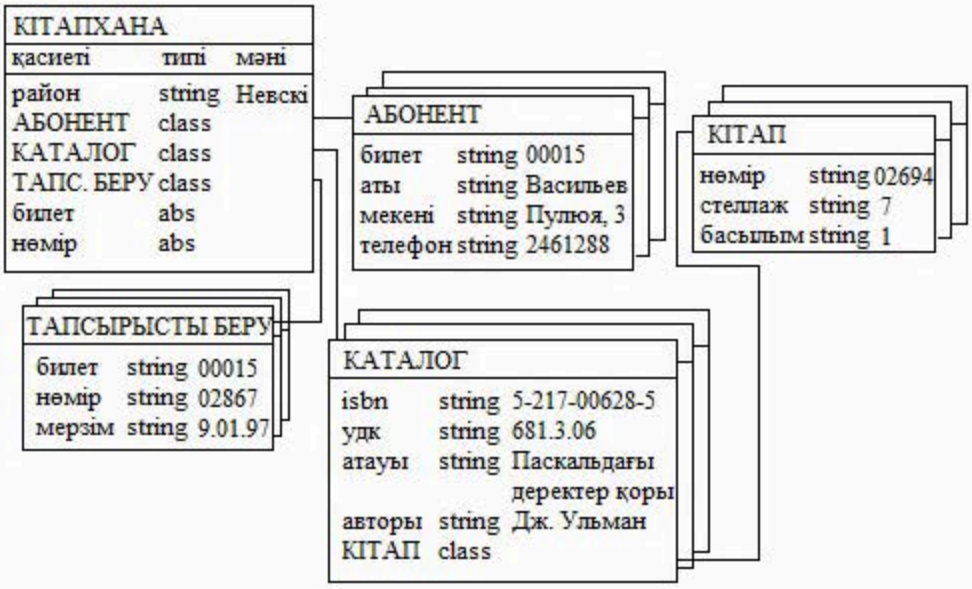
Деректердің бейнеленуінің объектілі-бағытталған үлгісінде қордың жекелеген жазбаларын теңдестіру мүмкіндігі бар. Деректер қоры жазбалары және оларды өңдеу функциялары арасында объектілі-бағытталған бағдарламалау тілінің сәйкес құралдары механиздерінің көмегімен өзара байланыс орнайды.

Стандартталған объектілі-бағытталған үлгі ODMG-93 (Object Database Management Group – объектілі-бағытталған қордың деректерін басқару тобы) стандарты ұсыныстарында сипатталған. ODMG-93 ұсынымдарын толық көлемде жүзеге асыру әзірше мүмкін емес. Кілттік идеяны иллюстрациялау үшін объектілі-бағытталған ДҚ-ның бірнеше қарапайым үлгісін қарастырайық.

Объектілі-бағытталған ДҚ құрылымы графикалық тұрғыда бұтақтәріздес, оның түйіндері объектілер болады. Объектілер қасиеттері қайсыбір стандартты типпен (мысалы, жолдық – string) немесе пайдаланушымен құралған типпен (class деп анықталады) суреттеледі.

Қасиеттер мәні string типті болса, онда ол символдар жолы болып табылады. Қасиеттер мәні class типі болса, онда ол сәйкес класс үлгісі болып табылатын объект. Класстың әрбір объект-үлгісі қасиет ретінде анықталған объектінің ұрпағы болып саналады. Класстың объект-үлгісі өз класына жатады және бір ғана шыққан ортасы бар. ДҚ-дағы туыстық қатынастар объектілердің байланысқан иерархиясын құрайды.

Мысалы кітапхана жұмысының объектілі-бағытталған ДҚ логикалық құрылымы 2.5-суретте келтірілген.



2.5-сурет. Кітапхана жұмысының ДҚ логикалық құрылымы

Мұнда КІТАПХАНА типті объект класстардың АБОНЕНТ, КАТАЛОГ және ТАПСЫРЫСТЫ БЕРУ объект-үлгілері үшін түпкі объект болып табылады. Әртүрлі КІТАП типті объектілердің бір немесе бірнеше түпкі объектісі болуы мүмкін. Бір немесе бірнеше түпкі объектісі бар КІТАП типті объектілер бір-бірінен ең болмағанда инвентарлық нөмірімен (әрбір кітап үшін бірегей) ажыратылуы керек, бірақ қасиеттерінің мәні бірдей *isbn, удк, атауы* және *автор* болуы мүмкін.

Объектілі-бағытталған ДҚ-ның логикалық құрылымы сырттай қарағанда иерархиялық ДҚ құрылымына ұқсас. Олардың арасындағы негізгі айырмашылық деректермен іс-әрекеттер жасау әдістерінде.

Қарастырылған үлгілерде деректермен іс-әрекеттер орындау үшін логикалық операциялар, күшейтілген объектілі-бағытталған инкапсуляция, мұрагерлік және полиморфизм механизмдері қолданылады. SQL командаларына ұқсас операциялар (мысалы, ДҚ құру үшін) шектеулі қолданылуы мүмкін.

ДҚ құру және түрлендіру автоматты түрде қалыптастырылады және деректердің құрамындағы ақпаратты жылдам табу үшін индекстерді түзетеді.

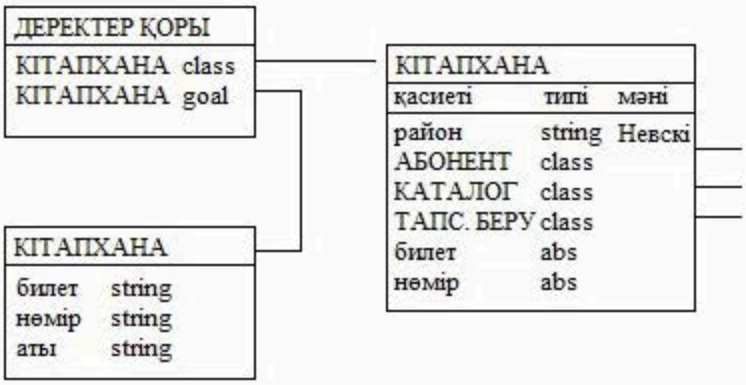
ДҚ-ның объектілі-бағытталған үлгілерінде қолданылатын инкапсуляция, мұрагерлік және полиморфизм түсініктерін қысқаша қарастырайық.

Инкапсуляция анықталған объекттен тыс қасиеттер атауының бейнеленуі облысын шектейді. Масылы, КАТАЛОГ типті объектіге *телефон* деген атаумен кітап авторының телефонын көрсететін қасиет енгізсек, онда біз АБОНЕНТ және КАТАЛОГ объектілерінің қасиеттерімен бірдей атауға ие қасиеттер аламыз. Бұл объектінің мағынасы инкапсуляция жасалған объектімен анықталады.

Мұрагерлік, керісінше, объектінің барлық бөлігіне қасиеттер көрінісінің облысын таратады. Мысалы, КАТАЛОГ типінің мұрагері болып саналатын КІТАП типті барлық объектілерге түпкі объектінің барлық қасиеттерін жазуға болады: *isbn, удк, атауы* және *автор*. Егер мұрагерлік механизмі әрекеттерін басқа да объектілермен толықтыру керек болса, онда оларды abs типті абстрактылы қасиетпен жалпы жағдайда анықтайды. Ендеше, КІТАПХАНА объектісінде *билет* және *номер* абстрактылы қасиеттерін анықтау осы қасиеттердің барлық ішкі АБОНЕНТ, КІТАП және ТАПСЫРЫСТЫ БЕРУ объектілеріне берілуіне әкеледі. Суретте көрсетілген АБОНЕНТ және ТАПСЫРЫСТЫ БЕРУ кластарының билет қасиеті мәні бірдей болуы кездейсоқ емес – 00015.

Объектілі-бағытталған бағдарламалау тілінде ***полиморфизм*** бір ғана бағдарламалық кодтың әртипті деректермен жұмыс жасау мүмкіншілігі барын білдіреді. Басқаша айтқанда, ол әртүрлі типті объектілерде бірдей атауға ие әдістер (процедуралар немесе функциялар) болуы мүмкіндігін білдіреді. Біздің қарастырған мысалдағы объектілі-бағытталған ДҚ полиморфизмы КАТАЛОГ класының әртүрлі бөлімінен шыққан КІТАП кластар объектілері әртүрлі қасиеттер жиынтығына ие екенін білдіреді. Сондықтан, КІТАП кластар объектілері мен жұмыс бағдарламалары полиморфты кодты.

Объектілі-бағытталған ДҚ-да іздеу ДҚ-да сақталатын объектілер арасындағы пайдаланушымен және объектімен енгізілген ұқсастықты анықтаудан тұрады. Пайдаланушымен анықталған объект объектілі-мақсатты (объект қасиеті *goal* типті) деп аталады, жалпы жағдайда ДҚ-да сақталатын барлық мүмкін болып табылатын объектілер жиыны. Сонымен қатар қордың өзінде объект-мақсат пен сұраныстың орындалу нәтижесі сақталуы мүмкін. Мысалы, оқырман билеті нөмірі және абонент атауы туралы сұраныс кітапханадан ең болмағанда бір кітапты көрсетеді. 2.6-суретті қараңыз.



2.6-сурет. Объектілі-мақсатты ДҚ фрагменті

Объектілі-бағытталған деректер үлгісінің негізгі ***құндылығы*** ақпараттар объектілерінің күрделі өзара байланысын бейнелеу мүмкіндігі болып табылады. Объектілі-бағытталған деректер үлгісі жекелеген жазба деректер қорын теңдестіру мүмкіндігін береді және оларды өңдеу функцияларын анықтайды.

Объектілі-бағытталған деректер үлгісінің ***кемшіліктері*** объектілі-бағытталған үлгілерді түсінуде қиындықтар туындайды, деректерді өңдеу ыңғайсыз және сұраныстардың орындалу жылдамдығы төмен.

90-шы жылдары объектілі-бағытталған деректер қорын басқару жүйелерінің экспериментальды түпнұсқалары болды. Қазіргі заманда олар кеңінен тарады, жеке жағдайда оларға келесі ДҚБЖ жатады: POET (POET Software), Jasmine (Computer Associates), Versant (Versant Technologies), O2 (Ardent Software), ODB-Jupiter (ғылыми-өндірістік орталық "Интеллект Плюс"), сонымен қатар Iris, Orion және Postgres.

2.5. Деректер типі

Алғашқыда ДҚБЖ қаржылық-экономикалық есептерді шешу үшін қолданылды. Сонымен қатар, көрініс үлгілерінен тәуелсіз, деректер қорында келесі негізгі деректер типі қолданылды:

- сандық. Деректердің мәндер мысалдары: 0.43, 328, 2E+5;
- символдық (алфавитті-цифрлық). Деректердің мәндер мысалдары: "жұма", "жол", "бағдарлама құрушы";

• мерзімдер, арнайы "Мерзім" типімен немесе кәдімгі символдық деректердің көмегімен беріледі. Деректердің мәндер мысалдары: 1.12.97, 2/23/1999.

Әртүрлі ДҚБЖ-де бұл типтердің бір-бірінен атауы, мәндердің өзгеру аумағы және көрініс түрі бойынша аздаған айырмашылығы болуы мүмкін. Нәтижесінде деректерді өңдеудің, мысалы, геоақпараттық және т.б. мамандандырылған жүйелері пайда бола бастады. Осыған сәйкес бағдарлама құрушы дәстүрлі ДҚБЖ-не жаңадан деректер типін енгізе бастады. Деректердің жаңа типтеріне келесілерді жатқызуға болады:

• уақытша және мерзімді-уақытша, уақыт және/немесе мерзім туралы ақпараттарды сақтауға арналған. Деректердің мәндер мысалдары: 31.01.85 (мерзімі), 9:10:03 (уақыт), 6.03.1960 12:00 (мерзім және уақыт);

• ұзындығы айнымалы символдық, үлкен көлемді мәтіндік ақпараттарға сақтауға арналған, мысалы, құжаттар;

• екілік, графикалық объектілерді, аудио- және видеоақпараттарды, кеңістіктік, хронологикалық және басқа арнайы ақпараттарды сақтауға арналған. Мысалы, MS Access бағдарламасында деректердің "OLE объектісі" типі ДҚ-да BMP (Bitmap) форматында графикалық деректерді сақтауға мүмкіндік береді және оларды ДҚ-мен жұмыс жасау кезінде автоматты түрде бейнелейді;

• гиперсілтемелер (hyperlinks), деректер қорынан тыс орналасқан әртүрлі ресурстарға (түйіндер, файлдар, құжаттар және т.б.) сілтемелерді сақтауға арналған. Мысалы, Internet желісінде, бірлескен intranet желісінде немесе компьютердің қатқыл дискісінде. Деректердің мәндер мысалдары: http:\\www.chat.ru, ftp:\\chance4u.teens.com.

Деректердің әртүрлі үлгілері бар заманауи ДҚБЖ-ларда деректердің барлық аталған типтері қолданылады.

Бақылау сұрақтары және тапсырмалар

1. Деректерді ұсынудың классикалық және заманауи үлгілерін атап шығыңыз.
2. Иерархиялық деректер үлгісінің құндылығы мен кемшілігін түсіндіріңіз.
3. Иерархиялық типті ДҚ-да деректердің физикалық орналасуы қалай ұйымдастырылады?
4. Желілік деректер үлгісін сипаттаңыз.
5. Реляциялық деректер үлгісін сипаттаңыз.
6. Деректердің постреляциялық және реляциялық үлгілерінің арасында қандай айырмашылық бар?
7. Постреляциялық үлгілердің құндылығы мен кемшілігін түсіндіріңіз.
8. Көпөлшемді деректер үлгісін сипаттаңыз.
9. Көпөлшемді үлгілер жағдайында деректермен орындалатын операциялар мағынасын атаңыз және түсіндіріңіз.
10. Объектілі-бағытталған деректер қорына қолданылатын инкапсуляция, полиморфизм және мұрагерлік принциптеріне анықтама беріңіз және мысалдар келтіріңіз.
11. Деректерді ұсынудың объектілі-бағытталған үлгілерінің құндылығы мен кемшілігін түсіндіріңіз.
12. Заманауи ДҚБЖ-ларда қолданылатын деректердің типін сипаттаңыз.
13. MS Access ДҚБЖ ДҚ-да суреттерді сақтауға және қарауға болады ма?

3. ДЕРЕКТЕРДІҢ РЕЛЯЦИЯЛЫҚ ҮЛГІСІ

Бөлімде деректерді ұсынудың неғұрлым кең таралған реляциялық үлгісі қарастырылады. Реляциялық үлгілердің анықтамасы және оның элементтерінің сипаттамасы беріледі. Индекстеу, кестелерді байланыстыру және байланыс тұтастығын бақылау суреттеледі. Сұраныстар тілдерін тұрғызудың теориялық негіздері қарастырылады: реляциялық алгебра және реляциялық есептеулер. QBE және SQL тілдерінің сипаттамасы, олардың сұраныстар көмегімен тұрғызу операторлары форматы және мысалдары беріледі.

3.1. Реляциялық үлгінің анықтамасы

Деректердің реляциялық үлгісі (ДРҮ) уақыт бойынша өзгеретін қатынастар жиынтығының қайсібір зерттеу облысы болып саналады. Ақпараттық жүйелер құрған кезде қатынастар жиынтығы деректер объектілерінің зерттеу облысы және олардың арасынағы байланыстар үлгілерін сақтауға мүмкіндік береді. ДРҮ элементтері және формалар 3.1-кестеде келтірілген.

3.1-кесте. Реляциялық үлгілер элементтері

Реляциялық үлгілер элементі	Формалар көрінісі
Қатынас	Кесте
Қатынастар сызбасы	Кесте бағаналары атауларының жолы (кесте атаулары)
Кортеж	Кестелер жолы
Мағынасы	Объект қасиетін сипаттау
Атрибуты	Кесте бағаналарының атаулары
Домен	Атрибуттың мүмкін мәндерінің жиыны
Атрибут мәні	жазбадағы өрістер мәні
Алғашқы кілт	Бір немесе бірнеше атрибуттар
Деректердің типі	Кестелер элементтері мәндерінің типі

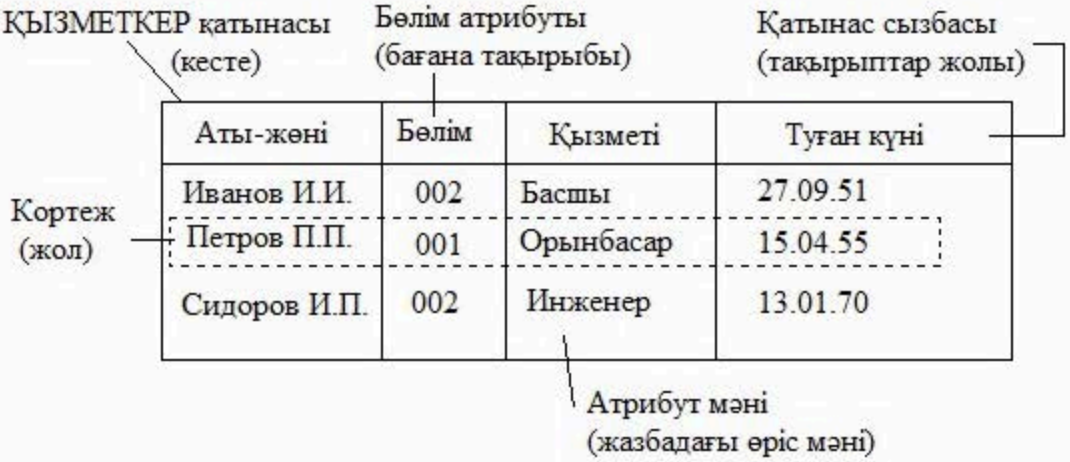
Қатынас маңызды ұғым болып табылады және қайсібір деректерді қамтитын екіөлшемді кесте болып саналады.

Мағынасы дегеніміз деректер қорында сақталатын кез-келген объект табиғаты туралы деректер. Болмыстар туралы деректер қатынастарда сақталады.

Атрибуттар мағынасын сипаттайтын қасиеттерді білдіреді. Кестелер құрылымында әрбір атрибутқа атау беріледі және оған қайсыбір кесте бағанының атауы сәйкес келеді.

Математикалық тұрғыда қатынасты келесі түрде сипаттауға болады. D1, D2, D3, ... , Dn n жиындар берілсін. Онда R қатынасы реттелген <d1, d2, d3 , ... , dn> кортеждер жиыны, онда dk ∈ Dk, dk – ***атрибут***, ал *Dk – R* қатынастар ***домені***.

3.1-суретте ҚЫЗМЕТКЕР қатынастары келтірілген.



3.1-сурет. ҚЫЗМЕТКЕР қатынастарының бейнеленуі

Жалпы жағдайда қатынастардағы кортеждер реті, қалай және қай жиында екендігі анықталмаған. Бірақ реляциялық ДҚБЖ-де ыңғайлылық үшін барлық кортеждерді ретке келтіреді. Көбінесе осы үшін қайсыбір атрибут таңдайды, ол бойынша жүйелер кортеждерді өсуі немесе кемуі бойынша автоматты түрде сұрыптайды. Егер пайдаланушы реттеу атрибутын тағайындамаса, онда жүйелер оларды автоматты түрде енгізу реті бойынша нөмірлейді.

Егер қатынастарда атрибуттардың орнын ауыстырса, онда жаңа қатынас алынады. Бірақ, реляциялық ДҚ-да атрибуттардың орнын ауыстыру жаңа қатынас тудырмайды.

Домен қатынастардың қандай да бір атрибутының барлық мүмкін мәндерінің жиыны болып саналады. ҚЫЗМЕТКЕР қатынасы 4 доменнен тұрады. *Домен 1* - барлық қызметкерлердің фамилиясы, *домен 2* – фирманың барлық бөлімдерінің нөмірі, *домен 3* – барлық қызметтердің атауы, *домен 4* – барлық қызметкерлердің туған күні. Әрбір домен бір типті деректердің мәнін құрайды, мысалы, сандық немесе символдық.

ҚЫЗМЕТКЕР қатынасы 3 кортежден тұрады. Қарастырылатын қатынастар кортежі 4 элементтен тұрады. Олардың әрқайсысы сәйкес доменнен алынады. Әрбір кортежге кестелер жолы сәйкес келеді (3.1-сурет).

Қатынастар сызбасы қатынастар тақырыбы, ал атрибуты атаулары тізімі болып саналады. Келтірілген мысалда қатынастар сызбасы ҚЫЗМЕТКЕР (аты-жөні, Бөлімі, Қызметі, Туған күні) түріне ие. Қатынастар кортежі жиынын әдетте ***қатынастар құрамы (денесі)*** деп атайды. Кортеждердің әрқайсысы бірмәнді өзгеретін қатынастар атрибуты ***алғашқы кілт*** (***қатынастар кілті, кілттік атрибуттар***) деп аталады. Мысалы, ҚЫЗМЕТКЕР (аты-жөні, Бөлімі, Қызметі, Туған күні) қатынастарында "АТЫ-ЖӨНІ" атрибуты кілттік болып табылады. *Кілт* ***құрама*** (***күрделі***) болуы мүмкін, яғни бірнеше атрибуттардан тұрады.

Әрбір қатынас міндетті түрде кілт рөлін атқаруы мүмкін атрибуттар комбинациясына ие. Қатынас - бірдей элементтер кортежін қамтымайтын жиын болғандықтан міндетті түрде болады. Яғни қатынастарда қайталанатын кортеждер жоқ, ал бұл атрибуттардың барлық жиынтығы қатынастардың кортеждерді бірмәнді сәйкестендіру қасиетіне ие екендігін білдіреді. Көптеген ДҚБЖ кілттерді анықтамай, қатынастар құруға рұқсат береді.

Қатынастың бірнеше атрибуттар комбинациясына ие жағдайлары кездесуі мүмкін, олардың әрқайсысы барлық кортеждерде бірмәнді анықтайды. Барлық атрибуттар комбинациялары қатынастардың ***мүмкін кілттері*** болады. Мүмкін кілттердің кез-келгенін *алғашқы* кілт ретінде таңдауға болады.

Егер таңдалған алғашқы кілт ең аз қажетті атрибуттар жиынынан тұрса, онда ол ***артық атрибутсыз*** деп аталады.

Кілттерді әдетте келесі мақсаттарға жету үшін қолданады:

1) Кілттік атрибуттарда қайталанатын мәндерді шығарып тастау (қалған атрибуттар есептелмейді);

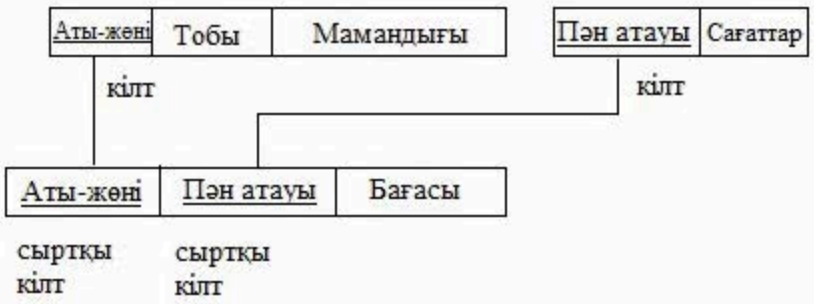
2) Кортеждерді реттеу. Барлық кілттік атрибуттар мәндерін өсуі немесе кемуі бойынша реттеуге сонымен қатар, аралас (біреуі – өсуі, басқалары – кемуі бойынша) реттеуге болады;

3) Қатынастар кортеждерімен жұмыс жасауды жылдамдату (3.2 бөлім);

4) Кестелерді байланыстыруды ұйымдастыру (3.3 бөлім).

R1 қатынастарда ал *кілттік емес* атрибуты болсын. Оның мәні басқа R2 қатынастардың *кілттік* атрибутының мәніне тең емес. Онда ал атрибуты R1 қатынастардың ***сыртқы кілті*** деп аталады.

Сыртқы кілттер көмегімен қатынастар арасында байланыстар орнатылады. Мысалы, екі қатынастар СТУДЕНТ (аты-жөні, Тобы, Мамандығы) және ПӘН (Пән атауы, Сағаттар) берілсін, олар СТУДЕНТ_ПӘН(аты-жөні, Пән атауы, Бағасы) қатынасымен байланысты (3.2-сурет). Байланыстыратын қатынастарда Аты-жөні және Пән атауы атрибуттары құрама кілт құрастырады. Бұл атрибуттар басқа қатынастардың алғашқы кілті болатын сыртқы кілттерді көрсетеді.



3.2-сурет. Қатынастар байланысы

Реляциялық үлгі деректер тұтастығын қамтамасыз ету үшін сыртқы кілттерге *сілтемелік тұтастық* деп аталатын шектеу қояды. Бұл сыртқы кілттің әрбір мәніне байланыстыратын қатынастардың жолдары сәйкес келуі керектігін білдіреді.

Себебі кез-келген кестені қатынаспен сәйкестендіре алмаймыз. Орындалуы кестені қатынас деп есептеуге мүмкіндік беретін шарттарды қарастырайық.

1. Кестелердің барлық жолдары өзгеше болуы керек, яғни алғашқы кілттері бірдей жолдардың болуы мүмкін емес.

2. Кесте бағаналары атаулары әртүрлі, ал мәндері қарапайым болуы керек, яғни бір жолдың бір бағанасында мәндер тобының болуы мүмкін емес.

3. Бір кестенің барлық жолдары, сәйкес атаулары және бағана типтері бір құрылымға ие болуы керек.

4. Кестеде жолдардың орналасу реті кез-келген болуы мүмкін.

Әдетте кесте қатынастарымен жеке файлда орналасады. Қайсібір ДҚБЖ-да бір жекелеген кесте (қатынас) деректердің қоры болып саналады. Басқа ДҚБЖ-да деректер қоры бірнеше кестелерден тұруы мүмкін.

Жалпы жағдайда ДҚ мағыналық құрамы, сонымен қатар қайсыбір қолданбалы есепті шешу үшін тұтастықты бақылау және ақпаратты өңдеу процедуралары бойынша біріктірілген бір немесе бірнеше кестені іске қосады деп есептеуге болады. Мысалы, Microsoft Access ДҚБЖ қолданған кезде ДҚ файлында кестелермен бірге қордың басқа обектілері: сұраныстар, есеп беру, формалар, макростар және модульдер сақталады.

Деректер кестесі әдетте магниттік дискте жеке файлдарда сақталады, сондықтан оған атаулары бойынша шектеулер қойылуы мүмкін. Өріс аттары кесте ішінде сақталады. Оларды қалыптастыру ережесі ДҚБЖ-да, қағидаға сәйкес, өрістің ұзындығы бойынша анықталады және онда қолданылатын алфавит маңызды шектеулер қоймайды.

Егер кестеде берілген қатынас кілттік болса, онда кесте де кілттік болып саналады және оны ***кілттік*** немесе ***кілттік өрістері бар кесте*** деп атайды.

Көптеген ДҚБЖ-да кестелер файлы басқарушы бөлімді (өріс типтерін сипаттау, өріс аттары және басқа ақпарат) және жазбалардың орналасу облысын іске қосады.

Қатынастарға бір қатынастардан басқа қатынастар алуға мүмкіндік беретін жүйелер операцияларын қолдануға болады. Мысалы, реляциялық ДҚ-на сұраныстың нәтижесі қолданыста бар қатынастардың негізінде есептелген жаңа қатынас болуы мүмкін. Сондықтан өңделетін деректерді сақталатын және есептелетін бөлімдерге бөлуге болады. Реляциялық ДҚ-да деректерді өңдеудің негізгі бірлігі қатынас болып табылады.

3.2. Индекстеу

Жоғарыда айтылғандай, кестелер үшін кілт анықтамасы жазбаларды автоматты түрде сұрыптау, жазбалардың кілттік өрістерінде мәндердің қайталанбауын қадағалау және кестеден іздеу операциялары орындалуының жылдамдығын арттыруды білдіреді. ДҚБЖ-да осы функциялардың жүзеге асуы үшін *индекстеуді* қолданады.

“Индекс” термині “кілт” түсінігімен, олардың арасында қайсыбір айырмашылықтар болса да, тығыз байланысты.

Индекс дегеніміз кестедегі жазбаларды іздеу операциясын, сонымен қатар іздеуді қолданатын басқа операцияларды: алу, түрлендіру, сұрыптау және т.б. жылдамдату құралы. Индекс қолданылатын кесте *индекстелген* деп аталады.

Индекс рөлін кестелер *тақырыбы* атқарады, оларды көру үшін кесте жазбаларына қатынас жасау керек. Қайсібір жүйелерде, мысалы Paradox, индекстер файлдар кестесінен бөлек индекстік файлдарда сақталады.

Ақпараттарға физикалық қатынас құруды ұйымдастыру мәселелерін шешу нұсқалары көбінесе келесі факторлардан тәуелді:

- индекстік файл жазбалары кілттік өрісінің құрамының түрі;
- негізгі кестелер жазбасында қолданылатын сілтемелер (көрсеткіштер) типі;
- қажетті жазбаларды іздеу әдісі.

Индекстік файл кілттік өрісінде индекстелген кестелер немесе кілттер топтамасы (хеш-код деп аталатын) кілттік өрісі мәнін сақтауға болады. Мәні орнына хеш-кодты сақтау артықшылығы: кілттер топтамасының ұзындығы кілттік өрістердің бастапқы мәнінің ұзындығынан тәуелсіз. Ол әрқашан қайсыбір тұрақты және жеткілікті аз шама (мысалы, 4 байт), бұл іздеу операцияларына кететін уақытты недәуір азайтады. Оның кемшілігі бүктеу операциясының орындалу қажеттілігі болып табылады (белгілі бір уақытты талап етеді), сонымен қатар пайда болған коллизиялармен күресу керек (әртүрлі мәндерді бүктеу бірдей хеш-код беруі мүмкін).

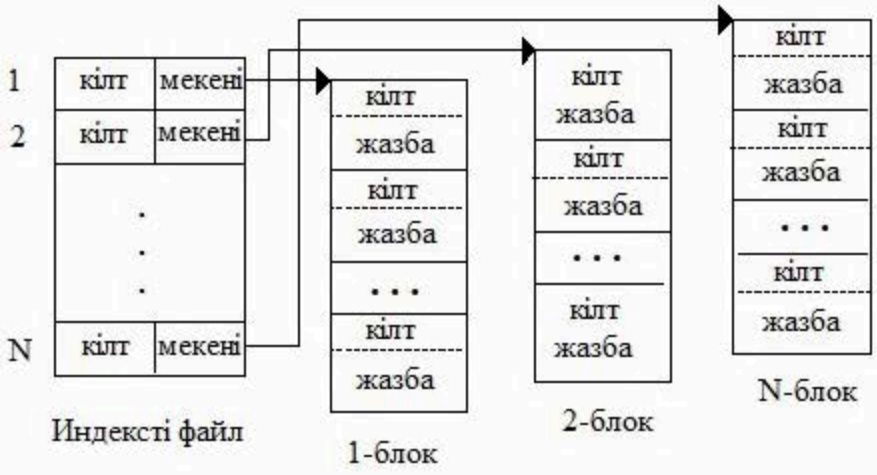
Жазбаларға *сілтемелер* ұйымдастыру кестелері үшін адресітеудің үш түрін қолданады: абсолюттік (шынайы), салыстырмалы және символикалық (идентификатор).

Іс жүзінде көбінесе *екі іздеу әдісі* қолданылады: тізбектелген және бинарлық (кесіндіні қақ бөлу әдісіне негізделген).

Кестеде индекстеуді ұйымдастыруды екі түрлі сызбамен сипаттайық: бір деңгейлі және екі деңгейлі. Сонымен қатар, әдетте заманауи есептеу жүйелерінде орындалатын жағдайлар қаратырайық. ОЖ магниттік дискіде деректерді ұйымдастыруды тікелей

сүйемелдейтін болсын, негізгі кестелер және индекстік файлдар жекелеген файлдарда сақталады. Файлдарда сақталатын ақпарат белгіленген өлшемді блоктар жиынтығы, мысалы, кластерлердің бүтін саны.

Бір деңгейлі сызба кезінде екі өрістен тұратын қысқа жазба индекстелген файлда сақталады: адресітелетін блоктың бастапқы кілтті қамтитын өрісі (хеш-код кілті) және адресі өрісі осы блоктың басы (3.3-сурет). Әрбір блокта жазба кілттің мәні бойынша өсу ретімен орналасады. Әрбір блоктың бастапқы кілті оның соңғы жазбасының кілті болып табылады.



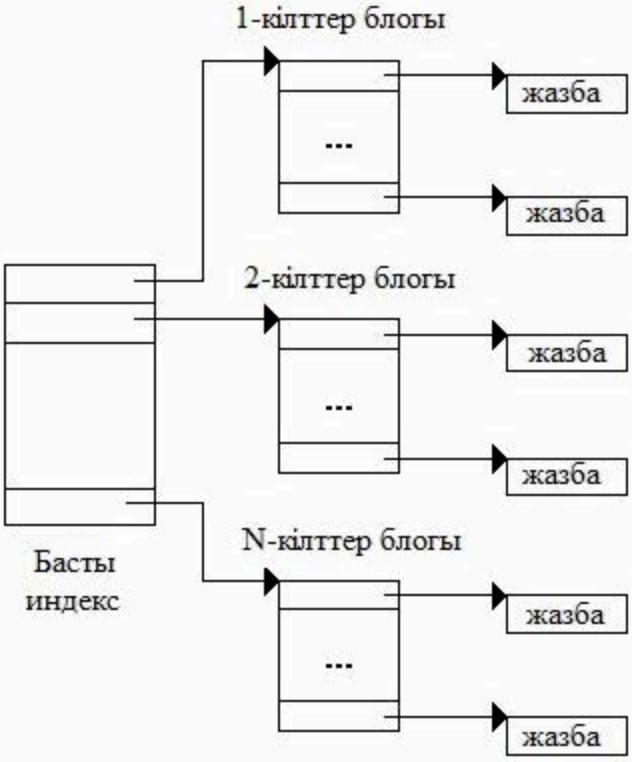
3.3-сурет. Индекстеудің бірдеңгейлі сызбасы

Егер индекстелген файлда индекстелген кестелердің кілттік өрісінің хеш-коды сақталатын болса, онда кестеде қажетті жазбаны іздеу алгоритмы (көрсетілген кілтпен) келесі үш кезеңді іске қосады:

1. Ізделінді жазба кілттік өрістер мәнін білу.
2. Индекстелген файлда блок туралы жазбаны іздеу, алғашқы өріс мәні алынған мәннен үлкен (бұл осы блокта ізделінді мәннің табылуына кепілдік береді).
3. Блок жазбаларын тізбектей көру ізделінді жазбамен файл блогы жазбасы сәйкес келгенге дейін орындалады. Коллизия жағдайында кілттің мәні ізделінді жазба кілті мәнімен сәйкес келегенге дейін ізделінеді.

Бір деңгейлі сызбаның негізгі *кемшілігі* жазбалар кілттері жазбалармен бірге сақталатығында болып табылады. Бұл жазбаларды іздеу уақыттың артуына (жазбалардағы деректердің мәнін жіберуге тура келеді) әкеледі.

Екі деңгейлі сызба кейбір жағдайларда неғұрлым рационалды болып келеді, онда кілттер жазбалар құрамынан ажыратылған (3.4-сурет). Бұл сызбада негізгі кестелер индексі файлдар жиынтығы бойынша орналасқан: басты индексті бір файлға және кілттер блоктарымен файлдар жиынында.

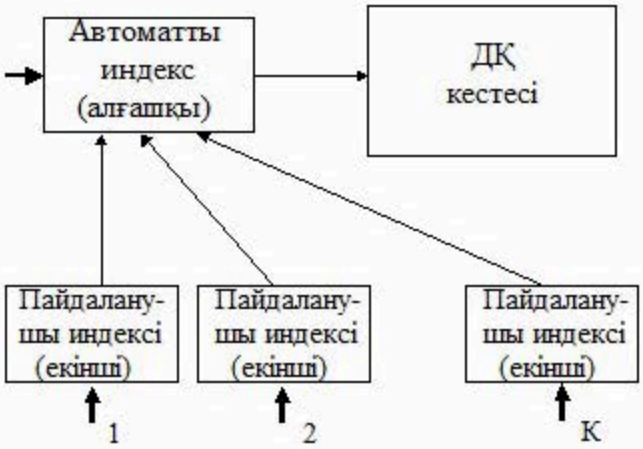


3.4-сурет. Екі деңгейлі сызбаны индекстеу

Іс жүзінде индекс құру үшін ДҚ қайсібір кестелерінде пайдаланушы индекстеуді талап ететін кестелер өрісін көрсетеді. Кестелердің кілттік өрістері көптеген ДҚБЖ-да, қағидаға сәйкес, автоматты түрде индекстеледі. Кестелердің кілттік өрістері бойынша құрылған индекстік файлдар әдетте *алғашқы индексті файл* деп аталады.

Пайдаланушы үшін құрылған кілттік емес өріс кей жағдайда ***екінші (пайдаланушы) индексі*** деп аталады. Мұндай индекстерді кірістіру кестелердегі жазбалардың физикалық орналасуын өзгертпейді, бірақ жазбаларды көру тізбегіне әсер етеді. Индекстік файлдар кестелердің екінші индексін сүйемелдеу үшін құрылған, әдетте *екінші индексті файлдар* деп аталады.

Екінші индекстің деректер қоры элементтерімен байланысы әртүрлі әдістермен орнатылған болуы мүмкін. Олардың біреуі – екінші индексті алғашқы кілтті алу үшін қолдану, ол бойынша қажетті жазбалар ізделінеді (3.5-сурет).



3.5-сурет. Екінші индекстерді пайдалану әдісі

Қайсыбір ДҚБЖ-да, мысалы Access, индекстерді алғашқы және екіншіге бөлу жоқ. Бұл жағдайда автоматты түрде құрылған индекстер және кілттік емес өрістердің кез-келгені бойынша пайдаланушымен анықталатын индекстер қолданылады.

Негізгі себеп индекстелген кестелердегі әртүрлі операциялардың орындалу жылдамдығын арттыру: негізгі бөліммен жұмыс кестелердің өзімен емес, кішігірім индекстік файлдармен жүзеге асады. Индекстелген кестелермен жұмыс өнімділігін арттырудың жоғары тиімділігі көлемі бойынша үлкен кестелер үшін қолжетімді болады. Индекстеу дискіден кішігірім қосымша орынды және индекстерді өзгерту жұмыс процесі үшін кішігірім процессор шығынын талап етеді. Жалпы жағдайда, индекстер ДҚ-на сұраныс жасау алдында, сұраныстарды орындағаннан кейін пайдаланушының арнайы командаларымен немесе бағдарламаның қосымшаны шақыруымен өзгере алады.

3.3. Кестелерді байланыстыру

Нақты ДҚ-ын жобалау кезінде ақпаратты әдетте бірнеше кестелерге орналастырады. Кестелер сонымен қатар ақпараттардың семантикасымен байланысты. Реляциялық ДҚБЖ-де кестедегі байланысты көрсету үшін оларды *байланыстыру* операциясын жүзеге асырады.

Нәтижесінде кестелерді байланыстыруды қамтамасыз ететін ұтымдылықты көрсетейік. Көптеген ДҚБЖ кестелерді байланыстыру кезінде деректер қорына енгізу тұтастығын бақылауды тағайындалған байланыстарға сәйкес автоматты түрде орындайды. Нәтижесінде бұл ДҚ-да сақталатын ақпараттардың *нақтылығын* арттырады.

Сонымен қатар, кестелер арасында байланыстар орнату деректерге қатынас құруды *жеңілдетеді*. Кестелерді байланыстыруды жүзеге асыру кезінде іздеу, көру, редактірлеу, таңдау және есеп беру операциялары әдетте байланысқан жазбалардың кез-келген өрісіне қатынас құру мүмкіндігін қамтамасыз етеді. Бұл деректер кестесіне қатынас құру санын және олардың әрқайсысының бұрмалану санын азайтады.

Кестелер байланысының негізгі түрі

Кестелер арасында бинарлық (екі кесте арасында), үштік (үш кесте арасында) және жалпы жағдайда, n-дік байланыстар орнатылуы мүмкін. Неғұрлым жиі кездесетін *бинарлық* байланыстарды қарастырайық.

Екі кестені байланыстыру кезінде негізгі және қосымша (бағынышты) кестелерді бөліп көрсетеді. Логикалық кестелерді байланыстыру ***байланыстар кілті*** көмегімен жүзеге асады.

Кілттік байланыстар кәдімгі кілттік кестелердің бір немесе бірнеше өрісінен тұрады, олар осы жағдайда ***өрістермен байланыстар*** (ӨБ) деп аталады.

Байланыстыру мағынасы негізгі және қосымша кестенің өрістері байланыстарын сәйкестендіруден тұрады. Өрістер байланысының негізгі кестелері кәдімгі және кілттік болуы мүмкін. Бағынышты кестелерде байланыстар өрісі ретінде көбінесе кілттік өрістер қолданылады .

Негізгі және қосымша кесте өрістері байланыстары қалай анықталғандығынан тәуелді екі кесте арасында жалпы жағдайда келесі негізгі төрт түрлі байланыстар орнатылуы мүмкін (3.2-кесте):

- бірге-бір (1:1);
- бірге-көп (1:M);
- көпке-бір (M:1);
- көпке-көп (M:M немесе M:N).

3.2-кесте. Кестедегі байланыстар түрі сипаттамасы

Түрі бойынша байланыстар өрісі сипаттамасы	1:1	1:М	М:1	М:М
негізгі кестелердегі өрістер байланысы	кілтпен болады	кілтпен болады	кілтсіз болады	кілтсіз болады
қосымша кестелердегі өрістер байланысы	кілтпен болады	кілтсіз болады	кілтпен болады	кілтсіз болады

Екі кесте арасындағы аталған байланыстар түріне сипаттама берейік және оларды пайдалануға мысалдар келтірейік.

1:1 түріндегі байланыс

1:1 түріндегі байланыс негізгі және қосымша кестенің барлық өрістерінің байланысы кілттік болған жағдайда қалыптасады. Себебі өрістердегі кілттің мәні екі кестеде қайталанбайды, осы кестенің жазбалары өзара-бірмәнді болады. Кестелердің өзі, мағынасы бойынша, теңқұқылы.

1- мысал.

Негізгі О1 және қосымша Д1 кестелер бар болсын. Кілттік өрістерді "*" символымен белгілейік, байланыстар өрістерін "+" символымен белгілейік.

О1-кесте

*	+
Өріс11	Өріс12
а	10
б	40
в	3

Д1-кесте

*	+
Өріс21	Өріс22
А	үстел
В	кітап

Келтірілген кестеде жазба (а, 10) кестелер О1 және жазба (а, үстел) кестелер Д1 арасында байланыс орнатылған. Бұның негізі өрістердегі байланыстар мәндерінің сәйкес келуі болып табылады. Ұқсас байланыс (в, 3) жазбалары және (в, кітап) кесте арасында бар. Кестелерде жазбалар кілттік өрістердің мәні бойынша сұрыпталған.

Екі кесте жазбаларын салыстыру мәні бойынша жаңа "виртуалдық жазбалар" (псевдожазбалар) мағлұматын білдіреді. Ендеше, жазбалардың алғашқы жұбын жаңа (а, 10, үстел) псевдожазба түрінде, ал екіншісін (в, 3, кітап) псевдожазбасы түрінде есептеуге болады.

Іс жүзінде 1:1 түріндегі байланыстар салыстырмалы түрде аз қолданылады, өйткені екі кестеде сақталатын ақпаратты бір кестеге біріктіру оңай және ЭЕМ жадысында әлдеқайда аз орын алады. Бір емес, екі және неғұрлым көп кестелермен жұмыс жасау тиімді жағдайлары болуы мүмкін. Мұның себебі өңдеуді жылдамдату, бірнеше пайдаланушылардың ортақ ақпараттармен жұмыс жасау ыңғайлылығын арттыру, ақпараттарды қорғаудың неғұрлым жоғары деңгейін қамтамасыз ету және т.б. қажеттіліктер болуы мүмкін. Соңғы айтылған себептерді суреттейтін мысал келтірейік.

2-мысал.

Қайсібір ғылыми-зерттеу жұмыстарын ұйымдастыруда орындалатын мәліметтер берілсін. Бұл деректер әрбір жұмыс бойынша келесі ақпараттарды қамтиды: тақырыбы (девиз және жұмыстың толық атауы), шифр (код), жұмысты бастау және аяқтау уақыттары, кезеңдер саны, басты атқарушы және басқа қосымша ақпараттар. Барлық жұмыстарға "Қызметтік пайдалану үшін" немесе "құпия" грифі қойылған.

Мұндай жағдайда барлық ақпаратты екі кестеде сақтаған жөн: олардың біреуінде – барлық құпия ақпаратты (мысалы, шифр, жұмыстың толық атауы және басты атқарушы), басқасында – барлық қалған құпия емес ақпаратты. Екі кестені жұмыс шифры бойынша байланыстыруға болады. Кестелердің алғашқысын рұқсат етілмеген қатынас құрулардан қорғаған жөн.

1:М түріндегі байланыс

1:М байланысы кестенің бір жазбасына қосымша кестелердің бірнеше жазбасы сәйкес келген жағдайда қолданылады.

3-мысал.

Екі байланысты О2 және Д2 кестелерін қарастырайық. О2-кестеде ДЭЕМ мультимедиа-құрылғылары түрлері туралы ақпарат, ал Д2-кестеде осы құрылғыны шығаратын фирмалар туралы, сонымен қатар қоймада ең болмағанда бір құрылғының бар екендігі туралы мәліметтер берілсін.

О2-кесте

* +

Коды	Құрылғы түрі
а	CD-ROM
б	CD-Recorder
в	Sound Blaster

Д2-кесте

* + *

Коды	Өндіруші фирма	Бар болуы
а	Acer	бар
а	Mitsumi	жоқ
а	NEC	бар
а	Panasonic	бар
а	Sony	бар
б	Philips	жоқ
б	Sony	жоқ
б	Yamaha	бар
в	Creative Labs	бар

Д2-кестеде екі кілттік өріс бар, өйткені бір фирма әртүрлі құрылғы шығаруы мүмкін. Мысалы Sony фирмасы компакт-дискілерді оқу және қайтадан жазу құрылғысын шығарады.

Екі кестеде жазбаларды "Код" өрісі бойынша салыстыру келесі псевдожазбаны туындайды: (а, CD-ROM, Acer, бар), (а, CD-ROM, Mitsumi, жоқ), (а, CD-ROM, NEC, бар), (а, CD-ROM, Panasonic, бар), (а, CD-ROM, Sony, бар), (б, CD-Recorder, Philips, жоқ), (б, CD-Recorder, Sony, бар) және т.б.

Егер псевдожазбаны жаңа кестеге қиюластырса, онда ДЭЕМ мультимедиа-құрылғыларының барлық түрлері, оларды өндіруші фирмалар, сонымен қатар нақты бір құрылғының қоймада бар болуы туралы толық ақпаратты аламыз.

М:1 түріндегі байланыс

М:1 түріндегі байланыс негізгі кестелердің бір немесе бірнеше жазбаларына қосымша кестенің бір жазбасы сәйкес келген кезде қолданылады.

4-мысал.

Кестедегі О3 және Д3 байланысын қарастырайық. Негізгі О3-кестеде бөлшектердің атаулары (Өріс11), бөлшектерді дайындауға арналған материалдар түрлері (Өріс12) және материалдар маркасы (Өріс13) туралы ақпарат сақталған. Қосымша Д3-кестеде бөлшектердің атаулары (Өріс21), жасап шығарудың жоспарланған уақыты (Өріс22) және тапсырыстардың бағасы (Өріс23) туралы мәліметтер болады.

О3-кесте

\+

Өріс11	Өріс12	Өріс13
бөлшек1	шойын	марка1
бөлшек1	шойын	марка2
бөлшек2	болат	марка1
бөлшек2	болат	марка2
бөлшек2	болат	марка3
бөлшек3	алюминий	-
бөлшек4	шойын	марка2

Д3-кесте

* \+

Өріс21	Өріс22	Өріс23
бөлшек1	4.03.98	90
бөлшек2	3.01.98	35
бөлшек3	17.02.98	90
бөлшек4	6.05.98	240

Осы кестелерді байланыстыру жазбалар арасында сәйкестендіру орнатуды қамтамасыз етеді. Олар келесі псевдожазбаларды құруға пара-пар: (бөлшек1, шойын, марка1, 4.03.98, 90), (бөлшек1, шойын, марка2, 4.03.98, 90), (бөлшек2, болат, марка1, 3.01.98, 35), (бөлшек2, болат, марка2, 3.01.98, 35), (бөлшек2, болат, марка3, 3.01.98, 35), (бөлшек3, алюминий, – , 17.02.98, 90), (бөлшек4, шойын, марка2, 6.05.98, 240).

Алынған псевдокесте басқару шешімдерін жоспарлау немесе қабылдау кезінде пайдалы болуы мүмкін. Себебі жауапты адамда әрбір бұйым бойынша тапсырыстардың орындалуының барлық мүмкін нұсқалары бар болуы керек.

Егер кестелер рөлін ауыстырса, нақтырақ айтқанда: Д3-кестесі негізгі, ал О3-кестесі қосымша болса, онда 1:М түріндегі байланыс аламыз. Әлбетте, О2 және Д2 кестелермен осылайша М:1 түріндегі байланыс алуға болады. Бұдан шығатыны, байланыстар (1:М немесе М:1) түрі қандай кесте негізгі, ал қандай қосымша болып табылатындығынан тәуелді.

М:М түріндегі байланыс

Ең жиі кездесетін М:М түріндегі байланыстар негізгі кестелердің бірнеше жазбаларына қосымша кестелердің бірнеше жазбалары сәйкес келетін жағдайда пайда болады.

5-мысал.

Негізгі О4-кестеде қай станокта қайсібір бригаданың жұмысшылары жұмыс жасайтыны туралы ақпараттар болсын. Д4-кесте жөндеу бригадасының қай жұмысшысы қай станокқа қызмет көрсететіндігі туралы мәліметтер жазылсын.

О4-кесте

* * +

Жұмыс жасайды	станокта
Иванов А.В.	станок1
Иванов А.В.	станок2
Петров Н.Г.	станок1
Петров Н.Г.	станок3
Сидоров В.К.	станок2

Д4-кесте

* * +

Қызмет көрсетеді	Станок
Голубев Б.С.	станок1
Голубев Б.С.	станок3
Зыков А.Ф.	станок2
Зыков А.Ф.	станок3

О4-кестесінің бірінші және үшінші жазбаларына Д4-кестесінің бірінші жазбасы сәйкес келеді (барлық осы жазбалардың мәні – “станок1”). О4-кестесінің төртінші жазбасына Д4-кестесінің екінші және төртінші жазбасы сәйкес келеді (“станок3” жазбасы).

Байланыстар өрісі анықтамасына сәйкес “О4+Д4” атаумен жаңа кесте құруға болады, оның жазбалары псевдожазба болады. Алынған кесте жазбаларына жұмыс жасау кезінде құрылатын ауысымдар мағынасын беруге болады. Ыңғайлылық үшін жаңа кестелер өрістерінің атаулары өзгертілген (көптеген заманауи ДҚБЖ осындай опрацияны ұсынады).

“О4+Д4”-кесте

Жұмыс жасайды	Станок	Қызмет көрсету
Иванов А.В.	станок1	Голубев Б.С.
Иванов А.В.	станок2	Зыков А.Ф.
Петров Н.Г.	станок1	Голубев Б.С.
Петров Н.Г.	станок3	Голубев Б.С.
Петров Н.Г.	станок3	Зыков А.Ф.
Сидоров В.К.	станок2	Зыков А.Ф.

Келтірілген кестені мысалы “Петров Н.Г жұмыс жасайтын станокқа кім қызмет көрсетеді?” деген сұраққа жауап беру үшін қолдануға болады.

Әлбетте, 1:1, М:М түріндегі байланыстар ұқсас, кесте арасында тәуелділік орнатпайды. Бұны тексеру үшін негізгі және қосымша кестенің орындарын ауыстыру мүмкін және байланыстыру жолымен ақпараттарды біріктіру орындалады. Нәтижелік “О4+Д4” және “Д4+О4” кестелерде бірінші және үшінші өрістердің орындалу реті және жазбалардың орналасу реті өзгерген.

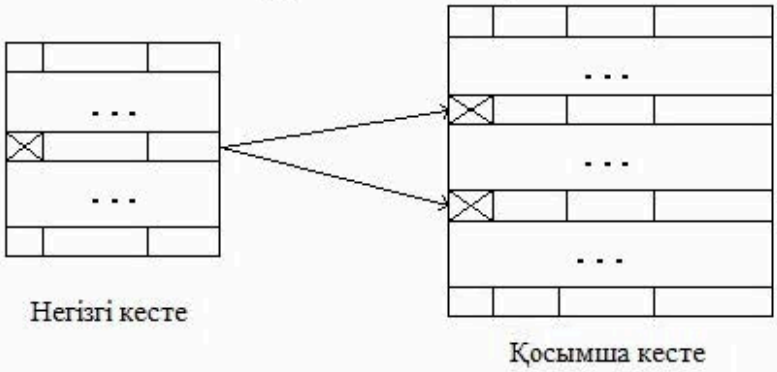
Ескерту.

Іс жүзінде байланысқа әдетте бірден бірнеше кесте қатысады. Сонымен қатар, кестелердің бірі бірнеше кестелермен әртүрлі байланысқа түсуі мүмкін. Байланысқан кестелер басқа кестелермен байланысты болса, онда иерархия немесе байланыс желісі құрылады.

3.4. Байланыс тұтастығын бақылау

Байланыстардың аталған түрлері арасынан неғұрлым кеңінен қолданылатын байланыстар 1:М, 1:1 түріндегі байланыстар деп есептеуге болады. М:1 түріндегі байланыс, мағынасы бойынша, 1:М байланыстың "айна-қатесіз суреті" болып табылады. М:М түріндегі байланыстар әлсіз байланыс түріне жатады. Сондықтан болашақта 1:М байланысты қарастырамыз.

Ескере кетейік, 1:М түріндегі байланыстар құрылған кезде негізгі кестелердің бір жазбасы (негізгі жазба) қосымша кестенің бірнеше жазбасымен (қосымша, бағынышты жазба) байланысты болады және 3.6-суреттегі жағдай орындалады.



3.6-сурет. Екі кесте жазбаларының 1:М байланысы

Байланыс тұтастығын бақылау әдетте екі кестенің құрамын келесі ережелердің орындалуына талдауды білдіреді:

- негізгі кестелердің әрбір жазбасы қосымша кестелердің ноль немесе бірнеше жазбаларына сәйкес келеді;
- қосымша кестеде негізгі кестенің түпкі жазбасы болатын жазбалар жоқ;
- қосымша кестелердегі әрбір жазба негізгі кестелердегі тек бір ғана түпкі жазбаға ие.

Кестелерде деректермен айла-амалдар жасау кезінде тұтастықты бақылау әрекетін сипаттайық. Екі кестенің деректерімен орындалатын үш негізгі операцияларды қарастырайық:

- жаңа жазбалар енгізу,
- жазбалармен іс-әрекеттер жасау,
- жазбаларды өшіру.

Қарастыру кезінде тұтастықты бақылауды ұйымдастырудың барлық мүмкін әдістерін қамтуға тырысайық. Нақты ДҚБЖ-да сипатталған әдіске ұқсас меншікті әдістер қолданылады.

Жаңа жазбалар енгізу кезінде жазбаларды тұтастықты бұзбай енгізу ретін анықтау мәселесі пайда болады. Жоғарыда келтірілген ережелерден шығатыны, деректерді алдымен негізгі кестеге, содан соң қосымша кестеге енгізу дұрыс болып табылады. Енгізу кезегі толық кесте немесе жекелеген жазбалар (бір уақытта бірнеше ашық кестеге енгізу жағдайы) деңгейінде орнатылған болуы мүмкін.

Негізгі кестелерді толтыру процесінде байланыстар өрісі мәндерін бақылау әдеттегі кілтті бақылау ретінде жүзеге асады (басқа жазбалар өрісі мәндерімен сәйкес келуі). *Қосымша* кестелердің байланыстар өрісін толтыру негізгі кестелер өрісі байланыстар мәндерімен сәйкес келуіне тексеріледі. Егер қосымша кестелердің байланыстар өрісіне қайтадан енгізілген мән бірде-бір негізгі кестелер жазбаларындағы мәнмен сәйкес келмесе, онда енгізудің ондай мәні оқшауланулы керек.

Жазбаларды түрлендіру. Байланысқан жазбалар өрісі құрамындағы өзгерістер, байланыстар өзгерістеріне қатыссыз, әлбетте, қарапайым түрде жүзеге асады. Байланыстар өрісіне өзгерістер енгізу механизмін қарастырайық.

Байланыстар өрісін редактірлеу кезінде қосымша кестелер айқын талап болып табылады да, жаңа байланыстар өрістері мәндері негізгі кестелердің сәйкес мәндерімен *ұқсас* болады. Яғни қосымша жазба *түпнұсқасын* өзгерте алады, бірақ онсыз болмауы керек.

Негізгі кестелердің байланыстар өрістерін редактірлеу келесі ережелердің біріне бағынады:

- бағынышты жазбалары жоқ жазбаны редактрлеу. Егер бағынышты жазба бар болса, онда байланыстар өрісімен іс-әрекеттер жасау оқшауланады;
- байланыстар өрістерінде өзгерістер енгізгенде негізгі жазба қосымша кестелердің барлық жазбаларының барлық байланыстар өрістеріне өзгерістерді жылдам жібереді (каскадты жаңарту).

Жазбаларды өшіру операцияларында байланысқан кесте үлкен еркіндікке ие, әлбетте, қосымша кестелердің жазбаларын қамтиды. Оларды өшіру іс жүзінде бақылаусыз жүзеге асуы керек. Негізгі кестелердегі жазбаларды өшіру келесі ережелердің біріне бағынады:

- бағынышты жазбалары жоқ жазбаны өшіруге болады;
- бағынышты жазбалары бар жазбаны өшіруге тыйым салынады немесе оны барлық бағынышты жазбаларымен бірге өшіру керек.

3.5. Сұраныстың теориялық тілдері

Қатынастармен орындалатын операцияларды екі топқа бөлуге болады. Алғашқы тобын жиындармен орындалатын операциялар құрайды, оларға жататын операциялар: біріктіру, қиылысу, айырмасы, бөлу және декарттық көбейту. Екінші топты қатынастармен орындалатын арнайы операциялар құрайды, оларға, жеке жағдайда, жататын операциялар: кескіндеу, жалғастыру, таңдау.

Әртүрлі ДҚБЖ-де қатынастармен орындалатын операциялардың, берілген ДҚБЖ-ның қандай да бір деңгейде мүмкіндіктерін және қиындықтарын анықтайтын қайсібір бөлімі жүзеге асқанда ДҚ-на сұраныс жасалады.

Реляциялық ДҚБЖ-де қатынастармен орындалатын операцияларды жүзеге асыру үшін Э.Кодд ұсынған сұраныстың теориялық тілдерінің *математикалық негіздерінің* екі тобын қолданады:

- Реляциялық алгебра;
- Реляциялық есептеулер.

Бұл тілдер нақты тілдердің деректермен іс-әрекеттер жасауының минимальды мүмкіндіктерін реляциялық үлгіге сәйкес көрсетеді және олар өздерінің мүмкіндіктері бойынша шамалас. Олардың арасында сұраныстарды түрлендірудің күрделі емес ережесі бар.

Реляциялық алгебрада барлық іс-әрекеттердің нәтижелері қатынастар болады. Реляциялық алгебра тілдері *процедуралық* болады, өйткені реляциялық ДҚ-на сұраныстың нәтижесі - қатынастарға қолданылатын реляциялық операторлар тізбегінің орындалуы кезінде есептелетін қатынас. Операторлар операндалардан тұрады, олардың рөлін қатынастар және реляциялық операциялар атқарады. Реляциялық операциялар нәтижесі қатынас болып табылады.

Есептеу тілдері, реляциялық алгебрамен салыстырғанда, *процедуралық емес* (сипаттамалық немесе декларативті) болады және бірінші ретті педикаттар көмегімен кортеждер немесе қатынастар домені қанағаттандыруы тиіс сұраныстарды сипаттайды.

Сәйкес тілдерді қолданып ДҚ-на сұраныс жасау тек қалаулы нәтиже туралы ақпаратты қамтиды. Осы тілдердің бар болуы үшін ережелер жиыны жасалған. Жеке жағдайда, бұл топтарға SQL тілі жатады.

Реляциялық алгебра тілдерін қарастыру және есептеу кезінде келесі кестелерді қамтитын деректер қорын қолданамыз:

- S (жеткізушілер);
- P (бөлшектер);
- SP (жеткізілім).

Осы кестенің алғашқы кілті сәйкесінше: П# (жеткізуші коды), Д# (бөлшектер коды) және құрама кілт (П#, Д#) болады. Кестенің құрамы 3.7-суретте келтірілген. Баяндау ыңғайлылығы үшін қарастырылған сұраныстар тілінде атрибут атауларында орыс алфавиті символдарын қолдануға шектеулер жоқ деп есептейік. SP-кестелерінде әрбір П# және Д# өрістері сыртқы кілттер қатынасы бойынша сәйкесінше S және P кестелерден жекеше болып табылады.

S

П#	Аты	Жағдайы	қала_П
S1	Сергей	20	Москва
S2	Иван	10	Киев
S3	Борис	30	Киев
S4	Николай	20	Москва
S5	Андрей	30	Минск

P

Д#	Атауы	Түрі	салмағы	Қала_Д
P1	гайка	қыздырылған	12	Москва
P2	бұрандама	жұмсақ	17	Киев
P3	бұранда	қатты	17	Ростов
P4	бұранда	қыздырылған	14	Москва
P5	сұққы	қатты	12	Киев
P6	түйреуіш	қыздырылған	19	Москва

SP

П#	Д#	Саны
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

3.7-сурет. Жеткізушілер, бөлшектер және жеткізілім кестесі

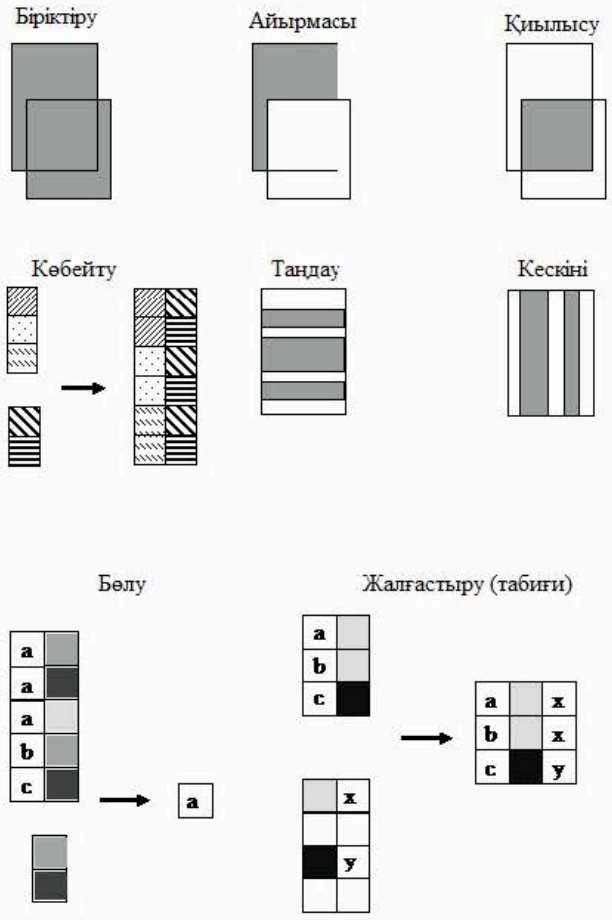
Домен атаулары (мүмкін мәндер жиынтығы) атрибут атауларымен сәйкес келеді деп есептейік: қалалар атауы жиынынан тұратын ортақ домені бар Қала_П (жеткізуші орналасқан қала) және Қала_Д (бөлшек шығарылатын қала) атрибуттарынан басқалары. Бұл домен атауы, мысалы, жай ғана Қала болуы мүмкін. Деректердің типі арқылы домендердің сипаттамалары: Д# – ұзындығы 5-ке тең символдар жолы, Аты – ұзындығы 20-ға тең символдар жолы, Жағдайы – ұзындығы 5-ке тең сандық, Қала – ұзындығы 15-ке тең символдар жолы, Д# – ұзындығы 6-ға тең символдар жолы, Типі – ұзындығы 6-ға тең символдар жолы, Салмағы – ұзындығы 5-ке тең сандық, Саны – ұзындығы 5-ке тең сандық.

3.6. Реляциялық алгебра

Реляциялық алгебра теориялық сұраныстар тілі ретінде реляциялық есептеулермен салыстырғанда қатынастармен орындалатын әрекеттерді неғұрлым көрнекі сипаттайды.

Сұраныстар тілінің мысалы, реляциялық алгебраға негізделген, ISBL тілі болып табылады (Information System Base Language – ақпараттық жүйелердің базалық тілі). Реляциялық алгебра негізінде тұрғызылған сұраныстар тілі заманауи ДҚБЖ-ларда кеңінен таралмады. Бірақ, басқа тілдерде қолданылатын реляциялық операциялар мағынасын түсіну пайдалы.

Кодд ұсынған реляциялық алгебра нұсқасы келесі негізгі *операцияларды* іске қосады: *біріктіру, айырмасы (алу), қиылысу, декарттық (тура) көбейту (немесе көбейту), таңдау (селекция, шектеу), кескіні, бөлу және жалғастыру*. Осы операциялардың қысқартылған графикалық бейнеленуі 3.8-суретте келтірілген.



3.8-сурет. Реляциялық алгебраның негізгі операциялары

Коддтың реляциялық алгебрасы, Дейттің әділетті ескертуі бойынша, бірнеше кемшіліктерге ие. Біріншіден, аталған сегіз операциялар өз функцияларын қамтуы бойынша, бір жағынан, тым көп, өйткені ең аз қажетті жинақ бес операцияларды құрайды: біріктіру, алу, көбейту, кескіні және таңдау. Басқа үш операцияларды (қиылысу, жалғастыру және бөлу) бес ең аз қажетті операциялар арқылы анықтауға болады. Мысалы, жалғастыру – бұл көбейтуді таңдау кескіні.

Екіншіден, осы сегіз операциялар реляциялық алгебра принциптеріне негізделген нақты ДҚБЖ тұрғызу үшін жеткіліксіз. Келесі операцияларды қамтитын кеңейтулер қажет: атрибуттардың атауын өзгерту, жаңа есептелетін атрибуттар қалыптастыру, қорытынды функцияларды есептеу, күрделі алгебралық өрнектер құру, меншіктеу, салыстыру және т.б.

Аталған операцияларды неғұрлым жан-жақты қарастырайық, алдымен – Коддтың реляциялық алгебрасы операцияларын, содан соң – Дейт енгізген қосымша операцияларды.

Коддтың реляциялық алгебрасы операцияларын екі топқа бөлуге болады: *бастапқы теориялық-жалпылама және арнайы реляциялық*. Операциялардың алғашқы тобы жиындар теориясының классикалық операцияларын іске қосады: біріктіру, айырмасы, қиылысу және көбейту. Екінші топ қарапайым теориялық-жалпылама операциялардың деректермен іс-әрекеттер жасаудың нақты тапсырмалары бағытында дамуы болып саналады, оның құрамына келесі операциялар кіреді: кескіні, селекция, бөлу және жалғастыру.

Реляциялық алгебра операциялары бір қатынаспен (мысалы, кескіні) немесе екі қатынастармен (мысалы, біріктіру) орындалуы мүмкін. Бірінші жағдайда операция унарлық, ал екіншісінде – бинарлық деп аталады. Бинарлық операциялар орындалуы кезінде қатысатын қатынастар құрылымы бойынша сәйкес болуы керек.

Қатынастар *құрылымдарының сәйкестігі* сәйкес домендердің атрибут атаулары мен түрлерінің сәйкестігін білдіреді. Сәйкестендірудің жеке жағдайы ұқсастық (тура келу) болып табылады. Бастапқы қатынастарда (атаулардың сәйкес болмауы қажет жағдайында) атрибут атауларындағы шиеленістерді жою үшін, сонымен қатар нәтижелік қатынастар атрибуттары атаулары кез-келген болуы үшін атрибуттардың атауын өзгерту операциясы қолданылады. Нәтижелік қатынастар құрылымы анықталған ережелерге сәйкес бастапқы қатынастардың қасиеттері құрылымын иеленеді. Қарастырылатын бинарлық реляциялық операциялардың көбісінде бастапқы қатынастар тақырыптары ұқсас, өйткені бұл жағдайда нәтижелік қатынастардың тақырыптарымен мәселелер туындамайды (жалпы жағдайда, тақырыптар сәйкес келмеуі мүмкін, онда нәтижелік қатынастардың тақырыптарын қалыптастыру ережелерін ескеру керек).

Өлшемдері бірдей екі үйлесімді R1 және R2 қатынастарын б***іріктіру*** (R1 UNION R2) бастапқы қатынастардың (қайталауларды есептемегенде) барлық элементтерін қамтитын R қатынас болып табылады.

Мысал 1. Қатынастарды біріктіру.

R1 қатынасы Лондоннан жеткізушілер жиыны, ал R2 қатынасы – P1 бөлшекті жеткізетін жеткізушілер жиыны болсын. Онда R қатынасы Лондонда тұратын жеткізушілерді немесе P1 бөлшекті шығаратын жеткізушілерді және т.б. білдіреді.

R1

П#	Аты	Мәртебесі	Қала_П
S1	Сергей	20	Москва
S4	Николай	20	Москва

R2

П#	Аты	Мәртебесі	Қала_П
S1	Сергей	20	Москва
S2	Иван	10	Киев

R (R1 UNION R2)

П#	Аты	Мәртебесі	Қала_П
S1	Сергей	20	Москва
S2	Иван	10	Киев
S4	Николай	20	Москва

Өлшемдері бірдей екі үйлесімді R1 және R2 қатынастарының ***айырмасы*** (R1 MINUS R2) R1 жататын, бірақ R2 қатынасына жатпайтын кортеждер жиынынан тұратын қатынас. Алдыңғы мысалдағы R1 және R2 қатынастары үшін R қатынасы Лондонда тұратын, бірақ P1 бөлшекті шығармайтын жеткізушілер болып табылады, яғни R={(S4, Николай, 20, Москва)}.

Азайту операциясының нәтижесі операндалардың жүзеге асу ретінен тәуелді екенін ескере кетейік, яғни R1 MINUS R2 және R2 MINUS R1 – бірдей емес.

Өлшемдері бірдей екі үйлесімді R1 және R2 қатынастарының ***қиылысуы*** (R1 INTERSECT R2) бір уақытта қатынастардың екеуіне де жататын кортеждерден тұратын R қатынасын береді. R1 және R2 қатынастары үшін нәтижелік R қатынасы барлық Лондонда тұратын, P1 бөлшекті шығаратын жеткізушілерді көрсетеді. R қатынасы жалғыз элементтен тұрады (S1, Сергей, 20, Москва).

Атрибуттарының атаулары бірдей емес к1 дәрежелі R1 қатынастары және к2 дәрежелі R2 қатынастарын ***көбейту*** (R1 TIMES R2) тақырыбы R1 және R2 қатынастарының тақырыптарын жалғастыруды білдіретін, ал құрамына алдымен R1 жиынының алғашқы к1 элементтері, содан соң R2 жиынының к2 элементтері кіретін (к1+к2) дәрежелі қатынас. Атаулары бірдей бір немесе бірнеше атрибуттары бар екі қатынасты көбейту қажет болған кезде атауын өзгерту RENAME операциясын қолданылады.

Мысал 2. Қатынастарды көбейту.

R1 қатынасы барлық ағымдағы жеткізушілер номерлерінің {S1, S2, S3, S4, S5} жиыны, ал R2 қатынасы – барлық ағымдағы бөлшектер номерлерінің {P1, P2, P3, P4, P5, P6} жиыны болсын. R1 TIMES R2 операциясының нәтижесі "жеткізуші-бөлшек" түріндегі барлық жұптар жиыны болып табылады, яғни {(S1,P1), (S1,P2), (S1,P3), (S1,P4), (S1,P5), (S1,P6), (S2,P1), ... , (S5,P6)}.

Жиындар теориясында қатынастарды тікелей көбейту нәтижесі әрбір элементі алғашқысы R1 жататын, екіншісі R2 жататын элементтер жұбы жиыны болып табылатынын ескере кетейік. Сондықтан бинарлық қатынастарды декарттық көбейту кортеждері келесі түрдегі кортеждер болып табылады: ((а, б), (в, г)), мұнда (а, б кортежі R1 қатынасына жатады, ал (в, г) кортежі R2 қатынасына жатады. Реляциялық алгебрада қатынастарды тікелей көбейтудің кеңейтілген нұсқасы қолданылады, онда екі бастапқы қатынастар элементтері қосылады да, нәтижесінде тек артық жақшаларды өшіру ғана қалады, яғни (а, б, в, г).

R қатынастарынан f формуласы бойынша ***таңдау*** (R WHERE f) R қатынастарының атаулары мен құрамы өзгертілмеген, f формуласымен берілген логикалық өрнекті қанағаттандыратын жаңа қатынас болып саналады. Формуланы жазу үшін операндалар – атрибут атаулары (немесе бағана номерлері), тұрақтылар, логикалық операциялар

(AND – және, OR – немесе, NOT – емес), салыстыру және жақшалар операциялары қолданылады.

Мысал 3. Таңдау.

P WHERE Салмағы < 14

Д#	Атауы	Тип	Салмағы	Қала_Д
P1	гайка	қыздырылған	12	Москва
P5	сұққы	қатты	12	Киев

SP WHERE П# = "S1" AND Д# = "P1"

П#	Д#	Саны
S1	P1	300

A қатынастарының X, Y, ... , Z (A [X, Y, ... , Z]), мұндағы {X, Y, ... , Z} жиын атрибутына ***кескіні ал*** қатынастары тақырыптары атрибуттарының толық тізімінің ішкі жиыны болып табылады, X, Y, ... , Z тақырыпты және ал қатынастарын қамтитын қатынас болып саналады, қайталанатын кортеждерді есептемегенде. Тізімде X, Y, ... , Z бірдей атрибуттардың қайталануына тыйым салынады.

Кескіндеу операциясы келесі қосымша нұсқалар жазбасына рұқсат береді:

- Атрибуттар тізімі барлық атрибуттарды көрсетуді білдіреді (барабар кескіндеу операциясы);
- R[ ] түріндегі өрнек *бос* кескіндеуді білдіреді, оның нәтижесі бос жиын болады;
- кескіндеу операциясының ерікті қатынастарға, сонымен қатар таңдау нәтижесінде қолданылуы мүмкін.

Мысал 4. Кескіндеу.

P [Тип, Қала_Д]

Тип	Қала_Д
қыздырылған	Москва
жұмсақ	Киев
қатты	Ростов
қатты	Киев

(S WHERE Қала_П="Киев") [П#]

П#	Қала_П
S2	Киев
S3	Киев

A және B атрибутты R1 қатынастарын B атрибутты R2 қатынасына ***бөлу*** (R1 DIVIDEBY R2) нәтижесі R1 қатынастарда (r, s) кортеждері бар болса, сонымен қатар s мәндер жиыны R2 қатынастарының B атрибуты мәндерінің жиынын іске қосатын болса, тақырыбы ал болатын және r кортеждерден тұратын R қатынасы болып табылады. Мұндағы A және B қарапайым немесе құрама атрибуттар, сонымен қатар B атрибуты – бір доменде анықталған ортақ атрибут.

Мысал 5. Бөлу қатынастары.

R1 – кескіні SP [П#, Д#], ал R2 –Д# тақырыпты және {P2, P4} денелі қатынас болсын, онда R1-ді R2-ге бөлу нәтижесі П# тақырыпты және {S1, S4} денелі R қатынасы болады.

R1

П#	Д#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

R2

Д#
P2
P4

R1 DIVIDEBY R2

П#
S1
S4

R1 және R2 қатынастарын f формуласымен берілген шарт бойынша ***жалғастыру*** C_f(R1, R2) R1 және R2 қатынастарды декарттық көбейтіп, нәтижені f формуласы бойынша таңдау арқылы алынған R қатынасы болып саналады. f формуласын жазу ережесі селекция операциясы сияқты.

Басқаша айтқанда, ал атрибуты бойынша R1 қатынастарды B атрибуты бойынша R2 қатынасымен (қатынастарда ортақ атрибут атаулары болмаса) жалғастыру келесі түрдегі операцияны орындау нәтижесі болып табылады:

(R1 TIMES R2) WHERE A Θ B,

мұндағы Θ – бір (бірнеше – құрама атрибут үшін) доменде анықталған атрибуттармен логикалық өрнек. C_f(R1, R2) жалғастыруы, мұнда f ерікті формула (бұдан әрі қарастырылатын жеке жағдайлардан айырмашылықтары) кейде *Θ-жалғастыру* деп те аталады.

Іс жүзінде жалғастырудың маңызды жеке жағдайлары эквижалғау және кәдімгі жалғастыру болады.

Эквижалғау операциясы формула операндалар теңдігін берген жағдаймен түсіндіріледі. Жоғарыда келтірілген мысал эквижалғау операциясының жеке жағдайда бір бағана бойынша жалғауды көрсетеді. Кей жағдайда екі қатынастарды эквижалғау атрибуттары екі қатынастарда сәйкесінше бірдей атаулар және домендерге ие бағаналарға орындалады. Бұл жағдайда ортақ атрибут бойынша эквижалғау туралы айтылады.

Табиғи жалғастыру (JOIN) операциясы ортақ атрибутты (қарапайым немесе құрама) екі қатынастарға қолданылады. Бұл атрибуттар қатынастарда бірдей атауға (атаулар жиынтығы) ие және бір доменде (домендерде) анықталған.

Табиғи жалғастыру операциясының нәтижесі R қатынасы болып табылады, ол R1 және R2 қатынастарын ортақ атрибуты бойынша эквижалғау (екі қатынастың атрибуттарының біріккен жиынтығы) кескіні болып саналады.

Мысал 6. Θ-жалғастыру.

S және P қатынастарын сәйкесінше Қала_П және Қала_Д атрибуттары бойынша жалғастыру керек, сонымен қатар нәтижелік қатынастар кортеждері "үлкен" (алфавит бойынша) қатынасын қанағаттандыруы керек.

(S TIMES P) WHERE Қала_П > Қала_Д

П#	Аты	Мәрте-бесі	Қала_П	Д#	Атауы	Тип	Сал-мағы	Қала_Д
S2	Иван	10	Киев	P1	гайка	қыздырылған	12	Москва
S2	Иван	10	Киев	P4	бұранда	қыздырылған	14	Москва
S2	Иван	10	Киев	P6	түйреуіш	қыздырылған	19	Москва
S3	Борис	30	Киев	P1	гайка	қыздырылған	12	Москва
S3	Борис	30	Киев	P4	бұранда	қыздырылған	14	Москва
S3	Борис	30	Киев	P6	түйреуіш	қыздырылған	19	Москва

Мысал 7. Эквижалғау.

Қаланы сипаттайтын S және P қатынастарын (S қатынастарда – бұл Қала_П, ал P қатынастарда – Қала_Д) ортақ атрибуты бойынша табиғи жалғастыру қажет болсын. Операция шарты жалғастыру орындалатын атрибут атауларының бірдей болуын талап ететіндіктен, онда атрибуттардың атауын өзгерту RENAME операциясы қолданылады.

(S RENAME Қала_П AS Қала) JOIN (P RENAME Қала_Д AS Қала)

П#	Аты	Мәрте-бесі	Қала	Д#	Атауы	Типі	Сал-мағы
S1	Сергей	20	Москва	P1	гайка	қыздырылған	12
S1	Сергей	20	Москва	P4	бұранда	қыздырылған	14
S1	Сергей	20	Москва	P6	түйреуіш	қыздырылған	19
S2	Иван	10	Киев	P2	бұранда-ма	жұмсақ	17
S2	Иван	10	Киев	P5	сұққы	қатты	12
S3	Борис	30	Киев	P2	бұранда-ма	жұмсақ	17
S3	Борис	30	Киев	P5	сұққы	қатты	12
S4	Николай	20	Москва	P1	гайка	қыздырылған	12
S4	Николай	20	Москва	P4	бұранда	қыздырылған	14
S4	Николай	20	Москва	P6	түйреуіш	қыздырылған	19

Дейт ұсынған реляциялық алгебраның қосымша операциялары келесі операцияларды қамтиды.

Атауын өзгерту операциясы қатынастар атрибуттарының атауын және түрін өзгертуге мүмкіндік береді:

RENAME <Бастапқы қатынас> <атрибуттың ескі атауы> AS <атрибуттың жаңа атауы>,

мұнда <Бастапқы қатынас> қатынастар атауымен немесе реляциялық алгебра өрнегімен беріледі. Соңғы жағдайда өрнекті дөңгелек жақшаға алады.

Мысалы, RENAME Қала_П AS Жеткізушінің_орналасқан_қаласы.

Ескерту.

Ыңғайлылық үшін өрнектер жазбасы бір уақытта бірнеше атрибуттың атауын өзгерте алады. Ол үшін Дейт келесі түрде жазылатын *көптеген атауды өзгерту* операциясын ұсынды:

RENAME <қатынас.> <атр.ескі атауы 1> AS < атр.жаңа атауы 1>,< атр.ескі атауы 2> AS < атр.жаңа атауы 2>, ... ,< атр.ескі атауы N> AS < атр.жаңа атауы N>.

Кеңейту операциясы кезінде бастапқы қатынасқа ұқсас, бірақ мәні қайсыбір скаляр есептеулер жолымен алынатын жаңа қатынас туындайды. Кеңейту операциясының түрі:

EXTEND <Бастапқы қатынас> ADD <өрнек> AS <жаңа атрибут>,

Мұнда бастапқы қатынасқа (ADD кілттік сөзі) ережелерге сай есептелген <өрнек> берілген <жаңа атрибут> қосылады. Бастапқы қатынас қатынастар атауымен және реляциялық алгебра өрнегі көмегімен, дөңгелек жақшаға алынып берілуі мүмкін. Сонымен қатар жаңа атрибут атауы бастапқы қатынастардың тақырыбына кірмеуі керек және <өрнектер>-де қолданылмайды. Қарапайым арифметикалық операциялар және салыстыру операцияларымен қатар, өрнектерде қорытынды деп аталатын әртүрлі функцияларды қолдануға болады: COUNT (саны), SUM (қосынды), AVG (орташа), MAX (ең үлкен), MIN (ең аз).

Мысалы, EXTEND (P JOIN SP) ADD (Салмағы * Саны) AS жалпы_Салмағы.

Ескерту.

- Кеңейту операциясын қолданып, атрибут атауын өзгертуді орындауға болады. Осы үшін өрнектерде атрибут атауын көрсету, AS құрылымында осы атрибуттың жаңа атауын анықтау керек, содан соң алынған қатынастардың атрибуттар жиынына, ескі атрибутты қоспағанда, кескінін орындайды. Осылайша, (EXTEND S ADD Қала_П AS Қала) [П#, Аты, Мәртебесі, Қала] жазбасы S RENAME Қала_П AS Қала құрылымына эквивалентті.
- Осылайша, *атауын өзгерту* операциясына сәйкес Дейт *жиынтық көбейту* операциясын анықтадық, ол бір синтаксистік құрылымда бірнеше жаңа атрибуттарды есептеуге мүмкіндік береді. Операция келесі түрде жазылады:

EXTEND <қатынас.> ADD <өрнек 1> AS <атр.1>, <өрнек 2> AS <атр.2>, ... ,<өрнек N> AS <атр.N>.

Қорытындасын шығару SUMMARIZE операциясы "вертикальды" немесе топтық есептеуді орындайды және келесі түрде жазылады:

SUMMARIZE <бастапқы қатынас> BY (<атрибуттар тізімі >) ADD <өрнек> AS <жаңа атрибут>,

Мұнда бастапқы қатынас қатынастар атауымен немесе дөңгелек жақшаға алынған реляциялық алгебра өрнегімен беріледі, <атрибуттар тізімі> үтірлермен бөлінген атрибуттар тізімі болып саналады, <өрнек.> – EXTEND операциясы өрнегіне ұқсас скалярлық өрнек, ал <жаңа атрибут> – құрылатын атрибут атауы. Атрибуттар тізімінде және өрнектерде <жаңа атрибут> қолданылмауы керек.

SUMMARIZE операциясының нәтижесі атрибуттар тізімінен тұратын тақырыпты, жаңа атрибутпен кеңейтілген R қатынасы болып табылады. Қатынастар денесін R алу үшін алдымен бастапқы қатынастардың A1, A2, ... , AN атрибуттарына кескіндеу орындалады (кескінді R1 деп атайық), содан соң әрбір кортеж үшін кескіндеу жаңа (N+1)-атрибутпен кеңейтіледі. Себебі кескіндеу, қағидаға сәйкес, бастапқы қатынасқа қатынасы бойынша (бірдей кортеждер жойылады) кортеждер санын азайтады.

Мысал 8. Қорытынды жасау.
Жеткізушілердің әрқайсысы бойынша жеткізілімдер санын анықтау талап етілсін.

SUMMARIZE SP BY (П#) ADD COUNT AS жеткізілім_саны

П#	жеткізілім_саны
S1	6
S2	2
S3	1
S4	3

Ескере кетейік, COUNT функциясы бастапқы қатынастарда топтардың әрқайсысындағы кортеждер санын анықтайды.

Жалпылама қорытынды шығару операциясы атауын өзгерту және кеңейту операцияларына сәйкес бір уақытта бірнеше "вертикальды" есептеулер орындайды және нәтижелерді жаңа атрибуттар бөліміне жазады. Мұндай операцияның қарапайым мысалы:

SUMMARIZE SP BY (Д#) ADD SUM Саны AS жеткізілімнің_жалпы_саны
AVG Саны AS жеткізілімнің_орташа_саны.

Қатынастар денесін өзгертуге мүмкіндік беретін негізгі операторларға келесі реляциялық операцияларды жатқызамыз: *меншіктеу, кірістіру, жаңарту және өшіру.*

Меншіктеу операциясын келесі түрде көрсетуге болады:

<өрнек-мақсат> := <өрнек-түпнұсқа>,

Мұндағы екі өрнек те қатынастар құрылымы бойынша үйлесімділікті береді (нақтырақ айтқанда, эквивалентті). Өрнектердің әдеттегі жағдайы: сол жақ бөлімде – қатынастар атауы, ал оң жақта – реляциялық алгебраның қайсыбір өрнегі. Меншіктеу операциясын орындау қатынастардың ағымдағы мәнін (егер қатынастар денесі бос болса, онда алғашқы мәні) жаңа мәнге ауыстыруға әкеледі.

Меншіктеу операциясы көмегімен қатынастардың барлық мәндерін ауыстырып қана қоймай, жаңадан кортеждер енгізуге немесе жоюға болады. Мысал келтірейік:

S := S UNION {{< П# : 'S6' >, < Аты : 'Борис' >, < Мәртебесі : '50' >, < Қала_П : 'Мадрид' >}}.

Неғұрлым қолайлы операциялармен (кірістіру, жаңарту және өшіру) өзгерістер енгізу қатынастар болады.

Кірістіру INSERT операциясы келесі түрде жазылады:

INSERT <өрнек-түпнұсқа> INTO <өрнек-мақсат>,

Мұндағы екі өрнек те құрылымы бойынша үйлесімді болуы керек. Операцияны орындау <өрнек-Түпнұсқа> есептеулерге және алынған кортеждерді <өрнек-мақсат> берілген қатынасқа қоюға әкеледі.

Мысалы: INSERT (S WHERE Қала_П='Москва') INTO Temp.

Жаңарту UPDATE операциясы келесі түрде жазылады:

UPDATE <өрнек-мақсат> <элементтер тізімі>,

Мұнда <элементтер тізімі> үтірлермен бөлінген <атрибут> := <скалярлық өрнек> меншіктеу операциялары тізбегі болып саналады. Жаңарту операциясының орындалу нәтижесі өрнекпен берілген қатынастар атрибуттарының мәндеріне сәйкес меншіктелгеннен кейін алынатын қатынас болып табылады.

Мысалы, UPDATE P WHERE Тип='қыздырылған' Қала := 'Киев'. Бұл операция Қала атрибуты мәнін (бастапқыда қандай болғанынан тәуелсіз) жаңа 'Киев' мәніне ауыстыруды ұсынады. Мұндай кортеждердегі P қатынастары атрибуты мәні 'қыздырылған'.

Өшіру DELETE операциясы келесі түрде жазылады:

DELETE <өрнек-мақсат>,

мұнда <өрнек-мақсат> өшірілетін кортеждерді сипаттайтын реляциялық өрнек болып саналады.

Мысалы, DELETE S WHERE Мәртебесі < 20.

Реляциялық салыстыру операциясы екі қатынасты тікелей салыстыру үшін қолданылуы мүмкін. Жазылуы:

<өрнек1> Θ <өрнек2>,

Мұндағы екі өрнек те қатынастар құрылымы бойынша үйлесімділікті береді, ал Θ белгісі – салыстырудың келесі операторларының біреуі: = (тең), ≠ (тең емес), ≤ (меншікті ішкі жиыны), < (кіші жиыны), ≥ (меншікті жоғары жиыны), > (жоғары жиыны).

Мысалы, "жеткізуші қалалар және бөлшектерді сақтау қалалары сәйкес келеді ма?" салыстыруын келесі түрде жазуға болады: S [Қала_П] = P [Қала_Д].

Өрнектерді жазудың негізгі ережелері. Жоғарыда байқалғандай, ерікті реляциялық операциялар нәтижесі қатынас болып табылады, ол өз кезегінде басқа реляциялық операцияларға қатысуы мүмкін. Реляциялық алгебрада бұл қасиет *тұйықтанушылық қасиет* деп аталады.

Тұйықтанушылық қасиет қарастырылған қарапайым операциялар негізінде *енгізілген реляциялық алгебра өрнегін* жазуға мүмкіндік береді: біріктіру, кескіні, қиылысу, таңдау және т.б.

Ерікті реляциялық алгебра өрнегін жазу кезінде келесілерді ескеру керек.

1. Реляциялық алгебрада орындалатын операциялар арасында басымдылық анықталуы керек (мысалы, қиылысу операциясы біріктіру операциясына қарағанда неғұрлым басым), оны өрнектерді жазу кезінде ескеру керек. Өрнектерде операциялардың орындалу ретіне өзгерістер енгізу үшін дөңгелек жақшаларды қолдануға болады.

2. Тепе-тең түрлендірулер бар, олар бір өрнекті әртүрлі жазуға мүмкіндік береді. Мысалы, келесі өрнектер эквивалентті (мұнда ал – қатынас, C, C1, C2 – өрнектер):

A WHERE C1 AND C2 және (A WHERE C1) INTERSECT (A WHERE C2),

A WHERE C1 OR C2 және (A WHERE C1) UNION (A WHERE C2),

A WHERE NOT C және A MINUS (A WHERE C).

3. Өрнек құру кезінде операцияларға қатысатын қатынастардың үйлесімділігін қамтамасыз ету керек. Тақырыптарға өзгерістер енгізу қажеттілігі кезінде атрибуттардың атауын өзгертуді орындаған жөн.

3.7. Реляциялық есептеулер

Жоғарыда байқалғандай, реляциялық алгебра мен реляциялық есептеулер арасында айтарлықтай айырмашылық жоқ, бірінші жағдайда ізделінді нәтиже алу процесі орындалуы керек операциялар жиынын көрсету жолымен сипатталады, ал екіншісінде – ізделінді қатынастар қасиеттері көрсетіледі.

Неғұрлым төмен деңгейде тәсілдемені қарастыру эквивалентті, , өйткені кез-келген реляциялық алгебра өрнегі реляциялық есептеулер өрнектеріне эквивалентті семантикалық түрленген болуы мүмкін және керісінше. Мұндай түрлендіру мүмкіндігі көптеген авторлармен дәлелденген, жеке жағдайда, осы үшін Коддтың *редукция алгоритмін* қолдануға болады.

Аталған алгоритмді түрлендіру үшін мазмұндық деңгейде реляциялық алгебра және реляциялық есептеулер көмегімен бірдей сұраныс жасауды қалыптастыру деңгейін қарапайым мысалмен көрсетейік.

Сұраныс келесі түрде берілген болсын: "P2 бөлшектерін шығаратын жеткізушілердің номерін және орналасқан қала номерін алу керек".

Сөз жүзінде бұл сұраныстың алгебралық нұсқасы былай сипатталады:

- П# атрибуты бойынша S және SP қатынастарын табиғи жалғастыру;
- Бұл жалғастырулар нәтижесінен P2 бөлшекті кортеждерді таңдап алу (Д# өрісінде P2 жолы болуы керек);
- П# және Қала_П атрибуттарымен алдыңғы операциялар нәтижесін ұлғайту.

Осы сұранысты реляциялық есептеулер терминдерінде келесі түрде тұжырымдауға болады: "жеткізушілер П# және ҚалаП үшін атрибуттар алу, жеткізушілер SP қатынастарда П# атрибуты және Д# атрибутта P" мәнімен жеткізілім жасайды".

Сұраныстың орындалу нәтижесі R қатынасы түрінде болады:

П#	Қала_П
S1	Москва
S2	Киев
S3	Киев
S4	Москва

Реляциялық алгебрадан реляциялық есептеулердің артықшылығы пайдаланушыдан не талап етілсе сол сұраныстың орындалу алгоритмі тұрғызылады. ДҚБЖ бағдарламасының (зиялылығымен) өзі тиімді алгоритм тұрғызады.

Ескере кетейік, қойылған есепті жедел жадыны неғұрлым тиімді пайдаланып шешуге болады. Реляциялық алгебра операциялары терминдерінде шешімнің неғұрлым тиімді нұсқасы төмендегідей:

- SP қатынастардан P2 бөлшектеріне қатысты кортеждерді таңдау керек;
- S қатынастарымен алдыңғы қадамда алынған қатынастарды табиғи жалғастыру орындалады;
- Ағымдағы қатынасты П# және Қала_П. атрибуттарымен жобалайды.

Осы алгоритмнің жүзеге асуы кезінде жадының үнемделуі алдыңғы нұсқамен салыстырғанда операцияларда қолданылатын, аралық нәтижелерді сақтауға арналған уақытша кестелердің өлшемдерінің азаюы есебінен жүзеге асады. Егер алдыңғы жағдайда уақытша кестелер өлшемі 12*6 (12 жол, 6 бағана) болса, онда соңғы жағдайда – 4*6.

Реляциялық есептеулердің математикалық негізі предикаттарды есептеу болып табылады – математикалық логиканың бөлімдерінің бірі. Реляциялық есептеулер түсінігін деректер қорымен жұмыс тілі ретінде алғаш рет Кодд ұсынды. Ол бағдарламалар жүзеге асқан QUEL тілінің түпнұсқасы ALPHA тілін де құрды , ол біршама уақыт SQL тілімен бәсекелес болды.

Есептеудің екі нұсқасы бар: *Кортеждерді есептеу* және *Домендерді есептеу*. Бірінші жағдайда қатынастарды сипаттау үшін мүмкін мәндері қатынастар кортеждері болатын айнымалылар, ал екінші жағдайда – домен элементтері қолданылады.

Кортеждерге негізделген реляциялық есептеулер (***Кортеждерді есептеу***) ALPHA тілімен ұсынылған және жүзеге асқан. Онда алдымен қолданылатын айнымалылар сипатталады, содан соң қайсыбір өрнектер жазылады.

Есептеулердің сипаттамалық бөлімдерін келесі түрде жазуға болады:

RANGE OF <айнымалы> IS <тізімі>,

мұнда бас әріптермен тілдің кілттік сөздері жазылған, <айнымалы> – кортеж айнымалысы идентификаторы (мәндер облысы), ал <тізімі> – үтірлермен бөлінген бір немесе бірнеше элементтер тізбегі, яғни x_1 [, x_2 [... , x_n]...] түріндегі құрылым.

RANGE құрылымы айнымалы идентификаторын және оның мүмкін мәндері облысын көрсетеді. x_1 [, x_2 [... , x_n]...] элементтер тізімі әрқайсысы қатынас немесе қатынастар өрнегі болып табылатын элементтерді қамтиды (өрнектерді жазу реті кейінірек сипатталады). Барлық элементтер тізімі түрі бойынша үйлесімді болуы керек, яғни қатынастардың сәйкес элементтері ұқсас тақырыптарға ие болуы керек. <айнымалы> мүмкін мәндері облысы тізімнің барлық элементтері мәндерін біріктіру жолымен құрылады. Ендеше, RANGE OF T IS X1,X2 түріндегі жазба айнымалыны анықтау T облысы қатынастардың барлық мәндерін өзіне қосытынын білдіреді, ол X1 және X2 қатынастарын біріктіру болып табылады.

Мысал 1. Сипаттамалар нұсқалары.

```
RANGE OF SX IS S;
RANGE OF SPX IS SP;
RANGE OF SY IS (SX) WHERE SX.Қала_П = 'Москва',
                (SX) WHERE EXISTS SPX (SPX.П# = SX.П# AND
                                       SPX.Д# = 'P1');
```

Алғашқы екі мысалда SX және SPX айнымалылары сәйкесінше S және SP қатынастарымен анықталған. Үшінші мысал кортеж айнымалысын анықтау кезіндегі өрнектер жазбасының мысалы болады. SY айнымалысы S қатынастар кортеждері жиынының мәндерін қабылдауы мүмкін, Лондоннан жеткізушілер немесе P1 бөлшекті жеткізушілер ретінде.

Есептеу өрнектерінің синтаксисі төменде қарастырылады. *Өрнектер жазбасы* сұранысты қалыптастыратын Бэкуса-Наура формалар көмегімен келесі түрде жазылады:

<өрнек> ::= (y_1 [, y_2 [... , y_m]...]) [WHERE wff]
y_i ::= {<айнымалы> | <айнымалы>.<атрибут> } [AS <атрибут>]
wff ::= <шарт> |
NOT wff |
<шарт> AND wff |
<шарт> OR wff |
IF <шарт> THEN wff |
EXISTS <айнымалы> (wff) |
FORALL <айнымалы> (wff) |
(wff)

Өрнектер жазбасының жалпы мағынасы нәтижелік (мақсатты) қатынастардың атрибуттарын санап шығу, бұл атрибуттар wff (well formulated formula – дұрыс құрылған формула) формуласының ақиқаттық шартын қанағаттандыруы керек. Мақсатты қатынастардың атрибуттар тізімі немесе *мақсатты тізім*, реляциялық алгебра терминдерінде кескіндеу операциясын, ал wff формуласы кортеждер селекциясын анықтайды.

<айнымалы>.<атрибут> жұбында алғашқы құраушы кортеж айнымалысын (RANGE құрылымымен анықталған) көрсету үшін, ал екіншісі – кортеж айнымалысы өзгеретін қатынастарды анықтау үшін қолданылады. "AS <атрибут>" міндетті емес бөлімі мақсатты қатынастар атауын өзгерту үшін қолданылады. Егер ол болмаса, онда мақсатты қатынастар атрибут атауы сәйкесінше бастапқы қатынастар атрибут атауын иеленеді.

Мысал 2. <айнымалы>.<атрибут> жазба жұбы нұсқалары.

SX.П#
SX.П# AS Қала_Жеткізуші
SX
SX.П#, SX.Қала_П AS Қала_Жеткізуші, PX.Д#, PX.Қала_Д# AS Қала_Бөлшектер

Келтірілген анықтамада wff <шарт> болып саналады немесе жақшаға алынған wff формула немесе келесі түрдегі қарапайым салыстыру:

<операнд1> Θ <операнд2>,

мұнда кез-келген операнда ретінде айнымалы немесе скалярлық тұрақты алынады, ал Θ символы салыстыру =, ≠, >, ≥, <, ≤ және т.б операциясын білдіреді.

Кілттік NOT, AND және OR сөздері логикалық операциялармен сәйкесінше белгіленеді: ЖӘНЕ, ЕМЕС және НЕМЕСЕ. Кілттік IF және THEN сөздері сәйкесінше "егер" және "онда" деп аталады. Кілттік EXISTS және FORALL сөздері ***кванторлар*** деп аталады. Алғашқылары – бар болу кванторы, ал екіншісі – жалпыламалық квантор. Осы кванторларды жан-жақты қарастырайық.

Wff формуласы: EXISTS x (f) мынаны білдіреді: "f формуласының мәні ақиқат болатындай x айнымалысының ең болмағанда бір мәні бар болады". FORALL x (f) өрнегі былай түсіндіріледі "x айнымалысының барлық мәндері үшін f есептеу формуласы мәні ақиқат". Жалпы жағдайда кортеж айнымалылары формулаларда *еркін* немесе *байланысқан* болуы мүмкін. EXISTS x (f) және FORALL x (f) формулаларда x кортеж айнымалылары әрқашан байланысқан болады.

Мысал 3. Өрнектер жазбасы.

Келесі сұранысқа сәйкес келетін өрнектер жазбасын қарастырайық: "барлық бөлшектерді жеткізетін жеткізушілер атын алу керек".

SX.Қала_П WHERE FORALL PX (EXISTS SPX (SPX.П# = SX.П# AND
SPX.Д# = SX.Д#))

Бұл келесі өрнекке барабар:

SX.Қала_П WHERE NOT EXISTS PX (NOT EXISTS SPX
(SPX.П# = SX.П# AND
SPX.Д# = SX.Д#))

Сипатталған есептеу толық емес, өйткені деректер қорында деректерді өңдеумен байланысты есептеулерді орындауға мүмкіндік бермейді. Бұл есептеулерге есептеу функцияларын кірістіру немесе қосу кеңейтулер арқылы жүзеге асады. Келесі функцияларды қорытынды ретінде қарастыруға болады: COUNT (саны), SUM (қосынды), AVG (орташа), MAX (ең үлкен), MIN (ең аз). Бүтін элементтер үшін келесі түрдегі өрнек "AS <атрибут атауы>", қорытынды атрибутқа атау беруге мүмкіндік береді.

Мысал 4. Сұраныс жазбасы.

Бөлшектер және жеткізілімнің жалпы салмағы туралы толық деректерімен әрбір жеткізілім жайлы ақпаратты алу талап етілсін. Қосымша функцияларды қамтитын сұраныс келесі түрде бейнеленуі мүмкін:

(SPX.П#, SPX.Саны, PX, PX.Салмағы * SPX.Саны AS Жалпы_Салмағы)
WHERE PX.Д# = SPX.Д#

Домендерге негізделген реляциялық есептеулер, (***Домендерді есептеу***) нұсқасын Лакроиксом және Пиротте (Lacroix and Pirotte) ұсынды, сонымен қатар олар осының негізінде сәйкес ILL тілін құрды. Домендерді есептеуге негізделген басқа тілдер: FQL, DEDUCE, сонымен қатар QBE.

Дейт пікірі бойынша, QBE тілі кортеждерді есептеу элементтерін және домендерді есептеуді іске қосады, бірақ екіншісіне неғұрлым жақын. Ол реляциялық толық емес болып табылады, өйткені квантордың бар болуын терістеу (NOT EXISTS) операциясын сүйемелдемейді. Бұл кемшілікке қарамастан, QBE тілі заманауи ДҚБЖ-ларда кеңінен таралды.

Домендерді есептеу кортеджерді есептеуге өте ұқсас. Кортеждерді есептеуден айырмашылығы домендерді есептеуде сұраныстың кез-келген өрнектері негізі ретінде домен айнымалылары қарастырылады. *Домен айнымалысы* – бұл скалярлық айнымалы, мәні қайсыбір доменнің элементтерін қамтиды.

Қарастырылатын есептеулердің айырмашылықтарының көбісі мынада: домендерді есептеу *тиістілік шарты* деп аталатын қосымша шартты сүйемелдейді. Тиістілік шартының жалпы түрде жазылуы:

R (A_1:ϑ_1, A_2:ϑ_2, ...),

мұндағы A_i – R қатынастар атрибуты , ал ϑ_i – домен айнымалысы немесе литерал. Егер A атрибуты бар R кортеж қатынастарында берілген өрнектер мәні сәйкесінше ϑ мәніне тең болса ғана тексерілетін шарт ақиқат болады.

Мысалы, егер SP қатынастарда ең болмағанда бір кортеж мәні П# атрибутта 'S1' және Д# атрибутта 'P1' болса, онда SP (П# : 'S1', Д# : 'P1') өрнегі ақиқат. Осыған сәйкес, егер SP қатынастарда П# атрибут мәні SX ағымдағы домен айнымалысы мәніне эквивалентті, ал Д# атрибут мәні X домен айнымалысының P ағымдағы мәніне эквивалентті болатын кортеж бар болса, онда SP (П# : SX, Д# : PX) өрнегі ақиқат.

Келесі мысалдарда SX (домен П#), PX (домен Д#), NAMEX (домен Аты) берілсін.

Мысал 5. Домендерді есептеу өрнектері.

(SX) WHERE S (П# : SX)

(SX) WHERE S (П# : SX, Қала_П : 'Москва')

NAMEX WHERE EXISTS SX (S (П# : SX, Аты : NAMEX)
AND FORALL PX (IF P (Д# : PX)
THEN SP (П# : SX, Д# : PX)))

Бірінші өрнек S қатынастарының барлық жеткізушілер номерлері жиынын білдіреді, екіншісі – Лондоннан жеткізушілер номерлері жиыны. Үшінші өрнек барлық бөлшектерді өндіруші жеткізушілер аттарына сұраныс жасауға сәйкес келеді.

3.8. QBE нұсқасы бойынша сұраныстар тілі

Қорда сақталатын деректерді қолмен өңдеуге болады, ол үшін ДҚБЖ-да бар сәйкес құралдардың көмегімен кестелердегі деректерді тізбектей қарастырамыз және редактрлейміз. Тиімділігін арттыру үшін деректерді жиынтық өңдеу мүмкіндігін беретін, яғни бір уақытта жазбалар жиынын енгізу, редактрлеу және өшіру, сонымен қатар кестеден деректерді таңдау орындайтын сұраныстар қолданады.

***Сұраныс* дегеніміз** деректер қорымен жүзеге асатын, сақталатын деректерді таңдау, өшіру немесе түрлендіру бойынша орындалатын операциялардың құрамын анықтайтын арнайы сипатталған талап. Әртүрлі ДҚБЖ-да сұраныс жасау үшін көбінесе сұраныстарды сипаттаудың негізгі екі тілі қолданылады:

- QBE тілі (Query By Example) – үлгі бойынша сұраныстар тілі;
- SQL (Structured Query Language) – құрылымдық сұраныстар тілі.

Деректермен іс-әрекеттер жасауды сипаттау кезінде аталған тілдердегі сұраныстар іс жүзінде барабар. Олардың арасындағы негізгі айырмашылықтар сұраныстарды қалыптастыру тәсілдерінде: QBE тілі сұранысты қолмен немесе визуальды қалыптастыруды жобалайды, ал SQL сұранысты бағдарламалауды білдіреді.

QBE тілінің сипаттамасы

Айнымалы-домендермен реляциялық есептеулердің теориялық негізі QBE тілі болып табылады. QBE тілі ДҚБЖ ұсынған сұраныс формаларын толтыру жолымен ДҚ-на күрделі сұраныстар жасауға мүмкіндік береді. Сұраныс жасаудың мұндай тәсілі жоғары көрнекілікті қамтамасыз етеді және операциялардың орындалу алгоритмін көрсетуді талап етпейді – күтілетін нәтижені сипаттау жеткілікті. Заманауи реляциялық ДҚБЖ әрқайсысының QBE тілінде өз нұсқасы бар.

QBE тілінде ***біркестелік*** және ***көпкестелік*** (бірнеше байланысқан кестеден деректерді таңдайтын немесе өңдейтін) сұраныстар жасауға болады.

QBE тілінде сұраныстар көмегімен келесі негізгі операцияларды орындауға болады:

- деректерді таңдау;
- деректермен есептеу жасау;
- жаңа жазбалар қосу;
- жазбаларды өшіру;
- деректермен іс-әрекеттер (өзгерістер) жасау.

Сұраныстың орындалу нәтижесі жаңа кесте болып табылады, ол ***жауап*** (алғашқы екі операцияда), немесе ***бастапқы кестенің жаңартылуы*** (қалған операциялар үшін) деп аталады.

Таңдау, кірістіру, өшіру және түрлендіру операциялары логикалық өрнектер көмегімен берілген шарттарға сәйкес шығарылады. Деректермен есептеулер жасау арифметикалық өрнектер көмегімен беріледі және жауап кестелерінде ***есептелетін*** деп аталатын жаңа өрістер тудырады.

Сұраныс формасы бастапқы кестелер өрістерінің аты және атауларымен сәйкес келетін кестелер түрінде болады. ДҚ-ның қолжетімді кесте аттарын білу үшін QBE тілінде кесте аттарына таңдау жасау сұранысы қалыптасқан. Бастапқы кестелер өрісі аттары үлгіге қолмен немесе автоматты түрде енгізілуі мүмкін. Екінші жағдайда бағана атауларын таңдау сұранысы қолданылады.

Заманауи ДҚБЖ-ларда, мысалы, Access және Visual FoxPro, QBE тілі көмегімен сұраныстар жасаудың көптеген әрекеттері визуальды түрде тышқан көмегімен атқарылады. Жеке жағдайда, сұранысты дайындау кезінде визуальды кестелерді байланыстыру тапсырманың элементтерімен емес, бір кестенің өрістерін басқа кесте өрістеріне тышқанмен "апару" көмегімен орындалады.

QBE алғашқы нұсқасы

Көбісінде заманауи ДҚБЖ-ның өз QBE нұсқасы бар, ол алғашқы QBE сипаттаудан сәл өзгеше. Мұны 1975-1977 ж.ж. Злуфф М.М. ұсынған. QBE алғашқы нұсқасы негізінде оның негізгі мүмкіндіктерін қарастырайық.

QBE тілінің құралдары мен мүмкіндіктерін көрсету үшін тауарлардың бірнеше түрін сататын фирмада қолданылатын және сауда саласына қатысты ДҚ кестелерін қолданамыз Деректер қоры келесі төрт кестені іске қосады:

- EMP (қызметкерлер): АТЫ-ЖӨНІ – қызметкердің фамилиясы және аты-жөнінің бірінші әріптері, ЖАЛАҚЫСЫ – лауазымдық қызметақы мөлшері, ЖЕТЕКШІ – жетекшінің фамилиясы және аты-жөнінің бірінші әріптері, БӨЛІМІ – қызметкер жұмыс жасайтын бөлім атауы;
- SALES (сауда): БӨЛІМІ – бөлімі атауы, ТАУАР – тауар атауы;
- SUPPLY (жеткізушілер): ТАУАР – жеткізілетін тауар атауы, ЖЕТКІЗУШІ – тауарды жеткізуді ұйымдастырушы атауы;
- TYPE (тауар түрлері): ТАУАР – тауар атауы, ТҮСІ – оның түсі, ҚҰНЫ – тауар құны.

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
	Киселев В.М.	1800	Белкин Б.Н.	шаруашылық тауарлары
	Гурский С.И.	1600	Томилов А.Н.	ойыншықтар
	Андреева Е.А.	2000	Петров А.С.	косметика
	Левин П.Г.	2200	Петров А.С.	кеңсе тауарлары
	Носов А.П.	1600	Томилов А.Н.	ойыншықтар
	Гофман В.Э.	2600	Андреева Е.А.	косметика
	Сорокина Т.В.	1700	Андреева Е.А.	косметика
	Белкин Б.Н.	1800	Петров А.С.	шаруашылық тауарлары
	Семин С.В.	2200	Левин П.Г.	кеңсе тауарлары
	Григорьев А.Н.	1900	Томилов А.Н.	ойыншықтар
	Томилов А.Н.	2000	Петров А.С.	ойыншықтар

SALES	БӨЛІМІ	ТАУАР
	кеңсе тауарлары	қағаз
	шаруашылық тауарлары	сабын
	кеңсе тауарлары	қарындаш
	косметика	ерін далабы
	ойыншықтар	ұшақ

	ойыншықтар	машина
	ойыншықтар	қуыршақ
	косметика	әтір
	кеңсе тауарлары	сия
	шаруашылық тауарлары	ыдыс
	кеңсе тауарлары	қалам

SUPPLY	ТАУАР	ЖЕТКІЗУШІ
	қалам	Pencraft
	қағаз	Pencraft
	сабын	Procter & Gamble
	қарындаш	Flic
	сия	Pencraft
	әтір	Beautex
	сия	Flic
	ыдыс	Cremco
	ерін далабы	Beautex
	ұшақ	Signal
	машина	Signal
	қуыршақ	Signal
	ыдыс	Flic
	қалам	Beautex
	қарындаш	Pencraft

TYPE	ТАУАР	ТҮСІ	ҚҰНЫ
	ыдыс	ақ	30
	ерін далабы	қызыл	17
	әтір		42
	қалам	жасыл	6
	қарындаш	көк	2
	сия	жасыл	4
	сия	көк	3
	қарындаш	қызыл	2
	қарындаш	көк	2

Кестелерде келтірілгендер толық емес және қысқартылған мәліметтер. Ендеше, TYPE кестеде SALES кестесінде келтірілген тауарлардың барлық түрі көрсетілмеген.

Жазбаларды іріктеу шартында айнымалыларды сипаттау үшін, сонымен қатар сұраныстағы үлгілерді байланыстыру үшін мысал элементтері қолданылады.

Мысал элементі айнымалы идентификаторы ролін атқарады (бағдарламалау тіліндегідей) және символдық-цифрлық тізбек көмегімен беріледі. Мысал элементтерін үлгілерде сызып көрсетейік. Мысал элементі түрі (ұзындығы және құрамы) роль атқармайды: негізгі бірнеше үлгіні қолданған кезде олар бірдей болуы керек. Осылайша, мысал элементтері ретінде, жеке жағдайда, example, x немесе y идентификаторларын қолдануға болады.

Жүйелерде қорытынды кестенің қайсыбір өрісіне “P.” қосу “баспаға шығару” қажеттілігін білдіреді.

Мысал 1. Таңдауға сұраныс.

Жоғарыда айтылғандардан барлық жасыл тауарларды таңдауға сұраныс келесі түрде жазылады:

TYPE	ТАУАР	TYCI	ҚҰНЫ
	Р. XX	жасыл	

Сөз жүзінде сұранысты келесі түрде тұжырымдауға болады: “түсі жасыл барлық XX тауарлар шығару”. Келтірілген үлгіде мысал элементі XX міндетті емес және оны қалдырып кетуге болады. Мысал элементтері жазу кезінде, сонымен қатар сұраныстарда кестелерді байланыстыру кезінде міндетті түрде логикалық шарттармен көрсетіледі.

Бос бағаналарды үлгіден өшіруге болады.

Мысал 2. Бағаналарды өшіру.

Бізде қолданылмайтын бағана ҚҰНЫ болып табылады. Келтірілген үлгіге барабар келесі үлгіні жазуға болады:

TYPE	ТАУАР	TYCI
	Р.	жасыл

Үлгіні толтырғаннан кейін нәтиже алу үшін сәйкес перне басылады, мысалы <Enter>, және сұранысты орындау басталады. Келтірілген сұраныстың орындалу нәтижесі келесі кесте болып табылады:

TYPE	ТАУАР
	қалам
	сия

Сұраныстарда қолданылатын операциялардың жоғарыда аталған түрлерінің негізгі мүмкіндіктерін қарастырайық.

Деректерді таңдау

Қарапайым таңдау. Қарапайым таңдау мысалы келесі сұраныс болып табылады: “TYPE кестесінен тауарлардың барлық мүмкін түстерін шығару”.

Мысал 3. Қарапайым таңдау.

Толтырылған үлгі бұл жағдайда келесі түрде жазылады:

TYPE	ТАУАР	TYCI	ҚҰНЫ
		Р.	

Жауап кестесінде бір ғана TYCI бағанасы бар, оның мәндері: ақ, қызыл, бос (мәні берілмеген), жасыл және көк. Қосарланушы мәндер жойылады.

Егер бастапқы кестелердің бірнеше өрісінен деректер шығару талап етілсе, әрбір сәйкес бағаналарға “Р.” жазылады. Үлгінің барлық бағаналарына “Р.” жазуды кестелер атауындағы бірінші бағанаға “Р.” жазумен алмастыруға болады.

Реттелген қарапайым таңдау. Шығарылатын мәндерді өсуі және кемуі бойынша реттеу үшін сәйкесінше “AO.” және “DO.” қолданылады. Егер талап етілсе бірнеше бағаналар бойынша реттеу орындайды: “AO(1).”, “AO(2).”.

Квалификатормен таңдау (шартпен). Бастапқы кестелерден жазбаларды таңдау жалпы жағдайда мыналарға негізделген болуы мүмкін: тура келу, ішінара сәйкес келу, салыстыру.

1. *Тура келу* сәйкес үлгі өрістеріне тұрақтыны енгізумен беріледі, жасыл түсті тауар бойынша сұраныс жағдайы сияқты.

2. *Ішінара сәйкес келу* мысал элементтері көмегімен беріледі. Жеке жағдайда, аттары “о” әрпінен басталатын тауарлардың барлық түрін шығару сұранысын құру үшін TYPE кестесінде ТАУАР өрісінде жазылған “P.oke” сөз тіркесін қолдануға болады. Мұнда: “P.” баспаға шығаруды қамтамасыз етеді, “о” – тұрақты, ал “ke” – айнымалы ролін атқаратын мысал элементі.

Мысал элементтерін пайдаланып, кестеден деректер мәндерінің ішінара сәйкес келуінің әртүрлі нұсқаларын беруге болады: басында “ike”, соңында “xa”, ортасында “x1ox2” және ерікті орында.

Себебі мысал элементін кез-келген символмен, сонымен қатар бос жағдайымен (символ болмауы) салыстырсақ, онда ішінара сәйкес келу шарты “x1ox2” “о” символы ортасында тұрған сөзді ғана емес, “о” символымен басталатын және аяқталатын сөздерді қанағаттандырады.

Мысал 4. Ішінара сәйкес келу.

Түсі көк, тауар атауы ортасында “р” әрпі бар таңдауға сұраныс үлгісі төмендегідей:

TYPE	ТАУАР	ТҮСІ	ҚҰНЫ
	P.xpy	көк	

Бұл жағдайда нәтиже келес түрде:

TYPE	ТАУАР
	қарындаш

3. *Салыстыру шарты* салыстыру операциялары көмегімен жазылады: тең (=), үлкен (>), аз (<), үлкен немесе тең (>=), кіші немесе тең (<=), тең емес (¬= немесе ¬), үлкен емес (¬>), кіші емес (¬<).

Мысал 5. Салыстыру шарты.

Ойыншықтар бөлімінде жұмыс жасайтын, жалақысы 1800 үлкен қызметкерлер атына сұраныс төмендегідей жазылады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
	P.	>1800		ойыншықтар

Сұраныстың нәтижесі келесі түрдегі кесте болып табылады:

EMP	АТЫ-ЖӨНІ
	Григорьев А.Н.
	Томилов А.Н.

Салыстыру операцияларында мысал элементтерін қолдануға болады.

Мысал 6. Салыстыру элементтеріне мысалы.

Сұраныс үлгісінің төменде келтірілген түрі - Левин П.Г. жалақысынан көп жалақы алатын қызметкерлердің атын таңдау. Сұранысты басқаша былай жазуға болады: “Левин П.Г. s мөлшерде жалақы алсын. Жалақысы s үлкен барлық қызметкерлерді тауып, олардың жалақысын баспаға шығару керек”.

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ
	P.	P.> s
	Левин П.Г.	s

EMP кестеде жалақысы Левина П.Г. үлкен қызметкерлер жоқ (мүмкін ол Петров А.С. болар, бірақ EMP кестеден бұны біле алмаймыз). Бұл жағдайда нәтижелік кесте бос болады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ

Сұраныстағы шарт бір немесе бірнеше бағаналар бойынша берілуі мүмкін. Сонымен қатар жекелеген шарттарды біріктіру логикалық және (AND) түрінде жүзеге асады.

Мысал 7. Шарттарды біріктіру.

Левин П.Г. жалақысынан көп жалақы алатын және ойыншықтар бөлімінде жұмыс жасайтын қызметкерлердің атын және жалақысын таңдау сұранысын құру үшін алғашқы сұраныстағы БӨЛІМІ бағанасының бірінші жолына "ойыншықтар" сөзін жазу жеткілікті.

Сұраныс түрі: "Белкин Б.Н. жалақысынан көп жалақы алатын және қалам сату бөлімінде жұмыс жасайтын қызметкерлердің атын және жалақысын таңдау" келесі түрде жазылады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
	P.	P.> s		department
	Белкин Б.Н.	s		

SALES	БӨЛІМІ	ТАУАР
	department	қалам

Бұл сұраныстың орындалу нәтижесі кесте түрінде:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ
	Левин П.Г.	2200
	Семин С.В.	2200

Мұнда мысал элементі department екі бастапқы кестелер БӨЛІМІ өрісі бойынша қарастырылған, ал мысал элементі s таңдау үшін қолданылады.

Мысал 8. Үлгісінде екі байланысы бар сұраныс.

Өз жетекшілерінен үлкен жалақы алатын барлық қызметкерлерді табу керек болсын. Бұл сұраныс көмегімен мысал элементтерінде былай жазуға болады: "жетекшілері head және s шамасынан үлкен жалақы алатын барлық қызметкерлерді шығару, мұндағы s – head жалақысы". Сәйкес сұраныстың үлгісі келесі түрде жазылады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
	P.	> s	head	
	head	s		

Мұнда head элементі үлгінің бірінші жолындағы жетекшінің байланыстары және екінші жолдағы аттары үшін, ал s элементі жалақыларды салыстыру үшін қолданылады.

Бұл сұраныстың орындалу нәтижесі кестесі:

EMP	АТЫ-ЖӨНІ
	Гофман В.Э.

Кестеден жазбаларды таңдау шарты ретінде терістеу операциясын қолдануға болады.

Мысал 9. Терістеу операциясымен таңдау.

Pencraft компаниясы жеткізбейтін тауарларды сататын барлық бөлімдерді шығару керек болсын. Бұл сұранысты былай жазуға болады: “Pencraft компаниясы жеткізбейтін t тауарларды сататын бөлім аттарын шығару”.

SALES	БӨЛІМІ	ТАУАР
	P.	t

SUPPLY	ТАУАР	ЖЕТКІЗУШІ
	¬ t	Pencraft

Бұл сұраныстың орындалу нәтижесі кестесі:

SALES	БӨЛІМІ
	шаруашылық тауарлары
	косметика
	ойыншықтар

Жазбаларды таңдау шарты үлгіде көрсетуге қиын және тиімсіз үлкен өрнектер болған жағдайда үлгіде ***шарттар блогын*** қолдануға болады. Оның түрі бос үлгіге ұқсас. Шарттар блогы логикалық өрнектерді жазуға арналған.

Бір үлгіде жазылған логикалық өрнектер, жалпы жағдайда логикалық көбейту (AND операциясы) және логикалық қосу (OR операциясы) операцияларын қамти алады.

Мысал 10. Шарттар блогын қолдану.

Жалақысы 2000 мен 2500 арасында, бірақ 2300 шамасына тең емес барлық қызметкерлердің атын шығару келесі түрде жазылады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ
	P. Jon	>2000
	Jon	<2500
	Jon	¬2300

Шарттар блогын пайдаланып, AND (символ &) операциясы арқылы осы сұранысты қарапайым түрде жазуға болады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ
	P.	s

CONDITIONS
s = (>2000 & <2500 & ¬2300)

Жалақысы 2000, 2300 немесе 2600 болатын барлық қызметкерлердің атын шығару сұранысы үлгісі келесі түрде жазылады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ
	P. Jon	2000
	P. Mak	2300
	P. Nik	2600

Үлгінің әрбір жолына әртүрлі мысал элементтері қолданылады, және сондықтан шарт тәуелсіз. OR (символ |) операциясы арқылы шарттар блогы көмегімен осы сұранысты неғұрлым көрнекі жазуға болады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ
	P.	s

CONDITIONS
s = (2000 | 2300 | 2600)

QBE тілінде логикалық өрнектерді жазу кезінде ***кіріктірме функциялар*** қолданылады: CNT. (санауыш немесе саны), SUM. (қосынды), AVG. (орташа), MIN. (минимум), MAX. (максимум), UN. (ерекше) және ALL. (барлық мәні, сонымен қатар қайталанушы мәндер). Алдыңғы бесеуі статистикалық болады, ал соңғы екеуі таңдау өзгешелігін анықтайды: таңдауға қайталанушы мәнді қосады немесе өшіреді.

UN. функциясын CNT., SUM. және AVG. функцияларына қосуға болады. Ендеше, CNT.UN. жазбасы тек айырмашылығы бар мәндердің санын білдіреді. Бұған қарама-қайшы, CNT.ALL. жазбасы барлық мәндер санын білдіреді. Әлбетте, MAX.UN. және MAX.ALL. функциялары бірдей нәтиже береді.

Мысал 11. Функцияларды қолдану.

Екіден көп қызметкер жұмыс жасайтын бөлім аттарын шығару талап етілсін. Бұл сұранысты үш операцияға бөлуге болады: қызметкерлерді бөлім бойынша топтау, әрбір бөлімдердегі қызметкерлер санын санау және екіден көп қызметкер жұмыс жасайтын бөлім аттарын шығару.

EMP	АТЫ-ЖӨНІ	БӨЛІМІ
	ALL.Employee	P. **Toy**

CONDITIONS
CNT.ALL.Employee > 2

Toy құрылымы ***montay*** (group-by) операциясын білдіреді, ALL.Employee функциясы әрбір бөлім бойынша барлық атаулар (нақтырақ айтқанда мультижиынды, ол мәннің қайталануына рұқсат береді) жиынын құрайды, ал CNT.ALL.Employee > 2 жазбасы логикалық шартты тексеруді қамтамасыз етеді. Мұнда топтар құру жаңа операциясы кездеседі, ол қалың қаріппен асты сызылып жазылады (түпнұсқада – мысал элементі аты екі рет сызылады).

Қорытынды кесте түрі:

EMP	БӨЛІМІ
	ойыншықтар
	косметика

Келесі мысал. Тек қана жасыл түсті тауарларды жөнелтетін бөлім аттарын шығару.

SALES	БӨЛІМІ	ТАУАР
	P. **Toy**	ALL.Ink

TYPE	ТАУАР	ТҮСІ
	ALL.Ink	жасыл

Берілген сұраныс үшін жауап кестесі бос болады.

Сұраныстағы есептеу

Сұраныстар көмегімен кестеден деректерді таңдауға және есептеуді шығаруға болады. Есептеу түрі үлгідегі өрнектер көмегімен беріледі. Өрнектерде, қарапайым арифметикалық операциялар (+, -, *, /) және жақшалармен қатар, кіріктірме функциялар: AVG., CNT., MAX., MIN. және SUM. қолданылады.

Мысал 12. Есептеулермен үлгілер.

Қызметкерлердің аты (АТЫ-ЖӨНІ), жалақы мөлшері (ЖАЛАҚЫ) және сыйақы мөлшері (СЫЙАҚЫ) өрістері бар EMP1 кестесі берілсін. Әрбір қызметкер бойынша қызметкерлердің атын, жалақы мөлшерін және сыйақы мөлшерін баспаға шығару керек. Осы үшін OUTPUT кестесінде үлгі құрамыз (кестелер аты жолын және оның өрісі аттарын толтырып) және онда есептеу түрілерін көрсетейік. Бұл үлгіні EMP1 кестесінің сұраныс үлгісімен байланыстырайық.

OUTPUT	АТЫ-ЖӨНІ	ҚОСЫНДЫ
	P. Employee	P. (s1+s2)

EMP1	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	СЫЙАҚЫ
	Employee	s1	s2

Себебі қосындыны табу операциясы бастапқы кестелердің әрбір жолы бойынша орындалады, есептеулердің бұл түрін *горизонтальды есептеу* деп атауға болады. Кіріктірме функциялар жазбалар тобымен жұмыс жасайды, сондықтан *вертикальды есептеу* жасауға болады.

Қызметкерлердің жалпы санын санау үшін келесі түрдегі сұраныс құрамыз:

EMP	АТЫ-ЖӨНІ	БӨЛІМІ
	P.CNT.ALL.Em	

Em мысал элементін мұнда ескермеуге болады.

Егер ойыншықтар бөліміндегі қызметкерлер санын санау талап етілсе, онда келесі түрдегі сұраныс құрамыз:

EMP	АТЫ-ЖӨНІ	БӨЛІМІ
	P.CNT.ALL.Em	ойыншықтар

Егер әрбір бөлімдердегі (бөлім атауларын шығара отырып) қызметкерлер санын санау талап етілсе, онда келесі түрдегі сұраныс құрамыз:

EMP	АТЫ-ЖӨНІ	БӨЛІМІ

	P.CNT.ALL. Em	P. **Dep**

Бұл жағдайда келесі нәтиже аламыз:

EMP	АТЫ-ЖӨНІ	БӨЛІМІ
	2	шаруашылық тауарлары
	3	ойыншықтар
	3	косметика
	2	кеңсе тауарлары

Үлгідегі сұранысқа біз топтау операциясын қолдандық (Мысал 11 қараңыз).

Кірістіру, өшіру және түрлендіру операциялары

Қарастырылған операциялардан кірістіру, өшіру және түрлендіру операцияларының айырмашылығы олар бастапқы кестелерді *өзгертеді*. Операциялар түрі (кірістіру – I., өшіру – D., түрлендіру – U.) үлгідегі кестелер атымен жазылады, ал тұрақтылар және шартты өрнектер таңдау операциялары ережелері бойынша көрсетіледі.

Мысал 13. Кірістіру, өшіру, түрлендіру операциялары.

EMP кестесінің ойыншықтар бөліміне жалақысы 2000 фамилиясы Деревянко Н.В., жетекшісі Белкин Б.Н. жаңа қызметкерді кірістіру үшін келесі түрдегі үлгі құрамыз:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
I.	Деревянко Н.В.	2000	Белкин Б.Н.	ойыншықтар

Ойыншықтар бөлімінің қызметкерлері туралы барлық ақпаратты өшіру керек болсын. Сәйкес сұраныс үлгісі келесі түрде жазылады:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
D.				ойыншықтар

Бір өрістің қайсыбір мәнін, мысалы Белкин Б.Н. қызметкердің жалақы мөлшерін түрлендіру үшін келесі үлгі құрамыз:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
U.	Белкин Б.Н.	2100		

Бос өріс оның өзгеріске ұшырамайтындығын білдіреді. Егер қайсыбір мәнді “бос мәнге” өзгерту талап етілсе, онда NULL кілттік сөзі қолданылады.

Ойыншықтар бөліміндегі қызметкерлер жалақысын 10% өсіру керек болса, онда келесі түрдегі үлгі құрамыз:

EMP	АТЫ-ЖӨНІ	ЖАЛАҚЫСЫ	ЖЕТЕКШІ	БӨЛІМІ
U.		1.1*s1		
		s1		ойыншықтар

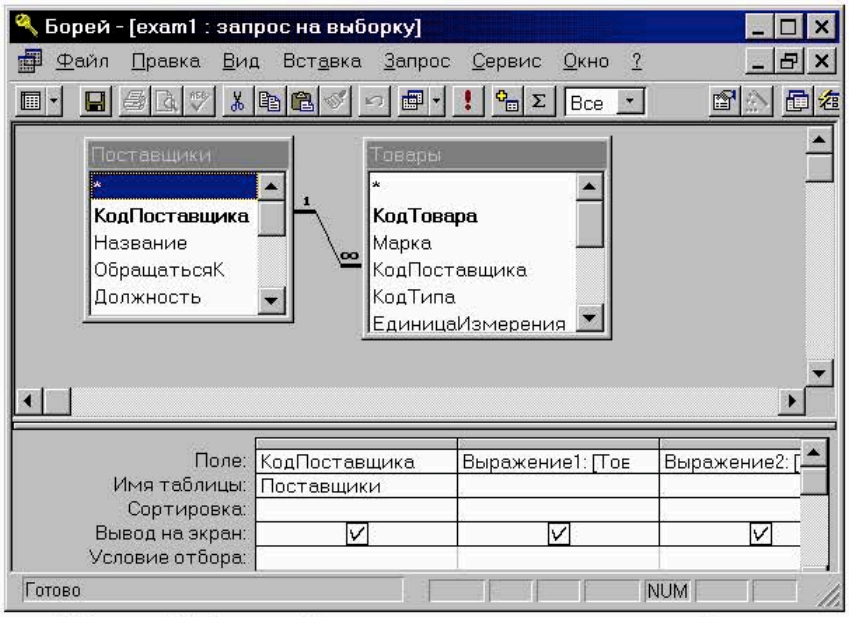
Бұл сұранысты екі кезеңмен жүзеге асырамыз: алдымен БӨЛІМІ өрісіндегі мәні “ойыншықтар” болатын барлық жазбаны таңдайды, содан соң таңдалғандардың ЖАЛАҚЫ өрістеріне өзгерістер енгізеді.

Заманауи ДҚБЖ QBE тілдерінің сипаттамасы

Заманауи ДҚБЖ QBE тілдерінің Злуфф М.М. ұсынған тілден айырмашылығы, қағидаға сәйкес, жекелеген реляциялық операцияларды талдауларға кішігірім өзгерістер енгізу, қосымша операциялар жасау және формалар көрінісін өзгерту болып табылады.

Мысалы, Paradox for Windows жүйелерінде басып шығару P. операциясы орнына өрістің сұраныс үлгісінде таңдалғандарды белгілеу әдісі қолданылған. Осы үшін әрбір өрістерде сұраныс формалары орналасады. Пайдаланушы өрістерді белгілей отырып, жауап кестесінде сұрыптау тізбегін көрсетеді. Бірнеше сұраныс формаларын бір көпкестелік сұранысқа байланыстыру үшін, сонымен қатар логикалық өрнектерде жазбаларды таңдау шарты *мысал элементтерін* қолданылады.

Microsoft Access сұраныс формалары неғұрлым көрнекі. Сұқбаттасу терезесі (3.9-сурет) сұраныс формаларын дайындау кезінде екі бөліктен тұрады: үстіңгі бөлімде бастапқы кесте өзара байланыстары үлгісі бойынша ақпараттар, ал төменде – әрбір өрістерден сұраныс туралы (мәндерді шығару қажеттілігі, сұрыптау түрі, таңдау шарты және т.б.) қалған ақпарат.



3.9-сурет. Microsoft Access бағдарламасында сұраныс формалары түрі

Сұраныс үлгісін дайындау пайдаланушымен тышқан көмегімен орындалады. Ендеше, сұраныста кестелерді байланыстыру мысал элементтерімен емес, бір кестенің өрістерін басқа кестелердің өрістерімен "жетекке алу" арқылы жүзеге асады. Егер кестелер арасында өзара байланыс болса, онда жүйелер автоматты түрде кестелердің сұраныс формасындағы барлық ақпараттарды байланыстырады. Сонымен қатар әрбір байланыс түріне сейкес белгіленеді (3.9-суретте Жеткізушілер және Тауарлар өрісі кестелері арасында КодЖеткізуші бойынша 1:М түріндегі байланыс құрылған).

Заманауи ДҚБЖ талдау QBE тілінің келесі бағыттарының дамуына мүмкіндік береді:

- Көрнекілігін және тиімділігін арттыру.
- ДҚБЖ жаңа мүмкіндіктеріне сәйкес құралдардың пайда болуы, мысалы, нақты емес немесе айқын емес сұраныстар құру, үлкен көлемді деректермен айла-амалдар жасау.
- деректердің жаңа түрлерін (графикалық, аудио-, видео- және басқалар) қолдану.
- Болашақта табиғи тілмен шектелген сұраныстар құру.
- Келешекте сөйлеу арқылы сұраныстар жасау.

Табиғи тілмен шектелген сұраныстар құру және сөйлеу арқылы сұраныстар жасау мүмкіндіктері қазір де бар. Бұл ДҚБЖ терминдерді сәйкестендіру сөздігін қолданған кезде жүзеге асады.

3.9. SQL құрылымдамалық сұраныстар тілі

SQL құрылымдық сұраныстар тілі *кортеж айнымалыларымен* реляциялық есептеулер жүргізуге негізделген. Аталған тілдің бірнеше стандарттары бар, неғұрлым кеңінен таралғаны SQL-89 және SQL-92 (9.3-бөлім) болады.

Тілдің жалпы сипаттамасы

SQL тілі кестелермен операциялар (құру, өшіру, өзгерістер құрылымы) және кесте деректерімен операциялар (таңдау, өзгерістер, кірістіру немесе қосу және өшіру), сонымен қатар басқа да операциялар орындауға арналған. SQL *процедуралық емес* тіл болып табылады және басқару, қосалқы бағдарламаларды ұйымдастыру, енгізу-шығару және т.с.с. операторларды қамтымайды. Байланыстарда SQL дербес қолданылмайды, әдетте ол ДҚБЖ бағдарламалау тілі ортасына енгізілген (мысалы, FoxPro ДҚБЖ Visual FoxPro, ObjectPAL ДҚБЖ Paradox, Visual Basic for Applications ДҚБЖ Access).

Заманауи, интерактивті интерфейсті ДҚБЖ-ларда басқа құралдарды, мысалы QBE пайдаланып сұраныстар құруға болады. Бірақ SQL қолдану көбінесе қордағы деректерді өңдеу тиімділігін арттыруға мүмкіндік береді. Мысалы, Access ортасында сұранысты дайындау кезінде сұраныстар конструкторы терезесінен SQL операторымен барабар терезеге өтуге болады. Осыған дейінгі қарастырылып жүрген жағдайларда өңдеу жолымен жаңа сұранысты дайындау SQL операторына өзгерістер енгізу жолымен оңай орындалады. Әртүрлі ДҚБЖ-да SQL операторлар құрамы өзгеше болуы мүмкін.

SQL тілі өңдеудің толыққанды функцияларын қамтымайды, ол деректерге қатынас құруға негізделген, сондықтан ол бағдарламаларды өңдеу құралдарын қамтиды. Бұл жағдайда ол *кіріктірілген SQL* деп аталады. SQL тілі стандарты келесі бағдарламалау тілдерінің заманауи жүзеге асуын сүйемелдейді: PL/1, Ada, C, COBOL, Fortran, MUMPS және Pascal.

Мамандандырылған жүйелерде клиент-сервер түріндегі қосымшаны өңдеу бағдарламалау ортасы құралдарымен, сонымен қатар, коммуникациялық құралдармен (ДҚ серверлерімен байланыс орнату және ажырату, желідегі қателіктерді табу және өңдеу және т.б.), пайдаланушы интерфейсін өңдеу құралдарымен, жобалау құралдарымен жүзеге асады.

Кіріктірілген SQL пайдаланудың екі негізгі әдісі бар: статикалық және динамикалық.

Статикалық (***статикалық SQL***) қолдануда бағдарламалар мәтінінде SQL тілі функциялары шақырылады, олар компиляциядан соң орындалатын модульды қамтиды. Шақырылатын функцияларда өзгерістер енгізу бағдарламалау тілі айнымалылары көмегімен жекелеген параметрлерді шақыру деңгейінде болуы мүмкін.

Динамикалық (динамикалық SQL) қолданыста SQL-функцияларды шақырудың динамикалық тұрғызылымын жобалайды және талдайды, мысалы, шалғай қорлардағы деректерге қатынас құру, бағдарламалардың орындалуы кезінде. Динамикалық әдіс әдетте, қосымшалардағы SQL-шақыру түрі алдын-ала белгісіз және ол пайдаланушымен сұқбат кезінде тұрғызылатын жағдайда қолданылады.

SQL тілінің (деректер қорымен жұмыс жасайтын басқа тілдер сияқты) негізгі тағайындалуы сұраныстарды дайындау және орындау болып табылады. Бір немесе бірнеше кестеден деректерді таңдау нәтижесі *көрініс* деп аталатын жазбалар жиыны болуы мүмкін.

Көрініс табиғаты бойынша сұраныстың орындалуы нәтижесінде қалыптасқан кесте болып табылады. Ол сақталатын сұраныстың бір түрі деуге болады. Бір кестеге бірнеше көрініс тұрғызуға болады.

Ыңғайлылық үшін SQL тілінде көріністермен жұмыс курсор түсінігімен енгізілген. ***Курсор*** жазбаларды өңдеу кезінде жазбалар жинағы бойынша орын ауыстыру үшін қолданылатын өзіндік сілтеуіш болып саналады.

SQL тілінде курсорды сипаттау және қолдану келесі түрде орындалады. Бағдарламалардың сипаттамалар бөлімінде курсор (CURSOR) типті айнымалыны SQL (әдетте SELECT операторымен) операторымен байланыстыру орындалады. Бағдарламаның орындалатын бөлімдерінде курсорды ашу (OPEN <курсор аты>), жазбалар бойынша курсорды жылжыту (FETCH <курсор аты>...), және сонымен қатар, курсорды жабу (CLOSE <курсор аты>) жүзеге асады.

Тілдің негізгі операторлары

SQL тілін оның Microsoft фирмасының ODBC (Open Database Connectivity – үйлесімді ашық деректер қоры) стандартты интерфейсінде жүзеге асуына сүйене отырып сипаттайық.

SQL тілі операторларын шартты түрде екі ішкі тілге бөлуге болады: деректерді анықтау тілі (Data Definition Language – DDL) және деректермен іс-әрекеттер жасау тілі (Data Manipulation Language – DML). Негізгі SQL тілі операторлары 3.3-кестеде келтірілген.

3.3-кесте. SQL тілінің операторлары

Түрі	Атауы	Тағайындалуы
DDL	CREATE TABLE	кестелер құру
	DROP TABLE ALTER TABLE	кестелерді өшіру
	CREATE INDEX	кестелердегі өзгерістер құрылымы
	DROP INDEX	индекс құру
	CREATE VIEW	индексті өшіру
	DROP VIEW	көрініс құру
	GRAND*	көріністі өшіру
	REVOKE*	артықшылықтар тағайындау
		артықшылықтарды жою
DML	SELECT	жазбаларды таңдау
	UPDATE	жазбалардағы өзгерістер
	INSERT	жаңа жазбалар кірістіру
	DELETE	жазбаларды өшіру

Кестеде “*” символымен белгіленген маңызды операторлардың форматын және негізгі мүмкіндіктерін, арнайы опрацияларды есептемей, қарастырайық. Мардымсыз операндалар және синтаксис элементтерін (мысалы, көптеген бағдарламалау жүйелерде оператор соңына “;” белгісі қойылады) ескермейміз.

1. ***Кестелер құру*** операторы келесі түрде жазылады:

CREATE TABLE <кестелер аты>

(<бағана аты> <деректердің типі> [NOT NULL]

[,<бағана аты> <деректердің типі> [NOT NULL]] ...)

Құрылатын кестелер аты және осы бағанада сақталатын деректердің типін көрсететін ең болмағанда бір бағананың аты міндетті оператор операндалары болады.

Кестелер құрған кезде жекелеген өріс үшін енгізілетін мәндерді бақылаудың қай-сыбір қосымша ережесі көрсетілуі мүмкін. NOT NULL (бос емес) құрылымы кестенің бұл бағанасының мәні міндетті түрде болуы қажеттігін білдіреді.

Жалпы жағдайда әртүрлі ДҚБЖ-да әртүрлі деректер типі (1 бөлімін қараңыз) қолданылады. ODBC интерфейсінде өзінің стандартты деректер типін сүйемелдейді, мысалы, символдық (SQL_CHAR, SQL_VARCHAR, SQL_LONGVARCHAR) және басқалар.

Мысал 1. Кестелер құру.

Тауарларды сипаттайтын goods кестесін құру талап етілсін. Ол кестеде төмендегідей өрістер болсын: type – тауар түрі, comp_id – өндіруші компанияның идентификаторы, name – тауар атауы және price – тауар бағасы. Кестелерді анықтау операторы келесі түрде болуы мүмкін:

CREATE TABLE goods (type SQL_CHAR(8) NOT NULL, comp_id SQL_CHAR(10) NOT NULL, name SQL_VARCHAR(20), price SQL_DECIMAL(8,2)).

2. ***Кестелер құрылымына өзгерістер енгізу*** операторы келесі түрде жазылады:

ALTER TABLE <кестелер аты>
({ADD, MODIFY, DROP} <бағана аты> [<деректердің типі>]
[NOT NULL]
[,{ADD, MODIFY, DROP} <бағана аты> [<деректердің типі>]
[NOT NULL]] ...)

Кестелердегі өзгерістер құрылымы мыналарды қамтиды: бір немесе бірнеше кесте бағаналарын қосу (ADD), өзгерту (MODIFY) немесе өшіру (DROP). ALTER TABLE операторының жазу ережесі CREATE TABLE операторы сияқты. Бағананы өшіру кезінде <деректердің типі> көрсету қажет емес.

Мысал 2. Кестелерге өрістер кірістіру немесе қосу.

Алдыңғы құрылған goods кестесіне тауар қорының мөлшері туралы ақпарат сақтау үшін number өрісін кірістіру (қосу) қажет болсын. Осыны жүзеге асыру үшін келесі түрдегі оператор қолданамыз:

ALTER TABLE goods (ADD number SQL_INTEGER).

3. ***Кестелерді өшіру*** операторы келесі түрде жазылады:

DROP TABLE <кестелер аты>

Оператор құрылған кестені өшіруге мүмкіндік береді. Мысалы, items атаулы кестелерді өшіру үшін келесі түрдегі операторды жазу жеткілікті:

DROP TABLE items.

4. ***Индекс құру*** операторы келесі түрде жазылады:

CREATE [UNIQUE] INDEX <индекс аты>
ON <кестелер аты>
(<бағана аты> [ASC | DESC]
[,<бағана аты> [ASC | DESC] ...)

Оператор кестемен орындалатын сұраныс және іздеу операцияларының орындалуын жылдамдату үшін берілген кестелердің бір немесе бірнеше бағаналарына индекс құруға мүмкіндік береді. Міндетті емес UNIQUE опциясын көрсетіп, бағаналардағы операторларда көрсетілген барлық мәндердің бірегейлігін қамтамасыз етуге болады. Табиғаты бойынша, UNIQUE көрсетіп индекс құру алдыңғы құрылған кесте кілтін анықтауды білдіреді.

Индекс құрған кезде бағаналардағы мәндерді автоматты түрде сұрыптау ретін көрсетуге болады – өсу реті бойынша ASC (үнсіз келісім бойынша), немесе кему бойынша DESC. Әртүрлі бағаналар үшін әдетте әртүрлі сұрыптау реті таңдалады.

Мысал 3. Индекс құру.

EMP кестелері берілсін, бұл кестенің келесі түрдегі өрістері бар болсын: NAME (аты), SAL (жалақы), MGR (жетекші) және DEPT (бөлімі). Аттарын алфавит бойынша және жалақы көлемін кемуі бойынша сұрыптау үшін main_indx индекс құру керек. Индекс құру операторы келесі түрде жазылады:

```
CREATE INDEX main_indx
    ON emp (name, sal DESC).
```

5. ***Индексті өшіру*** операторы келесі түрде жазылады:

DROP INDEX <индекс аты>

Бұл оператор бұрын құрылған атауы сәйкес индексті өшіруге мүмкіндік береді. Ендеше, мысалы, main_indx индексін өшіру үшін emp кестесіне DROP INDEX main_indx операторын жазу жеткілікті.

6. ***Көрініс құру*** операторы келесі түрде жазылады:

```
CREATE VIEW <көрініс аты>
        [(<бағана аты> [,<бағана аты> ] ... )]
        AS <SELECT операторы>
```

Берілген оператор көрініс құруға мүмкіндік береді. Егер бағаналар аты көріністе көрсетілмесе, онда SELECT операторымен сипатталған сұраныстағы бағаналар аты қолданылады.

Мысал 4. Көрініс құру.

Шығарылатын тауарларды сипаттайтын companies кестесінің келесі түрдегі өрістері бар болсын: comp_id (компания идентификаторы), comp_name (компания атауы), comp_address (адресі) және phone (телефон). Сонымен қатар, шығарылатын тауарлар goods кестесі келесі өрістерден тұрсын: type (тауар түрі), comp_id (компания идентификаторы), name (тауар атауы) және price (тауар бағасы). Кестелер арасындағы байланыс comp_id өрісі бойынша жүзеге асқан. Тауарлар және оларды өндіруші туралы қысқаша ақпараттары: тауар түрі, өндіруші аты және тауар бағасы және т.с.с. бар repr көрініс құру талап етілсін. Оператор келесі түрде болуы мүмкін:

```
CREATE VIEW
  repr
AS
  SELECT
    goods.type, companies.comp_name, goods.price
  FROM
    goods, companies
  WHERE
    goods.comp_id = companies.comp_id
```

7. ***Көріністі өшіру*** операторы келесі түрде жазылады:

DROP VIEW <көрініс аты>

Оператор бұрын құрылған көріністі өшіруге мүмкіндік береді. Ескере кетейік, өшіру кезінде сұранысқа қатысатын кестелер көрінісі өшірілмейді. Repr көрінісін өшіру келесі түрдегі операторымен жүзеге асады: DROP VIEW repr.

8. ***Жазбаларды таңдау*** операторы келесі түрде жазылады:

```
SELECT [ALL | DISTINCT]
      <деректердің тізімі>
      FROM <кесте тізімі>
      [WHERE <таңдау шарты>]
      [GROUP BY <бағана аты> [,<бағана аты>] ... ]
      [HAVING <іздеу шарты>]
      [ORDER BY <мамандану> [,<мамандану>] ...]
```

Бұл барлық SQL операторларының ішіндегі неғұрлым маңызды оператор. Оның функциональдық мүмкіндіктері үлкен. Олардың негізгілерін қарастырайық.

SELECT операторы бір немесе бірнеше кестелерден таңдауға және деректермен есептеулер жасауға мүмкіндік береді. Операторлың орындалу нәтижесі жауап кестесі болып табылады, онда қайталанатын жолдар болуы (ALL) немесе болмауы (DISTINCT) мүмкін. Үнсіз келісім бойынша жауап кестесі барлық жолдарды, сонымен қатар, қайталанатын жолдарды да қамтиды. Деректерді таңдауға бір немесе бірнеше кесте жазбасы қатысады, аталған тізімде FROM операндасы.

Деректердің тізімі сұраныста қолданылатын бағаналар атынан тұруы мүмкін. Қарапайым жағдайда өрнектерге бағаналар атын, арифметикалық операциялар белгілерін (+, – , * , /), тұрақтыларды және дөңгелек жақша жазуға болады. Егер деректердің тізімінде өрнек жазылған болса, онда деректерді таңдаумен бірге есептеулер орындалады, олардың нәтижелері жауап кестелердің жаңа (құрылатын) бағанасына жазылады.

Деректердің тізімінде бірнеше кестенің бағаналар аттарын қолданған кезде бағананың қайсыбір кестеге тиістілігін көрсету үшін келесі түрдегі конструкция қолданады: <кестелер аты>.<бағана аты>.

WHERE операндасы нәтижелік кесте жазбасы қанағаттандыруы тиіс шартты береді. <таңдау шарты> өрнегі логикалық болып табылады. Оның элементтері бағаналар аты, салыстыру операциялары, арифметикалық операциялар, логикалық байланыстар (ЖӘНЕ, НЕМЕСЕ, ТЕРІСТЕУ), жақшалар, арнайы функциялар LIKE, NULL, IN және т.б. болуы мүмкін.

GROUP BY операнды нәтижелік жиында жазбалар топтарын ерекшелеуге мүмкіндік береді. ***Топтар*** GROUP BY кілттік сөзімен аталған бағаналардағы мәндермен сәйкес келетін жазбалар болады. логикалық өрнектерде WHERE және HAVING операндаларын пайдалану үшін, сонымен қатар топтармен операциялар (есептеулер) орындау үшін топтарды бөлу талап етіледі.

Логикалық және арифметикалық өрнектерде келесі топтық операцияларды (функциялар) қолдануға болады: AVG (топтағы орташа мән), MAX (топтағы ең үлкен мән), MIN (топтағы ең кіші мәні), SUM (топтағы мәндердің қосындысы), COUNT (топтағы мәндер саны).

HAVING операндасы GROUP BY операндасымен бірлесе жұмыс жасайды және топтарды анықтау кезінде жазбалардың қосымша селекциясы үшін қолданылады. <іздеу шарты> жазу ережесі WHERE операндасының <таңдау шарты> құру ережесі.

ORDER BY операндасы нәтижелік жиынды сұрыптау ретін көрсетеді. Әдетте әрбір <мамандану> CREATE INDEX операторының сәйкес конструкциясына ұқсас және келесі түрде жазылады: <бағана аты> [ASC | DESC].

Ескерту.

SELECT операторы басқа неғұрлым күрделі синтаксистік конструкциялы болуы мүмкін, оны біз жан-жақты қарастырмаймыз, тек олардың мағынасын түсіндіреміз.

Мұндай конструкциялардың бірі, мысалы, ***ішкі сұраныстар*** болады. Олар *кірістірілген сұраныстарды* қалыптастырады, яғни логикалық өрнектерде таңдау шарты WHERE операндада бір SELECT операторының нәтижелері басқа SELECT операторының нәтижелерін қолданады.

SELECT операторын мамандандырылған пайдаланудың тағы бір мысалы – нәтижелік кестелерді біріктіру кезінде SELECT (UNION операндасы) операторы бірнеше рет орындалады.

Мысал 5. Жазбаларды таңдау.

Келесі өрістері бар EMP кестелері берілсін: NAME (аты), SAL (жалақы), MGR (жетекші) және DEPT (бөлімі). Қызметкерлердің атын және 100 бірлікке көбейген олардың жалақысын шығару талап етілсін. Таңдау операторын келесі түрде жазуға болады:

SELECT name, sal+100

FROM emp.

Мысал 6. Шартты таңдау.

EMP кестелерінен берілген уақытта жетекшілері жоқ бөлім аттарын шығару керек болсын. Осы сұраныс үшін SELECT операторын келесі түрде жазуға болады:

SELECT dept
FROM emp
WHERE mgr is NULL.

Мысал 7. Топтық таңдау.

EMP кестелеріндегі әрбір бөлімнің ең төменгі және ең жоғарғы жалақысын табу керек болсын. Осы сұраныс үшін SELECT операторын келесі түрде жазуға болады:

SELECT dept, MIN(sal), MAX(sal)
FROM emp
GROUP BY dept.

9. ***Жазбаларға өзгерістер енгізу*** операторы келесі түрде жазылады:

UPDATE <кестелер аты>
SET <бағана аты> = {<өрнек> , NULL }
[, SET <бағана аты> = {<өрнек> , NULL } ...]
[WHERE <шарт>]

Өзгеріс енгізуде UPDATE операторының орындалуы SET операндасының көмегімен бағана аттарын көрсетіп, WHERE операндасымен берілген шартты қанағаттандыратын жазбаларды табу арқылы орындалады.

Өріс жазбалардағы жаңа мәндер бос (NULL) болуы мүмкін, немесе арифметикалық өрнекке сәйкес есептеледі. Арифметикалық және логикалық өрнектерді жазу ережесі SELECT операторының ережелеріне ұқсас.

Мысал 8. Жазбалардағы өзгерістер.

EMP кестесі бойынша жалақысы 6000 бірліктен аз жұмысшылардың жалақысын 500 бірлікке өсіру керек болсын. Осы сұраныс үшін SELECT операторын келесі түрде жазуға болады:

UPDATE emp
SET sal = 6500
WHERE sal <= 6000.

10. ***Жаңа жазбалар кірістіру*** операторы екі түрлі нұсқаға ие:

INSERT INTO <кестелер аты>
[(<бағаналар тізімі>)]
VALUES (<мәндер тізімі>)

және

INSERT INTO <кестелер аты>
[(<бағаналар тізімі>)]
<SELECT ұсынымы>

Бірінші нұсқада VALUES операндасы көмегімен бағаналардағы деректер мәніне жаңа мәндер тізімі INSERT операторы арқылы енгізіледі. Бағаналар аттарының тізімі VALUES операндасы тізіміндегі мәндердің ретіне сәйкес болуы керек. Егер <бағаналар тізімі> берілмесе, онда <мәндер тізімінде> кестелердегі бағаналар құрылымы ретімен барлық мәндер атап өтілуі керек.

Екінші нұсқада INSERT операторы берілген кестеге SELECT көмегімен басқа кестелерден таңдап алынған жаңа жолдар енгізуге арналған.

Мысал 9. Жазбалар енгізу.

EMP кестесіне жаңа қызметкер туралы жазба енгізу. Осы сұранысты келесі түрде жазуға болады:

INSERT INTO emp

VALUES (“Ivanov”, 7500, “Lee”, “cosmetics”).

11. ***Жазбаларды өшіру*** операторы келесі түрде жазылады:

DELETE FROM <кестелер аты>

[WHERE <шарт>]

DELETE операторының орындалу нәтижесінде кестелерден WHERE операндасымен анықталған шартты қанағаттандыратын жолдар өшіріледі. Егер міндетті емес WHERE операндасы бос болса, яғни өшірілетін жазбаларды таңдау шарты болмаса, онда барлық жазбалар өшіріледі.

Мысал 10. Жазбаларды өшіру.

EMP кестесіндегі байланыстарда ойыншықтар (toy) бөлімін тарату негізінде осы бөлімнің барлық қызметкерлерін өшіру талап етілсін. Осы тапсырма үшін DELETE операторы келесі түрде жазылады:

DELETE FROM emp

WHERE dept = “toy”.

Қарытынды ретінде ескере кетейік, Дейт пікіріне сәйкес, SQL тілі реляциялық алгебра және реляциялық есептеулердің жиыны болып табылады. Онда алгебра элементері (UNION біріктіру операторы) және есептеулер (EXISTS бар болу кванторы) бар. Сонымен қатар, SQL тілі реляциялық толық.

Бақылау сұрақтары және тапсырмалар

1. Реляциялық үлгінің анықтамасын беріңіз және оның құрылымдық элементтерін атаңыз.
2. Деректердің реляциялық үлгісінің құраушы элементтерін сипаттаңыз.
3. Қатынастар түсінігін математикалық сипаттаңыз.
4. Қатынастар домені дегеніміз не?
5. Қатынастар сызбасының анықтамасын беріңіз.
6. Алғашқы кілттік қатынастар дегеніміз не? Кілт не үшін беріледі?
7. Қандай шарт орындалғанда кестені қатынас деп есептеуге болатынын атаңыз.
8. Индекс дегеніміз не? Индекстеу не үшін қолданылады?.
9. Бір деңгейлі индекстеу сызбасын суреттеңіз және оған сипаттама беріңіз.
10. Екі деңгейлі индекстеу сызбасын суреттеңіз және оған сипаттама беріңіз.
11. Екінші индекс деген не? Оның бірінші индекстеуден қандай айырмашылықтары бар?
12. Деректер қоры элементтерімен екінші индекстің байланыстарын ұйымдастырудың мүмкін сызбасын келтіріңіз.
13. Кестелерде деректермен айла-амалдар жасау кезінде тұтастықты бақылау механизмі әрекетін сипаттаңыз.
14. Сұраныстардың теориялық тілдіріне жалпылама сипаттама беріңіз.
15. Кодд ұсынған реляциялық алгебра операцияларын атаңыз және қиылысу мен көбейту операцияларын графикалық талдаңыз.
16. Жалғастыру операцияларының жалпы және жеке жағдайларын сипаттаңыз.
17. Реляциялық алгебра өрнегінің жазу ережесін атаңыз.
18. Дейт ұсынған реляциялық алгебраның қосымша операцияларын атаңыз және сипаттаңыз.
19. Реляциялық есептеулер нұсқаларын сипаттаңыз.
20. Келесі сұранысқа сәйкес кортеждердегі реляциялық есептеулер өрнегін жазыңыз: "барлық бөлшектерді жеткізетін жеткізушілер аттарын шығару".
21. QBE тілін сипаттаңыз.

22. Мысал элементі түсінігіне анықтама беріңіз және таңдауға сұраныс үлгісін пайдалану мысалын келтіріңіз.

23. Заманауи ДҚБЖ-ларда QBE тілінің жетілдірілген бағытын атаңыз.

24. SQL тілін сипаттаңыз.

25. Реляциялық алгебра келесі өрнектер ((S [П#] MINUS (SP WHERE Д# = 'P2') [П#]) JOIN S) [Аты] көмегімен P2 бөлшегін жеткізбейтін жеткізушілер атын алу керек. (3.7-сурет).

26. Pencraft фирмасы ұсынған тауарлардан басқа тауарларды жеткізуші фирмалардын аттарын анықтау.

4. ЖЕЛІДЕГІ АҚПАРАТТЫҚ ЖҮЙЕЛЕР

Желіде ақпараттық жүйелер құру және қолдану, бір жағынан, елеулі артықшылықтар береді, басқа жағынан, біршама мәселелер туындайды. Жеке жағдайда, мәселелер басқару және ақпараттарды қорғау жағынан пайда болады. Бөлімде компьютерлік желімен және ондағы ақпараттық жүйелермен байланысты негізгі ұғымдар, клиент-сервер архитектурасы нұсқалары, деректерді басқару және оларға қатынас құру, жергілікті желідегі ақпараттық жүйелер ерекшеліктері, Internet және intranet қарастырылады.

4.1. Негізгі ұғымдар

ЭЕМ желісінің негізгі ұғымдарын ашайық, желінің түрі және құрамын, бағдарламалық және аппаратық қамтамасыз етілуін, сонымен қатар ресурстарды басқару әдістерін қарастырайық.

Желінің түрі және құрамы

Желі деп деректерді жөнелту құралдарымен біріктірілген компьютерлер немесе жұмыс бекетіялары жиынтығын айтамыз. Деректерді жөнелту құралдары, өз кезегінде, жалпы жағдайда келесі элементтерден тұрады: байланысты компьютерлер, каналдық байланыстар (спутниктік, телефон желісі, сандық, талшықты-оптикалық, радио- және басқа), коммутациялық құрылғылар, ретрансляторлар, әртүрлі сигналдарды түрлендіргіштер және басқа элементтер мен құрылғылар.

Заманауи желіні әртүрлі белгілері бойынша жіктеуге болады: компьютердің шалғай орналасуы, топологиялар, тағайындалулар, атқаратын қызметтер тізімі, басқару принципі (орталықтандырылған және орталықтандырылмаған) бойынша, коммутация (коммутациясыз, телефондық коммутация, мақсаттар коммутациясы, хабарламалар, дестелер және дейтаграмдар) әдістері бойынша, жөнелту ортасы бойынша және т.с.с.

Компьютерлік желілерді қашықтығына байланысты ***жергілікті*** және ***ауқымды***.

Ерікті ***ауқымды желі*** жергілікті желімен қатар басқа ауқымды желілерді, сонымен қатар оған бөлек қосылатын компьютерлер (қашықтықтағы компьютерлер) немесе бөлек қосылатын енгізу-шығару құрылғысын қамтуы мүмкін. Ауқымды желінің негізгі төрт түрі ажыратады: қалалық, аймақтық, ұлттық және ұлтаралық. Енгізу-шығару құрылғысы ретінде, мысалы, басып шығару және көшірмелік құрылғы, кассалық және банк аппараттары, дисплейлер (терминалдар) және факстер қолданылады. Аталған желі элементтері бір-бірінен аздаған қашықтықта орналасуы мүүмкін. Ауқымды желі мысалы ретінде алғашқы желілердің бірі ARPANET, сонымен қатар Интернет желісін атауға болады.

Жергілікті есептеу желілерінде (ЖЕЖ) компьютерлер бірнеше километрге дейін қашықтықта орналасқан және әдетте 1-ден 10-ға дейін немесе одан да көп Мбит/с жылдамдығымен байланысқан байланыстардың шапшаң желілері көмегімен ақпарат алмасады. (компьютерлерді жылдамдығы төмен телефон желісі арқылы жалғастыру жағдайы да қарастырылады). ЖЕЖ әдетте қайсыбір қарым-қатынас жасау үшін корпорацияларда, мекемелерде ұйымдастырылады . Сондықтан оларды кей жағдайда *бірлескен жүйелер немесе желілер* деп атайды. Компьютерлер сонымен қатар, қағидаға сәйкес, бір мекеме, ғимарат немесе көршілес ғимарат ішінде орналасқан .

Ауқымды желіде тәуелсіз компьютерлердің арасындағы өзара әрекеттердің негізгі түрі хабарламалар алмасу болып табылады. Неғұрлым қарқынды ақпарат алмасу жергілікті желіде жүзеге асады. Оларда, табиғаты бойынша, барлық желіге кіретін компьютерлерді аппараттық-бағдарламалық ресурстармен басқару ұйымдастырылған. Бұл функцияларды желілік бағдарламалық қамтамасыз ету жүзеге асырады.

Бағдарламалық-аппараттық ресурстарға және оларды ЖЕЖ-де басқаруға қысқаша тоқталайық.

Жергілікті есептеу желісінде бағдарламалық қамтамасыз ету

Қандай желімен жұмыс жасайтынына тәуелсіз қайсыбір компьютерлер мен оларға орнатылған бағдарламалық қамтамасыз ету функцияларын шартты түрде екі топқа бөлуге болады: компьютердің *басқару ресурстары* (сонымен қатар және басқа компьютерлермен тапсырмалар орындау мақсатында) және басқа компьютерлермен *алмасып басқару* (желілік функциялар).

Компьютердің ресурстарын әдетте ОЖ басқарады. Желілік басқару функцияларын ***желілік бағдарламалық қамтамасыз ету*** жүзеге асырады, олар жекелеген желілік бағдарламалар дестесі немесе желілік ОЖ (ЖОЖ) түрінде жүзеге асуы мүмкін.

Желілік бағдарламалық қамтамасыз ету құру кезінде иерархиялық тәсілдеме қолданылады. Бұл бағдарламалар алгоритмін ерікті деңгейде, айтарлықтай өзгерістер енгізбей оңай түрлендіруге мүмкіндік береді. Жалпы жағдайда, қайсыбір деңгейде функцияны ықшамдауға немесе тіпті оны жоюға рұқсат береді.

Желілік бағдарламалық қамтамасыз етуді өңдеу және кез-келген есептеуіш жүйелердің өзара әрекетін қамтамасыз ету мүмкіндіктерін реттеу үшін халықаралық стандарттау бойынша ұйымдастыру (International Organization for Standardization – ISO) құрған ***Ашық жүйелердің өзара әрекетінің эталондық үлгісі*** (Open System Interconnection Reference model – үлгі OSI) бар.

Эталондық үлгі келесі жеті функциональдық деңгейді анықтайды:

- физикалық (physical layer);
- канальдық немесе жөнелтулерді (data link) басқару желісімен (бөліммен);
- желілік (network layer);
- көліктік (transport layer);
- сеанстық (session layer);
- өкілдік (presentation layer);
- қолданбалы немесе қосымша деңгей (application layer).

Желілердің бір бірінен айырмашылығы қолданылатын аппараттық және бағдарламалық қамтамасыз ету ерекшеліктерінен, бағдарлама құрушы фирмалардың нұсқауларын әртүрлі талдаулардан, шығарылатын есептерден жүйелерге қойылатын талаптардың әртүрлілігінен (ақпараттарды қорғау талабы, ақпарат алмасу жылдамдығы, деректерді қателіктерсіз жөнелту және т.б.) және басқа себептерден шығады. Жергілікті желіні желілік бағдарламалық қамтамасыз етуде жүзеге асатын деңгейді төмендету жиі байқалады.

ЖЕЖ аппараттық құралдары

ЖЕЖ негізгі аппараттық құрамдас бөліктері: жұмыс бекеті, серверлер, интерфейтік тақшалар және кабельдер.

Жұмыс бекеті (ЖБ) – бұл, қағидаға сәйкес, желіні пайдаланушылардың жұмыс орны болатын дербес ЭЕМ.

Кей жағдайда тікелей желілік кабельге қосылған жұмыс бекетінда магниттік дискідегі жинақтауыштар болмауы мүмкін. Ондай жұмыс бекеті ***дискісіз жұмыс бекеті*** деп аталады. Дискісіз жұмыс бекеттерінің негізгі *артықшылығы* құнының төмендігі, сонымен қатар жүйелерге рұқсатсыз қатынас құрудан және компьютерлік вирустардан қорғаудың жоғары деңгейлігі болып табылады. Дискісіз жұмыс бекеттерінің *кемшілігі* автономды режимде (серверге қосылмай) жұмыс жасау мүмкін емес және өзіндік деректер мен бағдарламалардың архиві жоқ.

ЖЕЖ-де ***серверлер*** желілік ресурстарда орналасқан функцияларды орындайды. Әдетте оның функциялары жеткілікті қуатты ДК, мини-ЭЕМ, үлкен ЭЕМ немесе арнайы ЭЕМ-серверде жүктеледі. Бір желіде бір немесе бірнеше сервер болуы мүмкін.

Әрқайсысы бір бөлімнің сервері болуы мүмкін. Соңғы жағдайда тек бөлім ресурстары ортақ пайдаланылады.

ЖЕЖ-де бірнеше серверлер болған кезде, олардың әрқайсысы өзіне қосылған жұмыс бекетінің жұмысын басқарады. Сервер компьютерлері жиынтығы және оларға жататын жұмыс бекеті әдетте ***домен*** деп аталады. Кей жағдайда бір доменде бірнеше серверлер орналасады. Әдетте олардың біреуі негізгі болып табылады, ал басқалары – сақтық қоры ролін орындайды (негізгі сервер істен шыққан кезде) немесе негізгі сервердің кеңейтпесі болады.

Қолданылатын желілік адаптерлер үш негізгі сипаттамаларға ие: компьютер шиналарының типі (ISA, EISA, Micro Channel және т.б.), дәрежелігі (8, 16, 32,64) және деректердің желілік каналына қатынас құру үшін қолданылатын әдісі.

ЖЕЖ-де компьютерлерді біріктірудің әртүрлі сызбасы бар. Классикалық әдістері "жұлдыз", "сақина" және "жалпы шина" топологиялары болып есептеледі. Желілік каналға қатынас құрудың келесі стандартты әдістері неғұрлым кеңінен қолданылады:

- Ethernet (шиналық топологияны сүйемелдейді);
- Arcnet (жұлдыз топологиясын сүйемелдейді);
- Token-Ring (сақиналық топологияны сүйемелдейді).

Элементтерді желіге жалғастыру коҚФигурациясы (топология) көбінесе желінің маңызды сипаттамаларын анықтайды: сенімділігі, өнімділігі, құны, қорғалуы және т.с.с.

ЖЕЖ топологиясын жіктеу тәсілдемелерінің бірі топологияны негізгі екі топқа бөлу болып табылады: ***кеңінен таратушы*** және ***тізбектей***.

Кеңінен таратушы коҚФигурациясында әрбір дербес компьютер басқа компьютерлер қабылдауы мүмкін сигналдар жібереді. Мұндай коҚФигурацияларға "жалпы шина", "ағаш тәріздес", "пассивті орталықты жұлдыз" топологиялары жатады. "пассивті орталықты жұлдыз" топологиясын "ағаш тәріздес" топологияның бір түрі ретінде қарастыруға болады.

Тізбектей конфигурациясында әрбір физикалық ішкі деңгей ақпаратты бір компьютерге жібереді. Тізбектей конфигурациясының мысалдары мыналар болады: ерікті (компьютерлерді ерікті жалғастыру), иерархиялық, "сақина", "тізбектелген", "интеллектуальды орталықты жұлдыз", "снежинка".

Компьютерлерді ЖЕЖ-де жалғастыру үшін көбінесе коаксиальды кабель (жіңішке және жуан) қолданады. Коаксиальды кабельден басқа оптоталшықты кабель қолданылуы мүмкін. Соңғы уақытта сымсыз радиожелі құру және енгізу бойынша интенсивті жұмыс жүргізіліп жатыр. Оларға негізделген жүйелер кабельді жүйелермен салыстырғанда деректерді жөнелтудің және алыс қашықтыққа жөнелтудің (жүздеген метр) жылдамдығы бойынша төмен.

ЖЕЖ қосымша құрылғыларына жататындар: үздіксіз қорек көзі, модемыдер, трансиверлер, қайталағыштар, сонымен қатар әртүрлі айырғыштар (коннекторлар, терминаторлар).

Басқару принциптері

ЖЕЖ-де басқарудың екі негізгі әдісі (принципі) бар: орталықтандырылған және орталықтандырылмаған. *Орталықтандырылған* желіде бір немесе бірнеше ДЭЕМ орталық болады, ал қалғандары – жұмыс бекеттері (ЖБ).

Компьютер-сервер (КС) деп аталатын желінің орталық түйіні бөлек компьютерде орналасуы мүмкін немесе ЖБ-мен бірге. Бірінші жағдайда ерекшеленген компьютерлік желі, ал екіншісінде – қосарланған компьютерлік желі туралы айтылады.

КЖ-нің негізгі тағайындалуы – желіде деректерді жөнелтуді басқару және көптеген ЖБ қолданатын файлдарды сақтау. Сонымен қатар, КС-ге әдетте қымбат құрылғылар (лазерлік принтерлер, факстер, модемдер, сканерлер және т.б.) қосылады.

Орталықтандырылған басқаруды жүзеге асыратын желілік операциялық жүйелер бар. Олар: Microsoft Windows NT Server, Novell NetWare (3.X және 4.X нұсқалары), Microsoft Lan Manager, OS/2 Warp Server Advanced, VINES 6.0 және басқалар.

Орталықтандырылған желі *артықшылығы* желілік ресурстарды рұқсат етілмеген қатынас құрулардан қорғау деңгейінің жоғарылығы, желіні басқарудың тиімділігі, желінің көптеген түйіндерімен қатынас құру мүмкіндігі болып табылады. Негізгі *кемшілігі* файл-сервер жұмыс қабілетінің бұзылуы жүйелер жұмысына да әсер етеді (бұл қиындықты бірнеше сервер жұмыс жасағанда немесе басқа да іс-шаралар көмегімен жоюға болады), сонымен қатар сервер ресурстарына қойылатын талаптардың жоғарылығы.

Орталықтандырылмаған басқару желісінде (біррангілі желі) КС болмайды. Оларда желіні басқару функциялары қайсібір тәртіпке сәйкес бір жұмыс бекетінен басқасына кезектесіп жіберіледі. Орталықтандырылмаған басқару, қағидаға сәйкес, мүмкіндігі төмен компьютерлер арасында қарапайым деңгейде файлдар алмасу үшін, сонымен қатар желі ресурстарына қатынас құру құқықтарын қатаң бақылаусыз қолданылады. Барлық ЖБ негізгі ресурсы әдетте ортақ пайдаланылады, бірақ жүйелер оларды бірлесе қолдануды әрқашан қамтамасыз етпедейді.

Біррангілі желі құруға мүмкіндік беретін неғұрлым кеңінен таралған бағдарламалық өнімдер келесі бағдарламалар және дестелер болады: Novell NetWare Lite, Windows for Workgroups, Windows 95/98, Artisoft LANtastic, LANsmart, Invisible Software NET-30 және басқалар.

Аздаған ЖБ үшін біррангілі желі қолдану неғұрлым тиімді және төзімді есептеу ортасын тұрғызуға мүмкіндік береді. Желілік бағдарламалық қамтамасыз ету оларда орталықтандырылған желілермен салыстырғанда неғұрлым қарапайым болып табылады. Мұнда файл-серверді (компьютерді және сәйкес бағдарламаларды) орнату талап етілмейді, олар жүйелерді анағұрлым арзандатады. Бірақ, ондай желі ақпараттарды қорғау және басқару тұрғысынан қарағанда әлсіз.

Желі туралы сөз болғанда жиі қолданылатын терминдер: “сервер” және “клиент”. Желідегі кез-келген өзара әрекет ең болмағанда екі элементті жобалайды: тұтынушы және ресурстарды ұсынушы (сервис немесе қызметтер). Ресурстарды тұтынушы - клиент, ал ресурстарды ұсынушы компонент – сервер деп аталады (1-бөлімді қараңыз). Ресурстардың негізгі түрлері: аппараттық (бүтіндей ЭЕМ, дискілік жинақтауыштар, басып шығару құрылғысы және т.б.), бағдарламалық және ақпараттық.

Егер ресурс ретінде бүтіндей ЭЕМ қарастырылса, онда компьютер-клиент және компьютер-сервер туралы сөз қозғалады. Егер қайсыбір аппараттық ресурс қарастырылса, онда келесі терминдер қолданылады: диск-сервер (файлдар сервері немесе файл-сервер), басып шығару сервері. Желінің әдеттегі клиенттері: ЭЕМ, пайдаланушы немесе бағдарлама.

Компьютердің ақпараттарды өңдеу кезінде тағайындалуы мүмкін. Онда хабарламалар сервері (қабылданған хабарламаларды өңдеу), ДҚ серверлері (ДҚ-на сұраныстарды өңдеу), қосымшалер сервері (пайдаланушының қосымшаны орындауы) және т.б. туралы сөз болады.

Кей жағдайда есептеуіш жүйелердің әртүрлі компонентері бір терминмен (аппараттық, бағдарламалық немесе аппараттық-бағдарламалық) аталады. Ендеше, басып шығару сервері принтер қосылған компьютерлерге, басып шығару бағдарламаларына немесе басып шығаруды жүзеге асыратын бағдарламалары бар компьютерге қызмет атқаруы мүмкін.

4.2. Клиент-сервер құрылымы үлгісі

ДҚ-мен жұмыс жасайтын үлестірімді АЖ тұрғызған кезде клиент-сервер құрылымы кеңінен қолданылады. Оның негізін ДҚ басқарған кезде клиент пен сервердің өзара әрекетін ұйымдастыру принциптері құрайды. Клиент-сервер архитектурасы нұсқаларының бірі 1.2-бөлімде қарастырылған.

ДҚ басқару процестерінің өзара әрекетінің негізгі сызбасын сипаттау үшін OSI ашық жүйелер архитектурасының эталондық үлгісін қолданамыз. Бұл үлгілерге сәйкес, ДҚ басқару функциясы қолданбалы деңгейге жатады.

Екі ең жоғары деңгейге тоқталайық: *қолданбалы* және *көрсеткіштік,* олар бағдарлама құрушы және пайдаланушы тарапынан өтң үлкен дәрежеде назар аударарлық болады. Қалған функцияларды алдыңғы екеуін жүзеге асыру үшін қажетті байланыстырушы функциялар деп есептейік. Сонымен қатар, ДҚБЖ ұғымын ДҚ-дағы ақпаратты қолданатын барлық бағдарламалық жүйелер ретінде кеңінен қарастырамыз.

Бағдарламаның пайдаланушымен интерфейсін сүйемелдейтін ДҚБЖ, жалпы жағдайда, келесі негізгі функцияларды жүзеге асырады:

- қордағы деректерді басқару;
- қолданбалы бағдарламалар көмегімен ақпараттарды өңдеу;
- пайдаланушы үшін тиімді түрде ақпараттардың бейнеленуі.

Егер жүйелер бір ЭЕМ-де орналасса, онда барлық функциялар бір бағдарламада жинақталған және 1-бөлімде қарастырылған сызба бойынша шақырылады.

Желіге ДҚБЖ орналастыру кезінде түйіндер бойынша функцияны орналастырудың әртүрлі нұсқалары бар. Түйіндер санына тәуелді, ДҚБЖ функцияларын олардың арасында бөлуде *екібуынды* үлгілер, *үшбуынды* үлгілер және т.б. атап айтуға болады. Функциялардың арасындағы үзік орны коммуникациялық функциялармен (ақпараттарды желіге жөнелту ортасы) жалғастырылады.

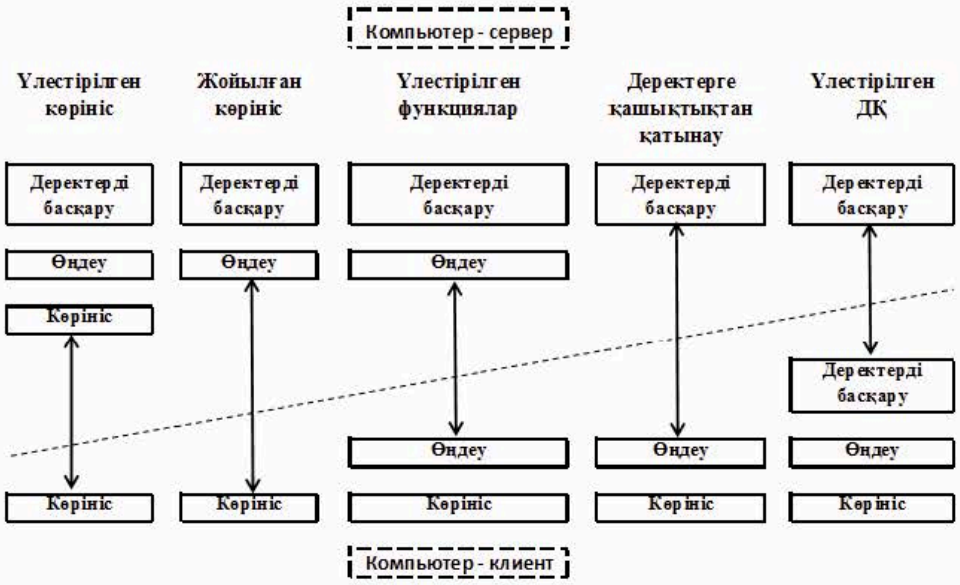
Функциялар орналасуының екібуынды үлгісі

Екібуынды үлгілер ДҚБЖ функцияларының *желінің екі түйіні* арасында орналасқанына сәйкес келеді. Міндетті түрде деректерді басқару функциясы бар компьютерді (желі түйіні) ***компьютер-сервер*** деп атайды. Пайдаланушы қолданатын, ақпараттарды өңдеумен айналысуы міндетті емес компьютер ***компьютер-клиент*** деп аталады.

Компьютер-сервер және компьютер-клиент арасындағы функцияларды бөлудің нұсқалары:

- орналасқан жердегі бейнеленуі;
- қашықтықтан бейнеленуі;
- орналасу функциясы;
- деректерге қашықтықтан қатынас құру;
- орналасқан ДҚ.

Аталған әдістер жүйелерде функциялардың орналасуының клиент-сервер архитектурасымен әртүрлі нұсқаларын бейнелейді: барлық жұмыстар жүзеге асатын қуатты серверден қуатты клиентке дейін, мұнда функциялардың көптеген бөлімі жұмыс бекетінде орындалады, ал сервер желі бойынша SQL-шақыруымен өзіне түскен деректерді өңдейді.



4.1-сурет. Клиент-сервер архитектурасының модельдері спектрі

Деректерге қашықтықтан қатынас құру және қашықтықтан бейнелену модельдері компьютер-клиент және компьютер-сервер арасында функцияларды қатаң бөлу арқылы жүзеге асады. Басқа модельдерде бір уақытта екі компьютерде келесі функциялардың бірі орындалады: деректерді басқару (үлестірімді ДҚ үлгісі), ақпаратты өңдеу (үлестірімді функциялар үлгісі), ақпараттардың бейнеленуі (көрініс үлгісі).

Алдымен, неғұрлым кеңінен таралған *деректерге қашықтықтан қатынас құру* және *қашықтықтан бейнеленуі (ДҚ сервері)* үлгілерін қарастырайық.

Деректерге қашықтықтан қатынас құру (Remote Data Access – RDA) үлгісінде бағдарламалар, ақпараттардың бейнеленуін жүзеге асыратын функциялар және қолданбалы өңдеу логикасы өзара байланысты және компьютер-клиентте орындалады. Деректерді басқару сервисіне қатынас құру SQL тілі операторлары көмегімен жөнелту ортасында немесе арнайы API (Application Programming Interface – қолданбалы бағдарламалау интерфейсі) кітапханасын шақыру функциясы арқылы жүзеге асады.

RDA-үлгілердің негізгі ***құндылығы*** SQL-интерфейстер және аспаптық құралдары бар бағдарламаларды жылдам құруды жүзеге асыратын дайын ДҚБЖ-дан тұрады. Өңдеу құралдары MS Windows операциялық жүйесінде көбінесе пайдаланушының графикалық интерфейсін, ODBC интерфейсі стандартын және автоматты түрде кодты іздестіру құралдарын сүйемелдейді. Көптеген өңдеу құралдары төртінші деңгейлі тілдерді қолданады.

RDA-үлгілердің ***кемшіліктері***: біріншіден, деректерді жөнелтуде жүйелердің жоғары талаптармен жүктелуі, себебі барлық логика қосымшаларға негізделген, ал өңделетін деректер шалғайдағы түйінде орналасқан. Жұмыс кезінде әдетте желі бойынша бүтіндей ДҚ жіберілетінін көреміз.

Екіншіден, RDA үлгілері негізінде тұрғызылған жүйелер өңдеу, түрлендіру және сүйемелдеу тұрғысынан қарағанда қолайсыз. Негізгі себеп: алынатын қосымшаларда қолданбалы функциялар және функциялар көрінісі өзара тығыз байланысты. Сондықтан, функцияларды тіпті кішігірім өзгертулер кезінде жүйелер оның барлық қолданбалы бөлімдерін өзгертуді талап етеді, бұл жүйелермен өңдеу және іс-әрекеттер жасауды қиындатады.

ДҚ сервері үлгісінің (DataBase Server – DBS) басқа үлгілерден айырмашылығы: компьютер-клиенттің функциялары ақпараттардың көрінісі функциясымен шектелген, компьютер-серверде орналасқан қосымшалармен қолданбалы функциялар қамтамасыз

етеді. Бұл үлгі RDA-үлгілерге қарағанда неғұрлым тиімді болып табылады және мұнда келесі ДҚБЖ қолданылады: Ingress, Sybase және Oracle. Сонымен қатар, қосымша *сақталатын процедуралар* түрінде жүзеге асады.

Процедуралар әдетте ДҚ сөздігінде сақталады және бірнеше клиенттермен бөлінген. Жалпы жағдайда, сақталатын процедуралар компиляциялар және талдаулар режимінде орындалуы мүмкін.

DBS үлгілерінің ***құндылықтары***: өңдеу кезеңінде қосымшаны орталықтандырылған басқару, сүйемелдеу және түрлендірулер мүмкіндігі, сонымен қатар есептеуіш және коммуникациялық ресурстарды тиімді қолдану. Соңғысы бағдарламаларды ұжымдық пайдалану режимінде орындауды талап етеді, бұл шығынды азайтады.

DBS үлгілері ***кемшіліктерінің*** бірі сақталатын процедураларды өңдеу құралдарына шектеулермен байланысты. Негізгі шектеу –сақталатын процедуралар операторының нақты ДҚБЖ-ға тәуелділігі. Сақталатын процедураларды жазу тілі, мағынасы бойынша, SQL тілінің процедуралық кеңейтпесі болып табылады және көрнекі құралдары және функционалдық мүмкіндіктері бойынша үшінші деңгейлі басқа дәстүрлі тілдермен, C және Pascal сияқты, таласа алмайды. Сонымен қатар, көбісінде ДҚБЖ-да сақталатын процедураларды жүзеге асыратын және тестілейтін құралдар жоқ, сондықтан бұл оларды қауіпті құрал механизмі етеді – тексерілмеген бағдарламалар ДҚ қателіктерге соқтыруы мүмкін, жүйелер жұмысы кезінде серверлік және клиенттік бағдарламалардың тоқтап қалуы мүмкін және т.с.с.

DBS-үлгілердің тағы бір кемшілігі: ЭЕМ есептеуіш ресурстарын пайдаланудың тиімділігі төмен, себебі компьютер-сервердің бағдарламаларына сұраныстар ағымын басқаруды, сонымен қатар басқа компьютер-серверлерге процедураларды ауыстыруды қамтамасыз етуді ұйымдастыру жүзеге аспайды.

Орналасқан жердегі бейнеленуі үлгілерінде қуатты компьютер-сервер бар, ал клиенттік бөлім практикалық тұрғыдан жасанды. Клиенттік бөлімдер функциялары ақпараттардың монитор экранында қарапайым бейнеленуі және жергілікті желі арқылы негізгі компьютермен байланысы болып табылады.

Осы сияқты ДҚБЖ *X-терминалдар* деп аталатын жұмысты сүйемелдейтін желілерде қолданылады. Оларда негізгі компьютер (хост-машина) бірнеше X-терминалдар тобына қызмет көрсету үшін жеткілікті түрде қуатты болуы керек. X-терминалдың да жылдам процессоры және жедел жадыда жеткілікті көлемі (дискілік жинақтауыштары жоқ) болуы керек. Командалардың қысқартылған жинағы бар компьютерлерде X-терминалын әдетте RISC-компьютерлер (restricted [reduced] instruction set computer) негізінде құрады. Барлық бағдарламалық қамтамасыз ету хост-машинада орналасады. X-терминалды бағдарламалық қамтамасыз ету X-терминалды серверден қосу кезінде желі бойынша жүзеге асады.

Орналасқандағы көрініс үлгісі алғашқы ДҚБЖ қолданылды және кіші, орташа және үлкен ЭЕМ-да жұмыс жасады. X-терминал ролін дисплейлік бекеттер және абоненттік пункттер (жергілікті және қашықтықтағы) атқарды. Бұл жағдайда ақпараттарды бейнелеу функциясының негізгі бөлімін ДҚБЖ өздері атқарды, ал нәтижелік тұрғызулар пайдаланушының терминалында соңғы құрылғыларда орындалды.

Орналасқандағы көрініс үлгісі бойынша ДҚ пайдаланушыларға қызмет көрсету жүйелері біртекті емес ортада жүзеге асады. Мұндай жүйелердің серверлік бөлімі әдетте қайсыбір ерекшеленген интерфейсті қамтамасыз етеді, ал клиенттік бөлімдер соңғы құрылғының мамандануын ескеру немесе ақпараттарды бір форматтан басқа форматқа түрлендіру функцияларын жүзеге асырады.

Орналасқандағы көрініс үлгісі есептеуіш ресурстарды басқарудың орталықтандырылған сызбасын жүзеге асырады. Қызмет көрсетудің және жүйелерге қатынас құруды басқарудың қарапайымдылығы мен салыстырмалы түрде

арзандығынан оның негізгі ***құндылығы*** шығады. ***Кемшіліктері***: орталық түйіндер сенімділігі төмен болған жағдайда жүйелердің тұрақсыздығы, сонымен қатар клиенттер саны көп болған кезде серверге өнімділігі бойынша қойылатын жоғары талаптар.

Үлестірімді функциялар үлгісінде деректерді өңдеу орналасқан екі түйін бойынша жүзеге асады. Бұл үлгіде АЖ болуы мүмкін, онда қолданбалы функциялардың жалпы бөлімі компьютер-серверде жүзеге асқан, ал ақпаратты өңдеудің арнайы функциялары компьютер-клиентте орындалады. Жалпы сипаттағы функциялар деректердің тұтастықты қамтамасыз ету стандартын қамтуы мүмкін, мысалы, сақталатын процедуралар, ал қалған қолданбалы функциялар арнайы қолданбалы өңдеуді жүзеге асырады. Осындай үлгі АЖ-де қолданылады.

Үлестірімді ДҚ үлгісі қуатты компьютер-клиентті қолдануды жобалайды, сонымен қатар деректер компьютер-клиентте және компьютер-серверде сақталады. Екі деректер қорының өзара байланысы екі түрлі болуы мүмкін: а) жергілікті және шеткері қорлардың жеке бөлімдері ортақ ДҚ-да сақталады; б) жергілікті және шеткері ДҚ бір-бірімен көшірмелерімен үйлестірілген болады.

Үлестірімді ДҚ үлгілері ***құндылығы*** оның негізінде құрылатын АЖ өзгеріске ұшырауға төзімділігі болып табылады, бұл компьютер-клиентке жергілікті және қашықтықтағы ДҚ өңдеуге мүмкіндік береді. Координация механизмдері болған кезде жүйелердің көшірмелерін бүтіндей сәйкестендіруге болады, сонымен қатар, ол төзімді, өйткені клиент және сервер байланысының үзілуі жүйе жұмысын істен шығармайды, ал оның жұмысы жалғастыру арқылы қалпына келтірілуі мүмкін. Үлгілердің ***кемшілігі*** компьютер-клиенттерде бірдей қосымшалардың орындалуы кезінде шығатын шығынның көптігі.

Функциялар орналасуының үшбуынды үлгісі

Функциялар орналасуының ***үшбуынды үлгісі*** типтік нұсқа болып саналады, онда қосымшаның үш функциясының әрқайсысы бөлек компьютерде жүзеге асады. Функциялар орналасуының нұсқалары бірнеше компьютерде орындалуы мүмкін, бірақ оларды қарастырмаймыз.

Біз қарастырып отырған үлгі ***сервер қосымшасы үлгісі*** немесе ***AS-үлгі*** (Application Server) деген атауға ие және 4.2-суретте көрсетілген.

4.2-сурет . Үшбуынды сервер қосымшасы үлгісі

Соңғы пайдаланушымен сұхбаттасу процесін ұйымдастыруға жауап беретін үшбуынды AS-үлгілерге сәйкес, әдетте функциялар көрінісі ақпараттардың орындалуын жүзеге асырады және DBS үлгілері сияқты қосымша компоненттерімен

өзара әрекет жасайды. Қосымша компоненті бөлек компьютерде орналасып, өз кезегінде, RDA үлгілері сияқты деректерді басқару компонентімен байланысты.

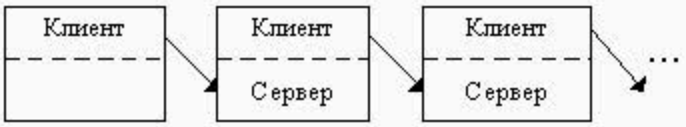
AS-үлгілердің орталық звеносы сервер қосымшасы болып табылады. Серверде қосымшаны бірнеше қолданбалы функциялар жүзеге асырады, олардың әрқайсысы осы бағдарлама талап ететін барлық қызмет түрін ұсынушы ретінде рәсімделген. Олар бірнеше болуы мүмкін, сонымен қатар олардың әрқайсысы өзіне тән қызмет түрін ұсынады. Қосымша қызметін пайдаланатын кез-келген бағдарлама ол үшін клиент болып табылады. Клиенттерден серверге түсетін сұраныстар қайсыбір тәртіпке сәйкес, мысалы, басымдылықтары бойынша орналасады.

Функциялар көрінісін жүзеге асыратын және сервер үшін клиент болып табылатын компонент, бұл үлгілерде әдеттегіден неғұрлым кеңінен таралған. Ол соңғы пайдаланушымен интерфейс ұйымдастыру, құрылғылардан (мысалы, датчиктерден немесе ерікті бағдарламалардан) деректерді қабылдауды қамтамасыз ету үшін қолданылуы мүмкін.

AS-үлгілер ***құндылығы*** қосымшаның функцияларын үш тәуелсіз құрылымға бөлу кезіндегі икемділігі және әмбебаптығы болып табылады. Көптеген жағдайларда бұл үлгі екібуындымен салыстырғанда неғұрлым тиімді болады. Үлгілердің негізгі ***кемшілігі*** – қосымша компонентері арасында ақпараттармен алмасу кезінде компьютерлердің ресурстарының неғұрлым жоғары шығыны.

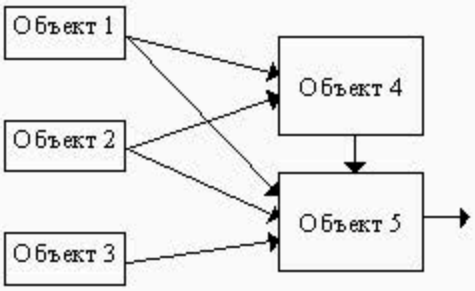
Өзара әрекеттің күрделі сызбалары

Өзара әрекеттің неғұрлым *күрделі сызбалары* болуы мүмкін, мысалы, қайсыбір клиент үшін сервер болып табылатын элемент, өз кезегінде, басқа серверге қатынасы бойынша клиент ролін атқаратын (4.3-сурет) сызбалар. Мысалы бұны біз AS-үлгілерден байқадық.



4.3-сурет. Клиент-сервер типті тізбектелген өзара әрекеттер

Үлестірімді есептеуіш жүйелерде ДҚ-мен жұмыс жасау кезінде бір объектіге қатынасы бойынша клиент, ал басқаға қатынасы бойынша – сервер (0.4-сурет) болатын *көптеген* байланыстар (статистикалық) болуы мүмкін.



4.4-сурет. Клиент-сервер типті көптеген байланыстардың өзара әрекеті

Динамикада объектілердің өзара әрекетін қарастыру кезінде өзара әрекеттің неғұрлым күрделі сызбалары алынады. Мұндай сызбаның мысалы жұмыс кезінде объектілер

ролі өзгеретін жағдайлар болып табылады: қайсыбір клиент үшін сервер болып, басқа серверге қатынасы бойынша клиент ролін атқаратын объект.

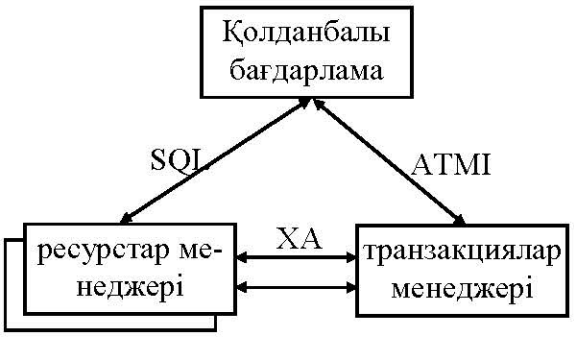
Транзакциялар мониторы үлгісі

Жоғарыда байқалғандай, ДҚБЖ функциялар орналасуының неғұрлым икемді және әмбебап үлгісі AS-үлгі болып табылады. Ол үш негізгі элементтердің: клиент, қосымшалар сервері және ДҚ серверінің өзара әрекетін сипаттайды, бірақ ақпараттарды өңдеу кезінде бағдарламалық қамтамасыз етудің жұмысын ұйымдастыру, жеке жағдайда транзакциялардың орындалуы кезіндегі мәселелерді қозғамайды. Бұл кемшілікті жою үшін *транзакциялар мониторы үлгісін* ұсынған.

Транзакцияларды өңдеу мониторы (Transaction Processing Monitor – TPM), немесе *транзакциялар мониторы* дегеніміз үлестірімді есептеуіш жүйелерде ақпараттық-есептеуіш ресурстарды тиімді басқаруды жүзеге асыратын аралық топ (Middleware) санатты бағдарламалық жүйелер. Олардың негізгі тағайындалуы – қосымшаларды өңдеу және басқару үшін, транзакциялар үлестірімдерін шапшаң орындайтын икемді, ашық орта ұйымдастыру. Гетерогенді есептеу жүйелерінде транзакциялар мониторын қолдану неғұрлым тиімді.

Қосымшалар жүйелердің көлемін өзгертуге, қосымшаның функциональдық толықтығын және тұтастығын сүйемелдеуге, сонымен қатар жөнелтпе құжат шығыны аз болатын деректерді өңдеу өнімділігін арттыруға мүмкіндік береді.

Ақпаратты өңдеуді ұйымдастыру принциптері ***транзакциялар мониторы үлгісімен*** X/Open DTP (Distributed Transaction Processing – үлестірімді транзакцияларды өңдеу) транзакциялар мониторы көмегімен суреттеледі. Бұл үлгі (4.5-сурет) үш объектіні іске қосады: қолданбалы бағдарламалау, ресурстар менеджері (Resource Manager – RM) және монитор, немесе транзакциялар менеджері (Transaction Manager – TM).



4.5-сурет. X/Open DTP транзакцияларды өңдеу үлгісі

Қолданбалы бағдарламалар ретінде ерікті бағдарлама-клиент қолданылуы мүмкін. RM қайсыбір түрдегі ресурстар сервері функцияларын орындайды. Қолданбалы бағдарлама RM көмегімен арнайы функциялар жинағымен немесе SQL (ДҚ сервері сервер болған кезде) операторларымен өзара әрекет жасайды.

ATMI (Application Transaction Monitor Interface – қосымшалар мониторы транзакциялар интерфейсі) интерфейсі қайсыбір бағдарламалау, мысалы, C тіліндегі функцияларды шақыруға мүмкіндік береді.

Ресурстар менеджері функцияларын әдетте ДҚ серверлері немесе ДҚБЖ орындайды. Үлестірімді транзакцияларды (есептеуіш желінің бағдарламалық объектілеріне қатысты транзакциялар) өңдеу процесін басқаруды ұйымдастыру кезінде RM-мен өзара әрекет жасайды, ол транзакциялардың екіфазалы фиксация хаттамасын сүйемелдеуі (4.3 қараңыз) және X/Open XA стандартын қанағаттандыруы керек.

Транзакциялардың екіфазалы фиксациясының хаттамасын сүйемелдейтін ДҚБЖ мысалдары Oracle 7.0, Open INGRES және Informix-Online 5.0 болады.

Транзакция түсінігі ТРМ-де ДҚБЖ-ға қарағанда анағұрлым кең мағынада қолданылады, мұндағы транзакция деректер қорымен қарапайым әрекеттерді іске қосады. Транзакция басқа да қажетті әрекеттерді қамтуы мүмкін: хабарламалар жіберу, ақпараттарды файлға жазу, құрылғыларды қарастыру және т.б. Бұл ТРМ есептеу процестерін басқарудың неғұрлым қуатты құралдарын ұсынатынын білдіреді. ТРМ сүйемелдейтін транзакциялар ***қолданбалы*** немесе ***бизнес-транзакциялар*** деп аталады.

X/Open DTP үлгісі ТМ құрылымын толық ашпайды, ол үлестірімді жүйелердің ақпаратты өңдеу компоненттерінің құрамын және бұл компоненттердің бір-бірімен өзара әрекеттесуін анықтайды. Практикалық тұрғыдан бұл үлгілердің жүзеге асуы, әрине, бір-бірінен өзгешеленуі мүмкін. Транзакциялар мониторын жүзеге асыру мысалы ретінде келесілерді атауға болады: ACMS, CICS, және TUXEDO System.

Қосымшаны құрушы бағдарлама үшін транзакцияларды өңдеу мониторы ТРМ қосымшаны бірнеше функционалдық деңгей бойынша стандартты интерфейстермен декомпозиция құру мүмкіндігіне байланысты қолайлы, ол оңай өзгертілетін АЖ құруға мүмкіндік береді.

Қолданбалы бағдарламалар, практикалық тұрғыдан қарағанда, пайдаланушымен және ресурстар менеджерімен интерфейстің нақты жүзеге асуынан тәуелсіз болып келеді. Біріншісі, функция көрінісінің жүзеге асуы үшін кез-келген тиімді және үйреншікті бағдарлама құрушы құралдарды (С тілінен CASE-жүйелерге дейін) таңдауға болатынын білдіреді. Ресурстар менеджерінен тәуелсіздік бір ресурстар менеджерін басқа ресурстар менеджеріне оңай ауыстыруға мүмкіндік береді, тек қана олар қолданбалы бағдарламалармен өзара әрекеттесу стандартын (ДҚБЖ үшін SQL тілі) сақтаса болғаны.

Егер ТРМ аппараттық-бағдарламалық платформалар жиынын, мысалы, TUXEDO System сүйемелдесе, онда құрылатын қосымшалар, сонымен қатар, бейімделгіш болып келеді.

Барлық қолданбалы функциялардың серверде шоғырлануы және басқару мүмкіндіктерінің зор болуы қолданбалы функцияларды (бизнес-функциялар) жаңартуды және олардың қарама-қайшылықтарға ұшырамауын бақылауды анағұрлым жеңілдетеді. Қолданбалы функцияларға өзгерістер енгізу, сонымен қатар, бағдарлама-клиенттерге ешқандай әсер етпейді.

Үлестірімді жүйелерді пайдаланушылар үшін ТРМ қажетті өтеу мүмкіншілігі және жауап беру уақыты көрсеткіштерін жақсартады, есептеу процесін тиімді ұйымдастыру негізінде деректерді шапшаң өңдеу режимінің құнын азайтады.

Жұмыс жасау көрсеткіштерінің жақсаруы жүктемені статистикалық және динамикалық теңгеруді жүзеге асыру арқылы қол жетімді болады. Жүктемені басқару алдын-ала орнатылған параметрлерден және жүйелердің ағымдағы жағдайынан тәуелді AS-процестерді (AS-үлгілердің бағдарламалық компоненттерін) жүктеу немесе тоқтатудан тұрады. Қажеттілік кезінде ТРМ желінің осы немесе басқа түйіндерінде AS-процестердің көшірмелерін көбейтуі мүмкін.

Үлестірімді жүйелердің ***администраторы***, ТРМ арқылы, АЖ-ні оңай масштабтау және ақпаратты өңдеу өнімділігін арттыру мүмкіндігіне ие. Мұнда, тігінен және көлденеңінен масштабтаудан басқа, *матрицалық масштабтау* деп аталатын масштабтаумен қамтамасыз етуге болады. Оның мағынасы – гетерогенді есептеуіш ортаның кез-келген нүктесіне қосымша архитектурасына өзгерістер енгізбей, жаңа ортада орындалатын қосымша ресурстар кірістіру. Бұл қосымша серверінің жұмысын тоқтатпай, кез-келген уақытта компьютер немесе ресурстар менеджерін қосуға болатынын білдіреді.

Сонымен қатар, жүйелер администраторы клиент-сервер жүйелерін бағдарламалық қамтамасыз етудің жалпы құнын төмендетуге мүмкіндік алады. Құнын төмендетуге ДҚ серверіне қосылатындар санын азайту арқылы қол жеткізуге болады.

ДҚ серверлері (немесе ДҚБЖ) құны бағдарламаларға бір уақытта қосылулар санынан тәуелді. ДҚ серверіне қосылатындар санын азайту сұраныстарды топтастыру жолымен немесе көптеген ақпарат көздерінен келетін сұраныстар легін бір немесе бірнеше ақпарат көздеріне логикалық түрлендіру арқылы жүзеге асады. ТРМ тиімді жүзеге асуы кезінде құны арзандап, өнімділігі недәуір артады.

4.3. Үлестірімді жүйелерде деректерді басқару

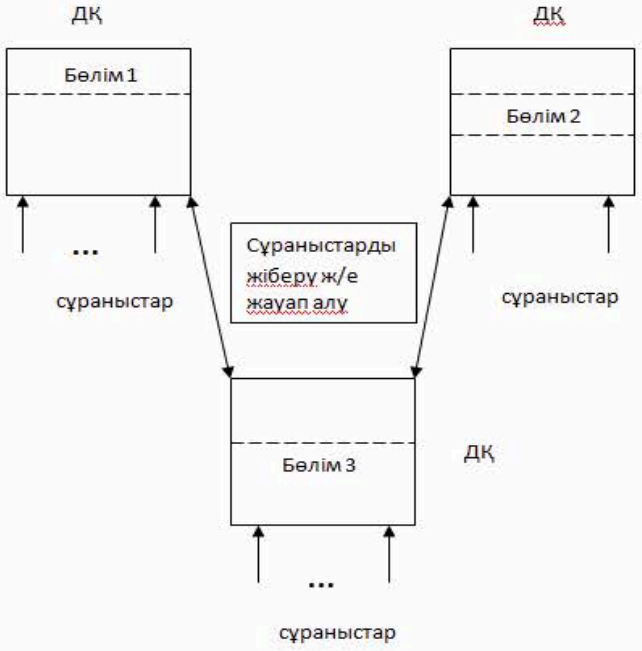
Үлестірімді жүйелерде деректерді басқаруға байланысты келесі екі мәселелер тобы бар: өзгерістер енгізілген ДҚ сәйкестендіруді сүйемелдеу және бірнеше пайдаланушылардың ортақ деректерге бірлесе қатынас құруын қамтамасыз ету.

Өзгерістер енгізілген ДҚ сәйкестендіруді сүйемелдеу

Заманауи үлестірімді жүйелерде ақпарат *орталықтандырылған* немесе *орталықтандырылмаған* түрде сақталуы мүмкін. Бірінші жағдайда барлық соңғы өзгерістерді енгізу бір жерде сақталады. Іс жүзінде ақпарат үлестірімді есептеуіш жүйелердің бірнеше түйіндерінде бір уақытта жиі өзгереді. Бұл жағдайда барлық ақпараттарға енгізілген өзгерістерді бақылау және оны барлық пайдаланушыларға шынайы түрде жеткізу мәселесі пайда болады.

Орталықтандырылмаған ДҚ басқарудың негізгі екі технологиясы бар: *үлестірімді ДҚ* (Distributed Database) және *таралымын көбейту* немесе *ДҚ репликациялау* (Data Replication).

Орнатылған ДҚ желінің әртүрлі түйіндерінде орналасқан, әралуан ДҚБЖ басқаратын бірнеше фрагменттерден тұрады. Үлестірімді ДҚ-на қатынас құратын бағдарламалар және пайдаланушылар тұрғысынан қарағанда соңғысы біртұтас жергілікті ДҚ ретінде қабылданады (0.6-сурет).



4.6-сурет. Үлестірімді ДҚ үлгісі

Үлестірімді ДҚ-ның әрбір бөлігінің қайда орналасқандығы туралы ақпарат және басқа қызметші ақпарат *деректердің жаламдық сөздігінде* сақталады. Жалпы жағдайда бұл сөздік түйіндердің бірінде сақталуы мүмкін немесе үлестірімді болуы мүмкін.

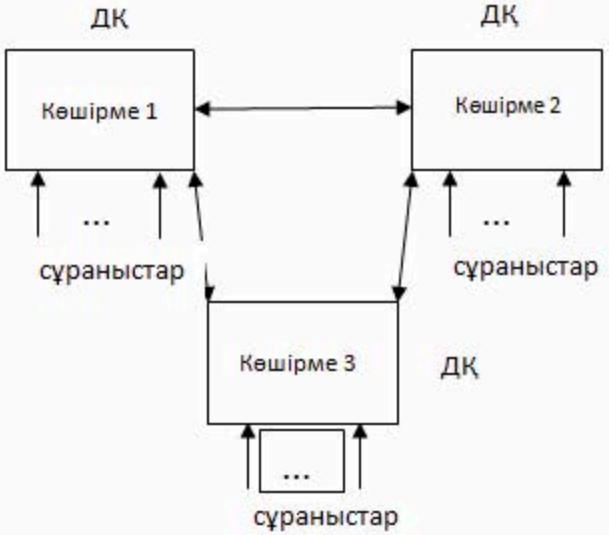
Заманауи жүйелерде үлестірімді ДҚ-на дұрыс қатынас құруды қамтамасыз ету үшін көбінесе *транзакциялардың екіфазалы бекіту* (two-phase commit) хаттамасы (әдісі) қолданылады. Бұл әдістің мағынасы барлық түйіндерінде болатын өзгерістерді екі кезеңмен синхрондаудан тұрады. Бірінші кезеңде желі түйіндеріне өзгерістер енгізу (әзірше қайтарымды) ДҚ-да жүзеге асырылады, ол туралы ескертулер үлестірімді транзакцияларды өңдеуді басқаратын жүйелер компоненттеріне жіберіледі.

Екінші кезеңде, операциялардың орындалу дұрыстығы туралы барлық түйіндерден хабарламалар алған соң (аппараттық-бағдарламалық қамтамасыз етуде тоқтау және істен шығу болмағанын білдіретін), басқарушы компонент барлық түйіндерге өзгерістерді бекіту командасын береді. Бұдан соң транзакция аяқталады, ал оның нәтижесі қайтымсыз болып саналады.

Үлестірімді ДҚ үлгілерінің негізгі ***құндылығы*** барлық түйіндер пайдаланушылары (коммуникациялық құралдар дұрыс жұмыс жасаған кезде) ақпаратты барлық соңғы өзгерістерімен алуы болып табылады. Екінші құндылығы - компьютерлердің сыртқы жадысын үнемдеп қолдану, ол үлкен көлемді ДҚ ұйымдастыруға мүмкіндік береді.

Үлестірімді ДҚ үлгілерінің кемшіліктеріне мыналар жатады: каналдық байланыстардың өнімділігі және сенімділігіне қойылатын қатаң талаптар, сонымен қатар транзакциялардың орындалуының барлық уақытында оларды байланыстыру үшін коммуникациялық және есептеуіш ресурстардың үлкен шығыны. Үлестірімді ДҚ-на қарқынды қатынас құру кезінде, өзара әрекеттесетін түйіндердің саны көп болғанда, байланыстардың жылдамдығы төмен және сенімді емес каналдарында бұл сызба бойынша сұраныстарды өңдеу практикалық тұрғыдан мүмкін емес.

Деректердің таралымын көбейту үлгісінің үлестірімді ДҚ технологиясынан айырмашылықтары олар желі түйіндерінде деректердің қосарланушылығын жобалайды (нақты көшірме құру) (0.7-сурет). Деректер әрқашан жергілікті деректер сияқты өңделеді. Асинхронды режимде бір-біріне ұқсастықты сүйемелдеуді *репликатор* (replicator) деп аталатын жүйелер компоненті қамтамасыз етеді. Сонымен қатар, желі түйіндері арасында ішінара және топтық өзгерістер енгізлуі мүмкін. Қайсыбір уақыт аралығында ДҚ көшірмелері бір-бірінен өзгешеленуі мүмкін.



4.7-сурет. ДҚ таралымын көбейту үлгісі

ДҚ таралымын көбейту үлгісінің негізгі ***құндылығына*** (алдыңғы үлгілермен салыстырғанда) жатады: деректерге қатынас құрудың неғұрлым жоғары жылдамдығы, өйткені олар әрқашан түйінде орналасқан; ақпараттардың ағынын байланыс каналдары бойынша жылдам жөнелту, себебі жүзеге асатын деректерге қатынас құрудың операцияларының барлығы емес, ДҚ-дағы өзгерістер ғана; үлестірімді деректерге қатынас құру механизмдерінің сенімділігі жоғары, себебі байланыстардың үзілуі жүйелердің жұмысына әсер етпейді.

ДҚ таралымын көбейту үлгісінің негізгі ***кемшілігі:*** қайсыбір уақыт аралығында ДҚ көшірмесінің "қайшылығы". Егер аталған кемшілік қолданбалы есептер үшін қауіпті емес болса, онда ДҚ таралымын көбейту сызбасының болғаны дұрыс.

Ортақ деректерге қатынас құру

ДҚ-на қызмет көрсету кезінде ортақ деректерге қатынас құру құралдарын басқаруды қамтамасыз ету ең болмағанда екі негізгі қатынас құру әдісін қамтуы керек: монополиялық және коллективтік. Әртүрлі жүйелерде негізгі объектілер арасында қатынас құру толықтай ДҚ-да, ішінара кестелерде, жазбаларда, өрістер жазбаларында болуы мүмкін. Объектілерге қатынас құру және өңдеуге мүмкіндік беретін ДҚБЖ-да, сонымен қатар есеп берулер және экрандық формалар, сұраныстар және бағдарламалар болуы мүмкін.

Монополиялық қатынас құру әдетте екі жағдайда қолданылады:

- біріншіден, объектілерге басқа пайдаланушылар тарапынан қатынас құруды болдырмау (мысалы, құпия ақпараттармен жасау кезінде) талап етілсе;
- екіншіден, ДҚ-мен операцияларды *жауаптылар* жүзеге асырған кезде, басқа операциялардың орындалуына жол бермеу, мысалы, ДҚ құрылымын өзгерту.

Бірінші жағдайда пайдаланушы ДҚБЖ сұқбаттасу құралдары немесе қолданбалы бағдарламалар көмегімен *шектеу* қояды. Екінші жағдайда пайдаланушы шектеу қоюы немесе ДҚБЖ-не сенім артуы мүмкін. Соңғысы әдетте, қажет болса, автоматты түрде айқын емес (пайдаланушыны немесе қосымшаны ескертпестен) шектеу қояды.

Коллективтік қатынас құру режимінде, қағидаға сәйкес, қолданылатын объектілерге толық шектеу қойылмайды. Коллективтік қатынас құру, мысалы, бір уақытта кестені көру мүмкіндігі. Объектілерге коллективтік қатынас құру кезінде монополиялық қатынас құруға қол жеткізу әрекеттеріне жол берілмейді. Мысалы, бір немесе бірнеше пайдаланушылар бір кестені қарастырған кезде басқа пайдаланушы оны өшіре алмайды.

ДҚБЖ-да коллективтік қатынас құруды ұйымдастыру үшін ***шектеу қою механизмі*** қолданылады. Шектеу қоюдың мағынасы – қандай да бір операциялардың орындалуы кезінде ДҚ-на басқа тұтынушылар тарапынан қатынас құруға уақытша тыйым салынады немесе шектеледі. Мысалы, кестелердің көшірмесін алу кезінде оған өзгерістер енгізуге шектеу қойылады, бірақ оның құрамын көруге рұқсат етіледі.

Шектеу қоюдың қайсыбір типтік жинағын қарастырайық. Нақты бағдарламаларда объектілерге шектеу қою сызбасы біз сипаттағаннан өзгеше. Шектеу қоюдың төрт түрін қарастырайық:

- толық шектеу қою;
- жазбаларға шектеу қою;
- жазбаларға шектеу қоюдан сақтандыру;
- толық шектеу қоюдан сақтандыру.

Толық шектеу қою. Негізгі объектілермен (кестелермен, есеп берулермен және экрандық формалармен) орындалатын барлық операцияларға толық шектеу қоюды білдіреді. Шектеу қоюдың бұл түрі әдетте кестелер құрылымын өзгерту кезінде қолданылады.

Жазбаларға шектеу қою. Кестені қолдануға болады, бірақ оның құрылымына немесе құрамына өзгерістер енгізуге болмайтын жағдайда қолданылады. Мысалы, екі кестедегі деректерді біріктіру операциясының орындалуы кезінде.

Жазбаларға шектеу қоюдан сақтандыру. Объектіні басқа операциялар тарапынан шектеу қою немесе жазбаларға шектеу қоюдан сақтайды. Шектеу қоюдың бұл түрі объектіні алдымен қолданушыға объектімен іс-әрекеттер жасауын аяқтауға мүмкіндік береді. Жазбаларға шектеу қоюдан сақтандыру жазбаларға шектеу қоюмен ұқсас, сонымен қатар толық шектеуден сақтайды (ары қарай караңыз). Бұл шектеуді пайдалану қажеттілігі мысалы бірнеше пайдаланушылардың кестелерді бірлесе өңдеу режимі болып табылады.

Толық шектеу қоюдан сақтандыру. Объектіні басқа операциялар тарапынан толық шектеу қоюдан ғана сақтайды. Ол объектілерді бірлесе пайдаланудың ең жоғары деңгейін қамтамасыз етеді. Мұндай шектеу қою мүмкіндігі, мысалы, бір уақытта бірнеше пайдаланушылардың бір кестені көруін қамтамасыз ету үшін қолданылады. Бір кестемен жұмыс жасайтын пайдаланушылар тобына мұндай шектеу қою ешкімге ортақ кестенің құрылымын өзгертуге мүмкіндік бермейді.

Қайсыбір объектімен орындалатын операциялар аяқталмаған кезде және осы объектімен жаңа операцияларды орындау сұранысы түскен кезде осы операциялар қатар орындалу арқылы жүзеге асады. Жоғарыда аталған шектеу қоюдың төрт түрі келесі ережемен жүзеге асады:

- толық шектеу қою кезінде шектеу қоюдың ең болмағанда бір түрі келтірілген операциялар орындалмайды (толық шектеу қою басқа шектеу қоюлармен үйлесімді емес);
- жазбаларға шектеу қою ұқсас шектеу қоюлармен үйлесімді және толық шектеу қоюдан сақтайды;
- жазбаларға шектеу қоюдан сақтандыру шектеу қоюдың басқа екі түрімен үйлесімді;
- толық шектеу қоюдан сақтандыру толық шектеу қоюдан басқа барлық шектеу қоюлармен үйлесімді.

Әдетте ДҚБЖ-да ДҚ-мен орындалатын операциялардың әрқайсысы объектіге операция қоятын шектеу қоюдың бір түрімен сәйкес келеді. Заманауи ДҚБЖ-лармен белсенді режимде жұмыс жасайтын пайдаланушыларға шектеу қою механизмдерінің барлығын есте сақтау қажет емес, себебі жүйелер талап етілген барлық жағдайларда автоматты шектеу қоюды жүзеге асырады. Сонымен қатар, жүйелер барлық пайдаланушыларға объектілерге неғұрлым еркін қатынас құруды ұсынуға ұмтылады. Қажет кезінде пайдаланушы және бағдарлама құрушы шектеу қоюды анықтау үшін командаларды немесе тілдер құралдарын пайдалана алады. Мысалы, Paradox ДҚБЖ-да кестелерді өңдеу кезінде бөлек жазбаларға шектеу қою үшін Record Lock командасы қолданылады.

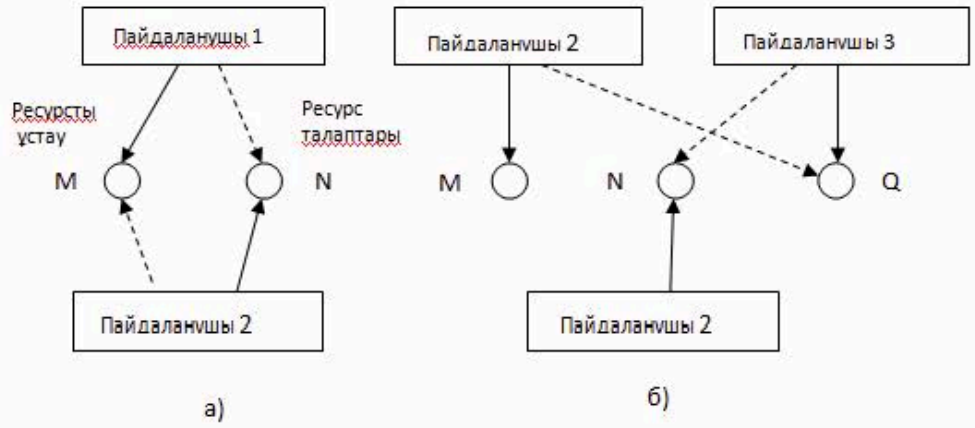
Тұйықтар

Егер объектілерді бірлесе қолдану кезінде қатынас құруды басқармасақ, онда ресурстарды пайдаланушылар арасында тұйыққа тірелетін жағдайлар (жік түсу, “қауіпті бірігулер” немесе шектеу қою) пайда болуы мүмкін. Объектілерге қатынас құруды бақылау мақсатында шектеу қою түсінігінің (біз осы терминді пайдаланамыз) жағдайдың тұйыққа тірелуіне шектеу қою мағынасынан айырмашылығын білу керек.

Тұйыққа тірелудің екі негізгі түрі бар: *өзара* (deadlock) және *біржақты* (livelock).

Өзара тұйыққа тірелудің қарапайым жағдайы екі пайдаланушылардың әрқайсысы басқа пайдаланушы қамтитын деректерді қамтуға тырысу болып табылады (0.8а-сурет). Бұл жағдайда 1-пайдаланушы N ресурстың босауын күтеді, осы уақытта қалай 2-

пайдаланушы М ресурстың бос болуын күтеді. Яғни, олардың ешқайсысы жұмысты жалғастыра алмайды.



4.8-сурет. Үлестірімді ДҚ-да өзара тұйыққа тірелудің мысалы

Шындығында, бірнеше ресурстарға үш және одан да көп пайдаланушылар қатынас құрған кезде неғұрлым күрделі жағдайлардың пайда болуы мүмкін. Мұндай жағдайдың бір мысалы 0.8б-суретте келтірілген.

Біржақты тұйыққа тірелу қайсыбір ресурсқа монополиялық қатынас құру талаптар орындалғаннан кейін ғана қолжетімді болып, орындала алатын жағдайда пайда болады.

Үлестірімді ДҚ басқару жүйелері, әлбетте, шиеленістерді болдырмайтын, сонымен қатар пайда болған шиеленістерді шешетін сәйкес құралдарға ие болуы керек. Неғұрлым күрделі мәселелердің бірі - егер біртекті емес жүйелерде қайсыбір бағдарлама шиеленістердің болуы туралы сигналдарды (ескертулерді) өңдемесе немесе қате өңдеген кезде шиеленістерді жою. Сонымен қатар, деректердің үлестірімді ДҚ-да тұтастықты және нақтылықты сақтау ғана маңызды емес, кей жағдайда пайдаланушылар және бағдарламалардың жұмысын тоқтатын есептеу процесін қалпына келтіру де маңызды.

Үлестірімді ортада пайдаланушылар және бағдарлама құрушылар тұйыққа тірелу туралы сигналдарды өңдеуді қарастыруы керек.

4.4. Жергілікті желідегі ақпараттық жүйелер

Жергілікті есептеу желісі пайдаланушыға, ең алдымен, аппараттық ресурстармен топтық жұмыс жасау және бірлесе пайдалануды неғұрлым тиімді ұйымдастыруға мүмкіндік береді: принтерлер, факстер, модемдер, сканерлер, дискілер және т.б., сонымен қатар бағдарламалық-ақпараттық ресурстар және деректер қоры.

ЖЕЖ-де негізгі *деректермен жұмыс жасауды ұйымдастыру сызбасын* қарастырайық .

ЖЕЖ-де ақпараттық жүйелер тұрғызу сызбасы қолданылатын ЖОЖ мүмкіндігінен тәуелді.

Заманауи ЖЕЖ негізгі сервисі оларды басқару түрінен (орталықтандырылған немесе біррангілі) тәуелсіз, бір компьютердің (компьютер-клиент) басқа компьютердің (компьютерд-сервер) дискілеріне, каталогтарына (бумалар) және файлдарына қатынас құруының көрінісі болып табылады. Ендеше, ЖОЖ сыртқы файлға қатынас құру кезінде алдымен компьютердің жедел жадысына желі бойынша файл (файл бөлігін) жібереді, ал онымен жұмыс аяқталған кезде, қажеттілік болса, жаңартылған нұсқаны

жүктейді. Сонымен қатар, компьютерге басқа компьютерлердің бағдарламаларын жүктеу мүмкіндігі пайдалы функция болып табылады.

Бұл мүмкіндіктердің мысалы келесі желілік ОЖ келтірілген: Novell NetWare 3.x (4.x), Windows NT және Windows 95/98. Неғұрлым әлсіз желілік дестелер және утилиттер ақпараттарға қатынас құруды бағдарламаларды автоматты түрде жүктемейақ ұсына алады. Бұл жағдайда пайдаланушы өзіне басқа компьютердің бағдарламалары және файлдарының көшірмесін жасап алуы керек, содан соң бағдарламаларды жүктейді.

Дербес компьютердегі сияқты, ЖЕЖ-де пайдаланушы ДҚ құралдары көмегімен қайсыбір ДҚБЖ немесе қосымшалармен жұмысты басқаруы мүмкін. Жалпы жағдайда, қосымшалар ДҚБЖ немесе оның ядросының басқаруымен орындала алады, немесе дербес бағдарлама ретінде тәуелсіз болады және орындалады. Ыңғайлылық үшін, бұдан былай ДҚБЖ ретінде осы нұсқалардың кез-келгенін қарастырамыз.

Жергілікті желіде дербес ЭЕМ ақпараттық жүйелер құрудың келесі үш нұсқасын бөліп көрсетеді:

- файл-сервер типті;
- клиент-сервер типті;
- intranet технологиясына негізделген;

Файл-сервер типті ақпараттық жүйелерді екі түрлі әдіспен тұрғызуға болады:

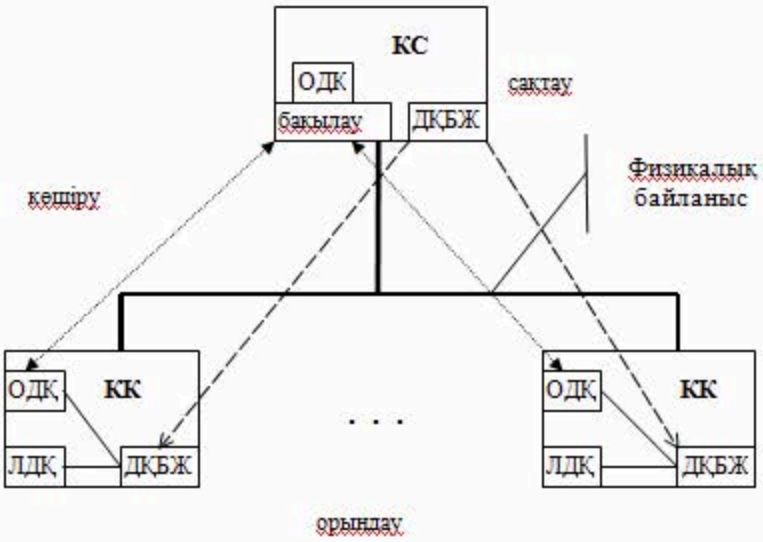
- жекелеген машинада қолдануға арналған *желілік емес ДҚБЖ*;
- ЖЕЖ-де қолдануға арналған *желілік ДҚБЖ*.

Желілік ДҚБЖ дегеніміз желіде қолдануға негізделген, ерікті деректер үлгісі (желілік болуы міндетті емес) бар жүйелер.

Желілік емес ДҚБЖ бағдарламалары және онда қолданылатын деректер компьютер-серверде (КС) және компьютер-клиентте (КК) сақталуы мүмкін.

Желілік емес ДҚБЖ жүктеу және жұмыс жасаудың компьютер-клиентте (КК) сақталатын және жергілікті деректермен жұмыс жасайтын жекелеген ДЭЕМ жұмысының кәдімгі режимінен айырмашылығы жоқ. Егер қолданылатын деректер компьютер-серверде (КС) сақталатын болса, онда желілік ОЖ-нің файлдық жүйелері ДҚБЖ үшін шеткері компьютердің қажетті файлын жүктеуді орындайды. Ескере кетейік, әрбір желілік емес ДҚБЖ кез-келген желілік ОЖ ортасында қателіксіз жұмыс жасамауы мүмкін.

Егер желілік емес ДҚБЖ-ын бірнеше пайдаланушылар қолданса, онда дискілік жадыны тиімді пайдалану мақсатында оның бағдарламаларын, сонымен қатар ДҚ немесе оның бөлімін КС-де сақтау тиімді. Сақталатын деректер қорын *орталық ДҚ* (ОДҚ), ал компьютердің деректер қорында сақталатындарын – *жергілікті ДҚ* (ЖДҚ) деп атаймыз. ДҚБЖ жүктеу кезінде әрбір КК-на ДҚБЖ-ның негізгі бағдарламаларының және ОДҚ-ның бір немесе бірнеше файлдарының толық көшірмесі жіберіледі (0.9-сурет). Жұмыс жасау аяқталғаннан соң ОДҚ файлы деректерді келістіру үшін КК-дан кері қарай КС-ға жөнелтіледі.

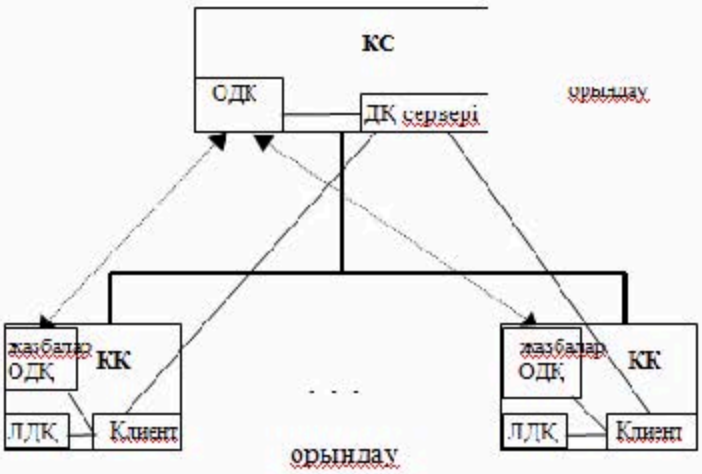


4.9-сурет. Желілік емес ДҚБЖ-мен файл-сервер типті жүйелер

Бұл нұсқаның ең басты *кемшілігі* - желілік емес ДҚБЖ қолдануы болып табылады, ол бір ДҚ-мен бірнеше пайдаланушылар бір мезетте жұмыс жасаған кезде деректер тұтастығының бұзылуына апаруы мүмкін. Себебі әрбір ДҚБЖ көшірмесі басқа ДҚБЖ көшірмесінің қалай жұмыс жасайтынын "білмей", шиеленістерді шешу үшін ешқандай қадамдар жасай алмайды. Сонымен қатар, бірдей файлдарды оқу-жазу қарапайым операцияларын, қағидаға сәйкес, желілік ОЖ бақылайды.

Желілік емес ДҚБЖ мысалдары: dBase Ш Plus, dBase IY және FoxBase жүйелерінің алғашқы нұсқалары.

Желілік ДҚБЖ-де аталған кемшілік жоқ, өйткені оларда "бақталастықты бақылау" (concurrency control) қарастырылған. Бақылау құралдары деректерге қатынас құруды үйлестіреді, мысалы, файлдарға, жазбаларға және тіпті жазбалар өрістері бөліміне шектеу қою (0.10-сурет).



4.10-сурет. Желілік ДҚБЖ-мен файл-сервер типті жүйелер

Желілік ДҚБЖ-да ДҚ файлдарын топтық қолдануда барлық ақпараттарды өңдеу жұмыстары КК-да жүзеге асады, ал КС функциялары үлкен дискілік жадыда ұсынылады. Мұндай тәсілдеме тиімді емес, өйткені ақпаратты өңдеу процесінің

жылдамдығын арттыру үшін КК-нің жедел жадыда үлкен көлемі болуы және жылдам әрекет жасауы керек. Сонымен қатар, ДҚ файлдарының көшірмесін жөнелту және байланыстар сызығы бойынша шектеу қоюды басқару командалары ішкі жүйелерге деректерді жөнелтудің жүктемесін анағұрлым арттырады, ал ол өз кезегінде желінің жалпы өнімділігін азайтады.

Желілік ДҚБЖ мысалдары: Windows үшін FoxPro 2.5, dBase IY, DOS үшін Paradox 3.5.

Ақпараттық клиент-сервер жүйелерінің файл-сервер типті жүйелерден айырмашылығы: ең алдымен, ДҚБЖ бағдарламалары, функционалдық тұрғыдан, *сервер* және *клиент* деп аталатын екі бөлімге бөлінген. Клиент және сервер бөлімдері арасында әртүрлі функциялар нұсқаларының орналасуы мүмкін (0-бөлім).

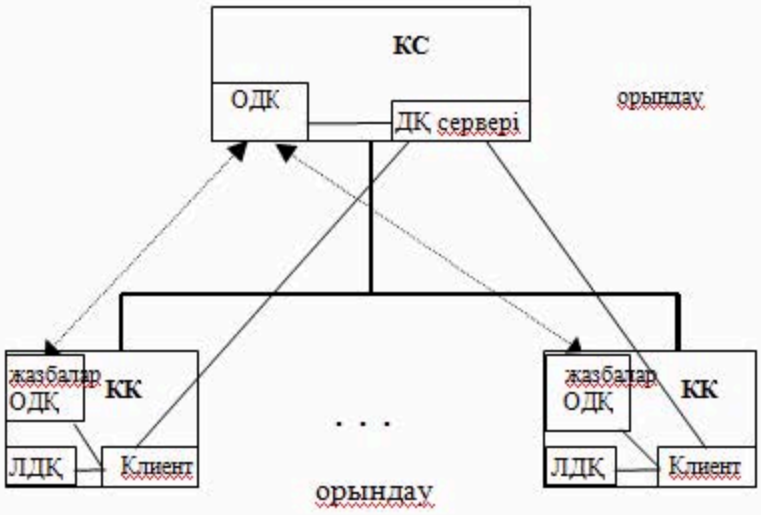
Клиент, немесе *фронтальды бағдарлама*, пайдаланушымен интерфейске жауап береді. Ол үшін оның сұраныстарын серверлік бөлімдерге сұраныстар командаларына түрлендіреді, ал нәтижелерді алғаннан соң пайдаланушы үшін ақпараттарды кері оқу және бейнелеуді орындайды.

Клиент ролін пайдаланушы (нақты қолданбалы есепті шешу үшін құрылған) немесе дайын бағдарлама атқарады, оның серверлік бағдарламалармен интерфейсі болуы керек. Дайын клиенттік бағдарламалар ретінде мәтіндік процессорлар, кестелік процессорлары және тіпті ДҚБЖ (мысалы, Access, FoxPro және Paradox) қолданылуы мүмкін.

Қордағы деректерді басқару және қорғау функцияларын атқаратын негізгі бағдарламалар ***сервер*** болып табылады. Сервердің функцияларын шақыру SQL тілінде орындалады, нақтырақ айтқанда жиі солай жүзеге асады, оны ***SQL-сервер*** деп атайды.

Сервер ретінде кәсіби реляциялық ДҚБЖ (мысалы, Informix 7.x және Sybase System 10) ядросы немесе қайсыбір SQL-сервер (мысалы, Novell NetWare SQL және Microsoft SQL Server) қолданылуы мүмкін.

Ақпараттық клиент-сервер жүйелері құрылымын қысқаша түрде 0.11-суреттегідей көрсетуге болады.



4.11-сурет. Ақпараттық клиент-сервер жүйелері

Сұраныстарды қалыптастыру бойынша ақпаратты өңдеудің негізгі бөлімі, есеп берулер, деректердің пайдаланушыға ыңғайлы түрде бейнеленуі және т.б. КК-да орындалады. ДҚ файлдарының толық көшірмесі КС-дан КК-ке және керісінше жіберілмейді, себебі толыққанды өзара әрекеті ұйымдастыру үшін, қағидаға сәйкес,

берілген уақыт аралығында КК-да қажетті ДҚ жазбасы болуы жеткілікті. Бұлардың барлығы желіге қосылу құнын недәуір азайтады, КК ресурстарына қойылатын талаптарды кемітеді, неғұрлым тиімді және сенімді ақпараттық жүйелер құруға мүмкіндік береді.

Соңғы уақытта компьютер-серверде, өз деректерінен басқа, деректерді өңдеу және сұраныстар бағдарламалары сақталады. Бұл деректерді өңдеу (бағдарламаны өңдеу немесе деректермен сұранысты қатар орындау) жылдамдығын арттырады, сонымен қатар бір орында (компьютер-серверде) бағдарламалар және сұраныстарды ортақ қолдану, сақтау және басқару тиімділігін береді.

Компьютер-серверде сақталатын бағдарламалар (процедуралар) деректерін өңдеу ***сақталатын процедуралар*** деп аталады.

Сақталатын процедуралардың бір түрі ***триггер*** деп аталады. Триггер (триггерлік процедура) ДҚ-да қандай да бір жағдайлар пайда болған кезде автоматты түрде шақырылады. Мынадай жағдайлар болуы мүмкін: операциялары кірістіру, жекелеген жазбаларды, жазбалар бағаналары және өрістерін жаңарту және өшіру және тағы басқалар. Триггер мысалы, ДҚ (жазбалар саны) шекті мәнінің мөлшеріне жетіп, электрондық почта бойынша хабарламалар жіберу процесін жүктейтін бағдарлама болып табылады.

Сервердің ДҚ-да қайсыбір жүйелерді және ***сақталатын командалар*** деп аталатын сұраныстардың өзін сақтауға болады. Сақталатын командалар жиынтығы – бұл SQL-сұраныс компиляциясы нәтижесінде алынатын командалар жиынтығы. Сақталатын командалар сәйкес SQL-сұраныстарға қарағанда анағұрлым жылдам орындалады. Жылдамдық артуының негізгі себебі: сұраныстардың синтаксистік жиынтығы не талап етсе сақталатын командалар соларды ғана орындайды. Сұраныстардың орындалу жылдамдығын арттыру ДҚ сервері сұраныстар командаларының кодын ғана емес, сақталған кодтың тиімділігін арттырған кезде жүзеге асуы мүмкін.

Сақталатын процедуралар және командалармен ***курсор*** түсінігі байланысты, мұның әдеттегі курсор түсінігінен айырмашылығы - монитор экранында ағымдағы орынды көрсетуші ретінде қолданылады. Әртүрлі ДҚБЖ – бұл ұқсас, бірақ айырмашылықтары бар ұғым. Неғұрлым кеңінен бұл ұғым SQLBase ДҚБЖ-да сипатталады. Мұнда курсор келесілерді білдіруі мүмкін:

- пайдаланушының ДҚБЖ-мен байланыстар кезеңінің идентификаторы;
- сақталатын командалар және процедуралар идентификаторы;
- нәтижелік жиын идентификаторы;
- клиент қосымшаларымен өңделетін нәтижелік жиында ағымдағы жолдар көрсеткіші.

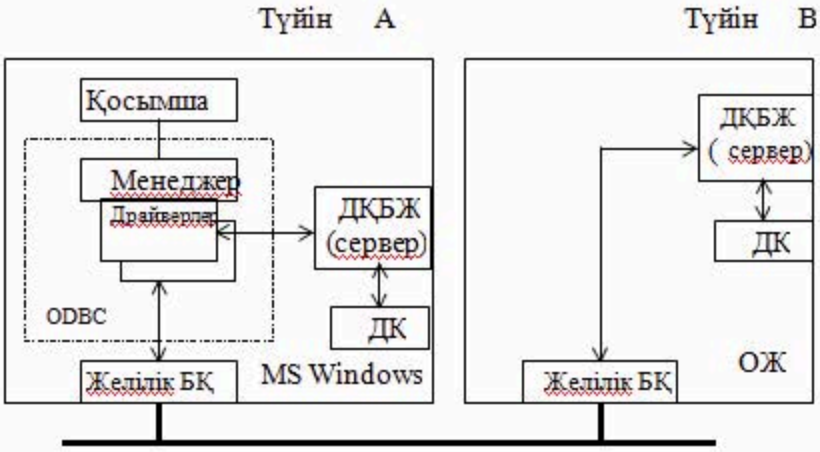
Сервер бағдарламалары (негізгі, сақталатын процедуралар мен триггерлер) қарапайым бағдарламалар ретінде (Windows 95/98) немесе желілік ОЖ арнайы жүктелетін модульдері (NLM-модули желіге Novell) ретінде орындалуы мүмкін. Клиент бағдарламалары жалпы жағдайда КС-да немесе КК-да, немесе екеуінде де сақталады.

Қазіргі уақытта бағдарламалық өнімдер арасында клиент-сервер жүйелері типті әмбебап (әртүрлі ДҚ серверлерімен жұмыс жасай алатын) өңдеу құралдары өте көп, олар: Delphi (Borland), Power Builder (Powersoft), ERwin (LogicWorks), Visual Basic (Microsoft), CA-Visual Objects (Computer Associates), SQL Windows (Gupta) және басқалар. Сонымен қатар, ДҚБЖ ретінде анықталған өңдеу құралдары (мысалы, үшін Oracle 7 – Designer/2000) бар. Барлық ұқсас құралдар, қағидаға сәйкес, CASE-жүйелерге жатады (7-бөлім).

Ақпараттық клиент-сервер жүйелерін тұрғызған кезде ДҚБЖ немесе қосымша тарапынан, бір ортада құрылған басқа ДҚБЖ деректеріне қатынас құру мәселесі пайда болады. Windows ортасында бұл мәселе Microsoft фирмасының ODBC (Open Database Connectivity – ашық деректер қоры үйлесімділігі) стандартты интерфейсі көмегімен

шешіледі. Оның негізгі тағайындалуы әртүрлі өндірушілердің жергілікті және шеткері деректер қорына жүйеленген қатынас құруды қамтамасыз ету болып табылады.

ODBC көмегімен деректер қорына қатынас құру сызбасы 0.12-суретте көрсетілген. Деректерге қатынас құру ODBC интерфейсінің стандартты функцияларын SQL тілінде шақыру арқылы жүзеге асады. Компьютер-клиентте, сонымен қатар, ODBC интерфейсті MS Windows операциялық жүйесі жұмыс жасауы керек.



4.12-сурет. ДҚ-на ODBC көмегімен қатынас құру сызбасы

Қосымшаның деректермен өзара әрекеті ДҚБЖ деректерінің форматына сәйкес қажетті драйверді іске қосатын драйверлер менеджері (диспетчер) көмегімен жүзеге асады. ДҚБЖ драйвері, желілік құралдарды пайдаланып, қағидаға сәйкес, нақты ДҚБЖ коммуникациялық модульдерін ДҚБЖ сервері SQL-операторына жібереді. Серверде сұраныстардың орындалу нәтижелері кері қарай қосымшаларға жіберіледі.

ДҚБЖ жұмыс жасауының қарастырылған сызбасы қосымшаларды тұрғызудың неғұрлым типтік нұсқаларына қатысты.

4.5. Internet және intranet-тегі ақпараттық жүйелер

Ақпараттарды өңдеу ортасында Internet желісінің жергілікті желіден, сонымен қатар, бөлек компьютерде ақпаратты өңдеуден елеулі айырмашылығы бар. Олардың неғұрлым маңыздыларын атап өтейік.

1. Коммуникациялық желілердің ұзындығының үлкендігі, ол уақытша ақпараттар алмасуға кері әсер етеді. Сонымен қатар, үлкен қашықтық бір компьютердің бағдарламаларын басқа компьютерге жүктеу маңызын жояды және жергілікті желі ДҚБЖ ретінде үлкен көлемді деректерді нақты уақыт аралығында жөнелтілуіне мүмкіндік береді.

2. АЖ үлестірімді элементтерінің өзара әрекеті *дестелермен* немесе *хабарламалармен* алмасу арқылы жүзеге асады. АЖ ішінара бір немесе әртүрлі өндірушілердің бағдарламалық компоненті болуы мүмкін. Соңғы жағдайда желілік хаттама және SQL тілі стандарттарын сүйемелдеу мәселелерін шешу маңызды роль атқарады.

3. Internet желісінің басқа ауқымды желілерден айырмашылығы – ол көлемі бойынша барлық басқа желілерден (басқа желілерді біріктіреді) үлкен және оны ұйымдастыру принципі желіде деректер қорын қолдануға елеулі ықпал етеді.

ДҚ пайдаланудың үлгілерін және механизмдерін қарастырмастан бұрын Internet ұғымына қысқаша сипаттама берейік.

Internet сипаттамасы

Internet желісіне қосылған кезде пайдаланушыларға ұсынылатын қызметтердің (сервистер) негізгі түрлері:

- электрондық почта (E-mail);
- телеконференциялар (UseNet);
- шеткері терминалдарды эмуляциялау жүйелері (TelNet);
- екілік файлдарды іздеу және жөнелту (FTP);
- жүйелер мәзірі көмегімен мәтіндік файлдарды іздеу және жөнелту (Gopher);
- гипермәтіндік сілтеме көмегімен құжаттарды іздеу және жөнелту (WWW немесе "дүниежүзілік өрмек").

Осы әдістерді құру және дамуы Internet тарихымен байланысты. Олардың әрқайсысы ақпараттармен алмасу хаттамаларын ұйымдастыруда өздерінің мүмкіндіктері мен өзгешеліктерімен сипатталады. Хаттама дегеніміз, жалпы жағдайда, өзара байланысқан жүйелер немесе желі объектілерінің жұмысын реттеуші нұсқаулардың жиынтығы.

Электрондық почта (E-mail) – Internet желісіне қатынас құрудың неғұрлым қарапайым және қолжетімді тәсілі. Ол жер шарының кез-келген нүктесіне электрондық почта адресі бойынша аз уақыт аралығында кез-келген уақыт мезгілінде әртүрлі типті файлдарды (мәтіндер, суреттер, дыбыстық файлдар) жөнелтуді орындауға мүмкіндік береді. Хабарламаларды жөнелту үшін алушының электрондық адресін білу қажет. Электрондық почта жұмысы желі бойынша ақпараттарды бір почталық серверден басқасына, хабарлама алушыға жеткенге дейін, тізбектей жөнелтуге негізделген. Электрондық почтаның құндылығы: жоғары жылдамдығы және құнының арзандығы. Электрондық почтаның кемшілігі: жіберілетін файлдар көлемінің шектеулілігі.

Телеконференциялар жүйелері UseNet мәтіндік ақпараттармен алмасу жүйелері ретінде құрылған. Ол барлық Internet пайдаланушыларға телеконференция деп аталатын барлық мүмкін мәселелерді талқылайтын топтық пікірталастарға қатысуға мүмкіндік береді. Қазір әлемде 10 мыңнан астам телеконференциялар бар. Телеконференцияларға жіберілетін ақпараттар осы конференцияға қосылған кез-келген Internet пайдаланушыға қолжетімді. Қазіргі уақытта телеконференциялар кез-келген типті файлдарды жөнелте алады. Телеконференциялармен жұмыс жасау үшін Web-құжаттарды көру және өңдеу бағдарламалау құралдары неғұрлым жиі қолданылады.

TelNet – бұл бір компьютерге басқа (шалғайдағы) компьютердің қорларын қолдануға мүмкіндік беретін хаттама. Басқаша айтқанда – бұл желіге шалғайдағы терминальдық қатынас құру хаттамасы.

FTP (File Transfer Protocol) – бұл желіге қосылған екі компьютердің арасында еркін форматтағы файлдарды жөнелтуге мүмкіндік беретін хаттама. FTP бағдарламалық қамтамасыз етуі "клиент-сервер" архитектурасы бойынша құрылған және екі бөлімге бөлінген: серверлік (FTP-сервер) және клиенттік. FTP-клиент, жалпы жағдайда, пайдаланушыларға FTP-сервердің файлдық жүйелерін көруге және олардың арасында файлдармен (өзге компьютерден файлдар жүктеу, шалғайдағы компьютердің файлдарының атауын өзгерту және өшіру) алмасуға мүмкіндік береді. Бұл хаттаманың құндылығы кез-келген типті файлдарды, сонымен қатар, орындалатын бағдарламаларды жөнелту мүмкіндігі болып табылады. FTP хаттаманың кемшілігі ретінде ақпараттардың орналасқан жерін (FTP-адресі) білу қажеттілігін атауға болады.

Gopher хаттамасы және оны бағдарламалық қамтамасыз етуді жүзеге асырушылар пайдаланушыларға ақпараттық қорлармен олардың орналасқан орнын алдын-ала білмей-ақ жұмыс жасауға мүмкіндік береді. Бұл хаттама бойынша жұмыс жасауды

бастау үшін бір Gopher-сервердің адресін білу жеткілікті. Бұдан соң жұмыс қарапайым және түсінікті мәзір түрінде берілген командаларды таңдауға негізделеді. Сонымен қатар, бір сервердің мәзір пункты басқа серверлердің мәзіріне сілтемелерден тұруы мүмкін, ол Internet желісіндегі қажетті ақпараттарды іздеуді жеңілдетеді. Gopher жүйелерімен жұмыс жасау кезінде бағдарлама-клиент Gopher-сервермен тұрақты жалғастырылады, ол желілік ресурстарды үнемдеуге мүмкіндік береді.

WWW (World Wide Web – дүниежүзілік өрмек) желілік ресурстарды ұйымдастыру құралдарының ішіндегі ең танымал және заманауи болып саналады. Ол ақпараттардың гипермәтіндік көрінісі негізінде құрылған.

Гипермәтіндік құжат (***гипермәтін***) ерікті құжаттарға, сонымен қатар, құжат мәтіндері үзіндісіне сілтемелер болып саналады. Гипермәтіндік құжат стандартты HTML (HyperText Markup Language – гипермәтінді белгілеу тілі) тілде дайындалады. Ол парақтардан (web-беттер) тұрады, оларға қатынас құру гипермәтінді жөнелту хаттамасына (HyperText Transfer Prococol, HTTP) негізделген.

Кез-келген мәтіндік редакторда көруге және өңдеуге мүмкіндік беретін ASCII-файл HTML-құжат болып саналады. Әдеттегі мәтіндік файлдан айырмашылықтары: онда арнайы командалар – ***тэгтер*** бар, олар құжатты форматтау ережелерін көрсетеді. Тэгтер көмегімен құжаттың әртүрлі элементтері суреттеледі: тақырыптар, азат жолдар (параграфтар), тізімдер, сілтемелер, формалар және т.б.

Гипермәтіннің қарапайым мысалы кітап болып табылады, мазмұнын парақ нөмері түріндегі сілтемелер қамтиды, сонымен қатар, кітапта қолданылатын ақпараттардың түпнұсқасына сыртқы сілтемелер бар.

Құжат фрагменті ақпарат түрінде әдеттегі мәтінді, графикалық бейнені, дыбысты және қозғалыстағы бейнені (анимация) қамтуы мүмкін. Мәтіндік емес құжаттары бар гипермәтін әдетте ***гипермедиа*** деп аталады.

Гипермәтіннің маңызды қасиеті онда территориялық шеткері орналасқан компьютерлердің құжаттарына сілтемелердің бар болуы болып табылады. Құжаттар әртүрлі адамдармен құрылуы және редакциялануы мүмкін. Өзара байланысқан құжаттардың барлық жиынтығы өте үлкен “өрмек” құрады. Бұл үлгі бізді қоршаған шексіз ақпараттық кеңістікке ұқсас, байланыстың қатаң иерархиясы болмайды, ал байланыс жиынының басы және аяғы жоқ.

Internet желісінде жұмыс жасау TCP/IP (Transmission Control Protocol/Internet Protocol – деректерді жөнелтуді басқару хаттамасы / Internet хаттамасы) хаттамасын қолдануға негізделген, онда деректерді жөнелту үшін ауқымды желі және көптеген жергілікті желілер қолданылады. TCP/IP көбінесе транспорттық және желілік деңгейдегі OSI (4.1-бөлім) үлгілері функцияларын жүзеге асырады. Ол коммуникациялық хаттамалар жиынтығы болып саналады, оларды тағайындалуы бойынша келесі топтарға бөлуге болады:

- екі компьютердің арасында деректерді жөнелтуді басқаруға арналған транспорттық хаттамалар;
- деректердің адресін өңдейтін және хабарлама алушыға ең қысқа қолжетімді жолды анықтайтын бағдарлау хаттамалары;
- компьютерлерді бірегей нөмері немесе атауы бойынша сәйкестендіруге арналған желілік адресті сүйемелдеу хаттамалары;
- барлық мүмкін желілік қызметтерге қатынас құруды жүзеге асыратын қолданбалы хаттамалар;
- желі жағдайы, сонымен қатар жергілікті желілер үшін деректерді өңдеу туралы ақпаратты және бағдарламалау хабарламасын желі бойынша жөнелтуге көмектесетін шлюздік хаттамалар;

- аталған деңгейлерге жатпайтын, бірақ клиентке желіде ыңғайлы жұмыс жасауға мүмкіндік беретін басқа хаттамалар.

Пайдаланушылардың Internet ресурстарына қатынас құруы әдетте бағдарлама-навигаторлар немесе ***броузерлер*** (ағылшынша browser) көмегімен жүзеге асады. Қазіргі уақытта неғұрлым танымал бағдарламалар қатарына мыналар жатады: Netscape Navigator/Communicator (Netscape) және MS Explorer (Microsoft). Бұл бағдарламалар HTTP хаттамасын қолдануға негізделсе де, олар басқа Internet сервистеріне: электрондық почта, жаңалықтар және т.б. қарапайым қатынас құруға мүмкіндік береді.

Броузер, пайдаланушының желі ресурстарына қатынас құруын жүзеге асыра отырып, табиғаты бойынша, бағдарлама-клиент (немесе Web-клиент) болып табылады. Ақпараттық ресурстарды ұсынатын бағдарламалар Web-сервер болып табылады. Ол әртүрлі ақпарат көздерінен ақпараттарды жинау және алу, содан соң Web-клиентке стандартты түрде ұсыну бойынша негізгі жұмысты атқарушы болып табылады. Пайдаланушының деректер қорында орналасқан ақпараттарды таңдауын ұйымдастырауды қарастырайық.

Internet және intranet желілеріндегі деректер қоры

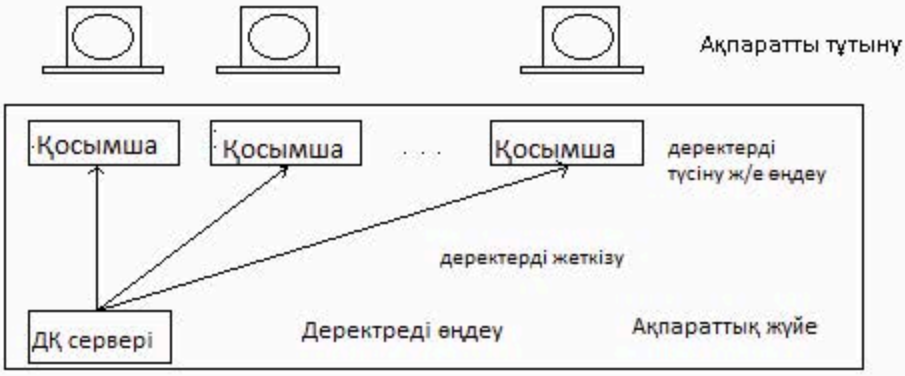
Intranet технологиясы, табиғаты бойынша, бірлескен АЖ ортасына көшірілген Internet технологиясы болып саналады. Internet құрылымы және intranet-тегі ақпараттық жүйелер алғашқы көппайдаланушылық орталықтандырылған есептеуіш жүйелерден (мэйнфреймдер) клиент-сервер жүйелері арқылы үлестірімді жүйелер типті ақпараттарды тікелей қолданатын орталықтандырылған өңдеу және даярлауға эволюциялық өтудің нәтижесі болып табылады. Даму кезеңдерін қысқаша қарастырайық.

1. ***Мэйнфреймдерде*** (4.13-сурет) есептеуіш ресурстар, сақталатын деректер және ақпаратты бағдарламаларды өңдеуы бір ЭЕМ-да шоғырланған. Қатынас құрудың негізгі құралдары ЭЕМ-мен басқарылатын алфавиттік-цифрлық терминал (дисплей) болды. Барлық ақпараттарды өңдеу және дайындау орталық ЭЕМ-де жүзеге асты. Терминалдардан, қағидаға сәйкес, машинаға пернелерді басу коды немесе экран буфері құрамы жіберілді, ал кері қарай терминалға – басқарудың сәйкес кодтарымен бейнеленген экрандар жіберілді.

4.13-сурет. Орталықтандырылған көппайдаланушылық жүйелер

Жүйелер ***құндылығы*** оны басқару, ақпараттарды қорғау және түрлендірулердің қарапайымдылығы болып табылады. Кемшіліктеріне процессорлар және байланыстар желісінің (көптеген пайдаланушылар жұмыс жасаған кезде жүйелер реакциясының төмендігінің салдары ретінде) үлкен жүктемесін, сенімділіктің аздығын (ЭЕМ істен шығуы барлық жүйенің жұмысын тоқтатады), жүйелерді масштабтаудың қиындығын және басқаларды жатқызуға болады.

Тарихи тұрғыда ақпараттық жүйелер облысындағы келесі шешім клиент-сервер құрылымы (4.14-сурет) болды.



4.14-сурет. Клиент-сервер жүйелері

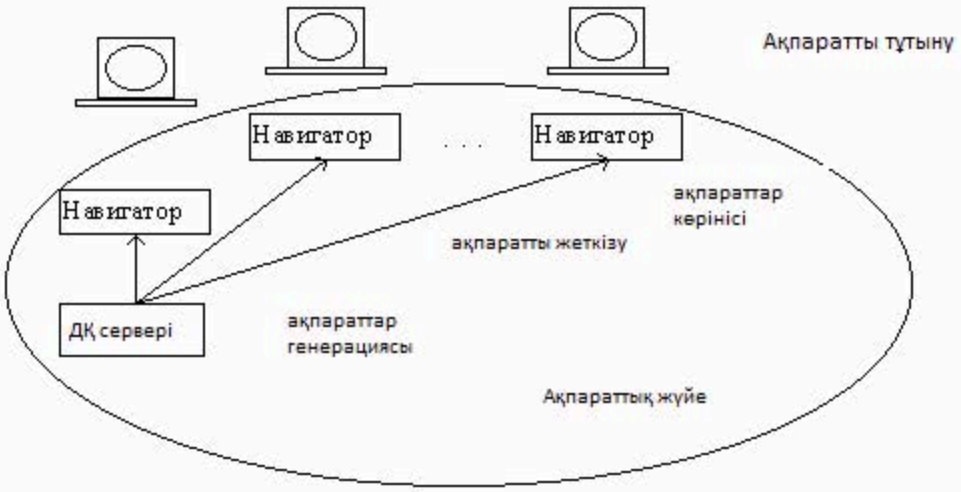
Осы жүйелерде терминал орын ДЭЕМ басты, ал мэйнфрейм қызметін компьютер-сервер атқарды. Бұрын біз функциялар арасындағы компоненттерді әртүрлі үлестіретін ұқсас жүйелер үлгілерін қарастырдық. Егер "орналасқандағы көрініс" (мағынасы бойынша орталықтандырылған көппайдаланушылық жүйелер үлгісін қайталайды) үлгісін есепке алмасақ, онда клиент-сервер жүйелері келесі ***құндылықтарға*** ие болады: жоғары өміршеңдік және сенімділік, масштабтаудың жеңілдігі, сапалы пайдаланушы интерфейсі, бір уақытта бірнеше қосымшамен жұмыс жасау мүмкіндігі, ақпаратты өңдеу шапшаңдығы.

Клиент-серверлік жүйелердің негізгі ***кемшілігі*** олардың ақпараттарға емес, деректерге негізделгені болып табылады. Бұл пайдаланушыдан зерттеу облысын ғана емес, қолданбалы бағдарламалардың ерекшеліктерін де білуді талап етеді.

Тағы бір кемшілігі - жүйелер компоненттерінің өзара әрекетінің "жабықтығынан" қолданылатын хаттамаларда мұндай жүйелерді басқа компьютерлік платформаларға көшірудің және басқа дестелермен интеграциялаудың қиындығы.

Мұндай жүйелерді басқару қиын және пайдаланушының немесе компьютерлік вирустардың болжап болмайтын немесе бүлдірушілік әрекеті кезінде осалдық танытады.

Бірлескен intranet жүйелерінің, клиент-сервер жүйелерінен айырмашылығы, олар деректерге емес, мамандандырылған пайдаланушы пайдалану үшін ақпаратты нақты және жарамды түрде беруге негізделген. (4.15-сурет).

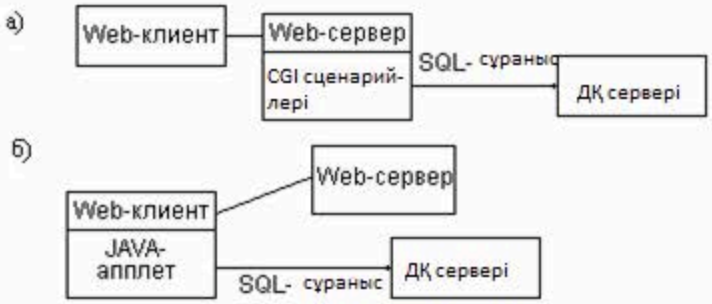


4.15-сурет. Ақпаратпен қамтамасыз ететін жүйелер

Жаңа жүйелер орталықтандырылған көппайдаланушылық жүйелердің және клиент-сервер типті жүйелердің артықшылықтарын қамтиды. Оларға тән қасиеттер:

- серверде деректер емес, пайдалану үшін жарамды ақпараттар туындайды (мысалы, ДҚБЖ жағдайында – ДҚ жазбасы);
- клиенттік және серверлік бөліктер арасында ақпарат алмасуда қандай да бір фирманың емес, ашық стандартты хаттама қолданылады;
- қолданбалы жүйелер серверде орналасады, және сондықтан пайдаланушының жұмыс жасауы үшін компьютер-клиентте бағдарламалау-навигаторы болуы жеткілікті.

Түпнұсқа ақпараттар Internet және intranet желілеріндегі ДҚ болған жағдайда WWW және дәстүрлі ДҚБЖ компонентері өзара әрекеттеседі. ДҚ-на қатынас құру бойынша бағдарламалық қамтамасыз етумен жұмыс жасаудың екі келесі нұсқасын ажыратады: Web-сервер жағында (0.16а-сурет) және Web-клиент жағында (0.16б-сурет).



4.16-сурет. Internet желісінде деректер қорына қатынас құру үлгілері

ДҚ-на сервер жағынан қатынас құру үлгілерінде ДҚ серверіне қатынас құру әдетте *сыртқы* қатынас бойынша оларға сәйкес бағдарламалар интерфейстерінің бірімен: CGI (Common Gateway Interface – жалпы шлюздік интерфейс), FastCGI немесе API (Application Program Interface – қолданбалы бағдарламалау интерфейсі) Web-сервердің бағдарламаларын шақыру жолымен жүзеге асады.

Сыртқы бағдарламалардың ДҚ серверімен өзара әрекеттесуі SQL тілінде нақты серверге немесе ODBC драйверін пайдаланып (9.3-бөлімін қараңыз) тікелей қатынас құру арқылы жүзеге асады.

Сыртқы бағдарламалар Си, Си++ және Паскаль сияқты қарапайым бағдарламалау тілінде немесе мамандандырылған Perl немесе PHP тілінде жазылады. CGI интерфейске сәйкес құрылған бағдарламалар ***CGI-сценарийлер*** немесе ***CGI-скрипттер*** деп аталады.

Бұл механизмді ***клиент жағынан*** сүйемелдеу үшін HTML тілінде ДҚ-на сұраныстарда құжат формасына (HTML тілінде <FORM> тэгі) қосылатын құралдар бар.

ДҚ-на қатынас құру процедурасы CGI интерфейсін қолдану арқылы келесі кезеңдерді іске қосады.

1. Егер пайдаланушының құжатты көруі кезінде Web-клиент белгілі бір бетке сілтемені кездестіретін болса, онда Web-сервер беттерге ДҚ-на қатынас құру формасын қамтитын сұраныс жасайды.

2. Web-клиентпен алынған беттегі ДҚ-на сұраныс формаларын толтыру және оны Web-серверге жөнелту. Формаларды толтыру дұрыстығын HTML-беттерде тікелей орналасқан, формаларды сипаттайтын күрделі емес бағдарламалар көмегімен бақылауға болады (әдетте осы үшін VBScript немесе JavaScript тілдерін қолданады).

3. Web-сервер, осы форманы алып, оған параметрлерді (адресі және аты осы бағдарламалардың аты көрсетілген ACTION атрибутының <FORM> тэгі) бере отырып, сәйкес сыртқы CGI-бағдарламалауды жүктейді.

4. Сыртқы бағдарлама сипатталған ДҚ-на сұраныс формасын ДҚ серверіне қатынас құратын SQL тіліндегі сұраныс мәтініне сәйкес өзгертеді.

5. Сұраныс нәтижелерін алғаннан кейін сыртқы бағдарлама талап етілген HTML-бетті қалыптастырады, оны Web-серверге жібереді және өз орындауын аяқтайды.

6. Web-сервер қалыптасқан HTML-бетті Web-клиентке жібереді.

Деректерге қатынас құруды ұйымдастыру деректер қоры үшінна CGI мамандануын қолданудың негізгі *құндылықтары:*

- бағдарламалау тілдерінен тәуелсіздік – сценарий практикалық тұрғыдан жолдарды өңдеу құралдары бар кез-келген бағдарламалау тілінде жазылған болуы мүмкін;
- процестің жекелігі – сценарий сервердің қорғалған жүйелік ақпараттарына қатынас құрмай, серверде жеке процесс ретінде орындалады;
- кеңінен таралғандығы, өйткені әрбір Web-серверде CGI-стандартын пайдалануға болады;
- архитектуралық тәуелсіздік – CGI-стандартына сәйкес құрылған бағдарламалар сервер архитектурасынан тәуелсіз.

CGI мамандануын қолданудың негізгі кемшіліктеріне мыналар жатады:

- әрдайым деректер қорымен байланысты тағайындау және үзу қажеттілігі, себебі Web-сервер және ДҚБЖ-да тұрақты байланысты сүйемелдеу құралдары жоқ;
- бастапқы ақпараттарда сұраныстарды орындау, өңдеу үшін және сұраныстардың орындалу нәтижелеріне қойылатын шектеулер жоқ;
- бағдарламаларды бөлек процесс ретінде жүктеуге байланысты бағдарламалардың орындалуының қиындығы. Бұл деректерге сұраныстар жасауды баяулатады және сервер жұмысын тиімділігін азайтады.

CGI-мамандану кемшіліктерін жою үшін API мамандану құрылған. Бұл мамандану бойынша құрылған бағдарламалар жылдам және тиімді орындалады, себебі динамикалық кітапхана түрінде ұйымдасқан. Бұл түрдегі екі интерфейс неғұрлым белгілі: NSAPI (Netscape компаниясы) және ISAPI (Microsoft компаниясы).

Бұл технологияның құндылығы бағдарламалардың жылдам орындалуы болып табылады, өйткені бағдарлама негізгі серверлік процес шеңберінде орындалады. Бағдарламалардың өздері CGI-сценарийлерге қарағанда жоғары мүмкіндікке ие, мысалы, сервер файлдарына қатынас құруды бақылау мүмкіндігі пайда болды. API мамандандудың *кемшіліктеріне* мыналар жатады:

- бағдарламалау тілдерінен тәуелділігі – бағдарлама API деректерді сүйемелдейтін тілде ғана жазылған болуы мүмкін. Көбінесе бұл C және C++ (CGI-скриптерді жазу үшін кеңінен қолданылатын Perl тілі қолданылмайды) тілдері;
- серверді қолданбалы бағдарламалау кезінде қателіктерден қорғаудың әлсіздігі және жүйелік ресурстарға рұқсат етілмеген қатынас құру мүмкіндіктері, себебі API-бағдарлама сервердің негізгі процесінің адрестік кеңістігінде орындалады;
- бағдарламалар, қағидаға сәйкес, басқа бағдарламалық-аппараттық платформаларға тасымалданбайды, себебі сервердің интерфейсі және архитектурасына бекітілген.

FastCGI интерфейсі алдыңғы технологиялардың артықшылықтарын үйлестіреді және CGI интерфейсіне неғұрлым жақын. Оларға сәйкес жазылған қосымшалар ішінара жекелеген процестер ретінде жүктеледі, бірақ әрқашан белсенді болады және деректерге CGI-сценарий ретінде қатынас құрғаннан кейін аяқталмайды.

ДҚ-на ***клиент жағынан*** қатынас құрғанда Web-клиент және ДҚ серверінің өзара әрекеті механизмдері жүзеге асуының негізгі құралдары Java тілі болып табылады. Сонымен қатар, процедуралық құралдарды қосу негізінде ресми түрдегі HTML тілінің мүмкіндіктерін кеңейту үшін (меншіктеу, салыстыру және т.б. операторлары жоқ) құрылған JavaScript қолданылуы мүмкін. Бағдарламалар JavaScript тілінде компьютерде Web-броузермен талдаулар режимінде орындалады.

Егер HTML-құжатта ДҚ-нан деректер алу талап етілсе, онда ол келесі түрде жүзеге асады.

1. Java тілінде қосымшалар *апплеттер* (Java-applets) деп аталатын бағдарламалармен жазылады, содан соң олар компиляцияланады. Нәтижесінде бейімделгіш (машиналық-тәуелсіз) бағдарламалар алынады. Соңғысы талдаулар режимінде орындалуы мүмкін немесе бастапқы ақпараттармен бағдарламаларды тудыру үшін әртүрлі аппараттық-бағдарламалық платформаларда орындалуға дайын.
2. HTML-құжат мәтінінде қажетті орындарда сәйкес апплеттерге сілтемелер қойылады. Бағдарламалардың өздері серверде сақталады.
3. Гипермәтінмен жұмыс кезінде мәтінінде апплетке сілтемелер кездескен кезде Java-бағдарламалары серверден броузердің орындалу ортасына автоматты түрде жөнелтуді және орындауға жіберуді жүзеге асады. Соңғысы пайдаланушымен сұқбаттасу кезінде ДҚ-на сұраныс параметрлерін (Java тілінде пайдаланушының графикалық интерфейсін сүйемелдеу құралдары бар) нақтылайды.
4. Java-апплет басқаруды алып, ДҚ серверімен өзара әрекетті жүзеге асырады, нәтижесінде қордан алынған ақпарат пайдаланушыға ұсынылады.

ДҚ серверіне қатынас құру үшін ODBC концепциясына негізделген JDBC стандарты (Java DataBase Connectivity – Java деректер қоры үшін үйлесімділігі) құрылған. JDBC стандарты Sun/JavaSoft фирмасымен құрылған және деректердің әртүрлі қорына Java тілінде әмбебап қатынас құруды қамтамасыз етеді.

Қарастырылған екі сызбаның ішінен бірінші немесе екінші нұсқауды бірмәнді айрықша атауға болмайды. Барлығы клиент-серверлік бағдарламалардың өңдеу мақсаты мен шарттарынан тәуелді.

Сервер жағынан қатынас құру үлгілерінің ***құндылығы*** бағдарлама-навигаторлардың (Web-клиенттер) салыстырмалы түрде қарапайымдылығы және жүйелерді басқарудың ыңғайлылығы болып табылады, өйткені бағдарламалық қамтамасыз етудің негізгі бөлімі Web-сервер машинасында орналасады. Айқын ***кемшілігі:*** Web-серверге үлкен жүктемелер болған кезде және оның қуаты жетпегенде ақпараттарды алу шапшаңдығы азаяды.

Екінші үлгілер бірінші үлгіге қарағанда қиынырақ. Бұл навигатор жұмысын қиындатады, бірақ онда Web-сервер уақытша жүктеледі.

Салыстырмалы түрде Microsoft фирмасының кейіннен шыққан бағдарламалық өнімдері екі сызбаны да сүйемелдейді. Клиент жағынан (Internet Explorer ортасында) динамикалық HTML қолдану мүмкіндігі бар, ол VBScript тілінде жүзеге асырылады. Деректерге қатынас құруды сервер жағынан asp-беттер (Active Server Pages – белсенді серверлік беттер) деп аталатын беттер көмегімен жүзеге асады. Asp-беттер – бұл VBScript сценарий тілінде жазылған, Web-серверде (Internet Information Server) сақталады және орындалады.

Бақылау сұрақтары және тапсырмалар

1. Компьютер-сервер және компьютер-клиент арасында екібуынды үлгілер үшін функцияларды бөлудің типтік нұсқаларын атаңыз.
2. Деректерге қашықтықтан қатынас құру үлгісін сипаттаңыз.

3. Деректер қоры сервері үлгілерінің құндылықтары және кемшіліктеріи атаңыз.
4. Қосымшалар сервері үшбуынды үлгілерінің құрылымдық сызбасын суреттеңіз.
5. Транзакциялар мониторы қалай тағайындалады?
6. Транзакциялар мониторы үлгісінің сызбасын суреттеңіз және сипаттаңыз.
7. Орталықтандырылмаған ДҚ басқарудың негізгі технологиясын атаңыз.
8. Үлестірімді ДҚ үлгілерінде жұмыс жасау динамикасын сипаттаңыз.
9. Үлестірімді ДҚ үлгілерінің артықшылықтары және кемшіліктерін түсіндіріңіз.
10. Екіфазалы транзакциялар фиксациясы әдісін түсіндіріңіз.
11. Деректердің таралымын көбейту үлгісін сипаттаңыз.
12. Деректерге қатынас құрудың негізгі әдістерін атаңыз және оларды пайдалану артықшылығын түсіндіріңіз.
13. Жинаққа шектеу қоюға мысал келтіріңіз.
14. Шектеу қою ережесін түсіндіріңіз.
15. Тұйыққа тірелудің негізгі түрлерін атаңыз.
16. Үлестірімді ДҚ-да өзара тұйыққа тірелуге мысал келтіріңіз.
17. Жергілікті желіде ақпараттық жүйелер құрудың негізгі нұсқаларын түсіндіріңіз.
18. Желілік ДҚБЖ-мен файл-сервер типті жүйелермен жұмыс жасау сызбасын түсіндіріңіз.
19. Желілік ДҚБЖ-мен файл-сервер типті жүйелердегі ақпараттық өңдеу қалай ұйымдастырылады?
20. Ақпараттық клиент-сервер типті жүйелермен жұмыс жасау сызбасын сипаттаңыз.
21. Сақталатын процедуралар және триггерлер қалай тағайындалады?
22. Сақталатын командалар және курсор түсінігіне анықтама беріңіз.
23. ODBC интерфейсі көмегімен деректерге қатынас құру қалай ұйымдастырылады?
24. Internet желісіне жалпы сипаттама беріңіз.
25. Internet желісінде ДҚ-на қатынас құрудың негізгі үлгілерін түсіндіріңіз.
26. Internet желісінде ДҚ-на қатынас құру үшін қандай бағдарламалау тілдері қолданылады?
27. CGI-сценарийлер қалай тағайындалады?
28. Intranet технологиясын сипаттаңыз.

5. ДЕРЕКТЕР ҚОРЫН ЖОБАЛАУ ЖӘНЕ ҚОЛДАНУ

Бұл бөлімде реляциялық деректер қорын жобалау мәселелері қарастырылады. Жобалау мәселелерінің сипаттамасы және оларды жүзеге асыру әдістері беріледі.

5.1. Жобалау мәселелері

Деректер қорын қамтитын ақпараттық жүйелерді жобалау физикалық және логикалық деңгейде жүзеге асады. Жобалау мәселелерін физикалық деңгейде шешу көбінесе қолданылатын автоматтандырылған ДҚБЖ-дан тәуелді. Кейбір жағдайларда пайдаланушыға үлкен мәселелер туындатпайтындай, жекелеген жүйелер параметрін баптау мүмкіндігін ұсынады.

Логикалық жобалау кестенің санын және құрылымын анықтау, ДҚ-на сұраныстарды қалыптастыру, құжаттарда есеп беру түрлерін анықтау, ақпаратты өңдеу алгоритмін құру, қордағы деректерді енгізу және өңдеу үшін форма құру және тағы басқалардан тұрады.

ДҚ логикалық жобалау мәселелерін шешу көбінесе маманданған есептерді зерттеу облысында анықталады. Мұнда неғұрлым маңыздысы деректерді құрылымдау мәселесі болып табылады, біз соған басты назар аударамыз.

Деректердің құрылымын жобалау кезінде автоматтандырылған жүйелер үшін үш негізгі әдісті атауға болады.

1. Шығарылатын есептің объектілері туралы, бір кестенің (бір қатынас) және оның бірнеше өзара байланысқан кестелермен соңғы декомпозиция процедуралары негізінде, ақпараттарды жинау.

2. Жүйелер туралы білімді (бастапқы деректердің типін және олардың өзарабайланысын анықтау) қалыптастыру және деректерді өңдеуге қойылатын талаптарды анықтау, CASE-жүйелер (автоматтандырылған жобалау және деректер қорын өңдеу жүйелері) көмегімен ДҚ-ның дайын сызбасын немесе тіпті дайын қолданбалы ақпараттық жүйелер алу.

3. Ақпараттық жүйелерде пайдалану үшін ақпараттарды құрылымдау процесі жүйелерге талдау жасау ережелері және нұсқаулар жиынтығы негізінде жасалады.

Классикалық және тарихи тұрғыда алғашқы болып табылатын әдістердің біріншісін қарастырайық. Ең алдымен, реляциялық үлгілер қатынастарында деректердің құрылымын анықтау кезінде орын алатын негізгі мәселелерге сипаттама береміз.

Деректердің артық қосарланушылығы және ауытқушылықтар

Деректердің қарапайым (артық емес) және артық қосарланушылығын айыра білген жөн. Бұның алғашқысының деректер қорында бар болуына рұқсат беріледі, ал деректердің артық қосарланушылығы – деректерді өңдеу кезінде мәселелер туындатуы мүмкін. Қосарланушылықтың екі нұсқасына да мысалдар келтірейік.

Деректердің *артық емес қосарланушылығының* 5-мысалдың 1-суретінде келтірілген: С_Т қатынасының Қызметкер және Телефон атрибуттары бар. Бір ғимаратта жұмыс жасайтын қызметкерлердің телефон нөмерлері бірдей болады. Әрбір қызметкердің телефон нөмері бірегей болса да, 4328 телефон нөмері бірнеше рет кездеседі. Сондықтан нөмерлердің ешқайсысы артық болып табылмайды. Шындығында, телефон нөмерлерінің бірін өшіру кезінде қызметкерлердің біріне қай нөмер бойынша хабарласуға болатындығы туралы ақпарат өшеді.

Қ_Т

Қызметкер	Телефон
Иванов	3721

Петров	4328
Сидоров	4328
Егоров	4328

5.2-сурет. Артық емес қосарланушылық

Артық қосарланушылықтың (артықтық) 5-мысалдың 2а-суретінде келтірілген. Қ_Т_Б қатынасының Қ_Т қатынастарынан айырмашылығы: Н_комн (қызметкер бөлмесінің нөмері) қосылған. Әрине бір бөлмеде отырған барлық қызметкерлердің телефон нөмерлері бірдей. Сондықтан, қарастырылған қатынаста деректердің артық қосарланушылығы бар. Ендеше, байланыстарда Петров отырған бөлмеде Сидоров және Егоров орналасқандықтан, олардың нөмерін Петров туралы мәліметтерден білуге болады.

5.2б-суретте Қ_Т_Б сәтсіз қатынастарының мысалы келтірілген, онда Сидоров және Егоров телефон нөмерлерінің орнына "сызық" (мәні анықталмаған) қойылған. Артық қосарланушылықты шағарып тастау келесі түрде жүзеге асады. Біріншіден, бағдарламалау кезінде кестелерде "сызық" қойылған ақпараттарды іздеу механизмін құруға қосымша уақыт кетеді. Екіншіден, деректердің қосарланушылығы болса да, жады барлық "сызық" қойылған атрибуттарға бірдей бөлінеді. Үшіншіден, ең маңыздысы, коллективтен Петровты шығарған кезде қатынастардан оның барлық мәліметтерімен бірге 111-бөлме телефон номері туралы ақпарат та өшеді, бұған жол беруге болмайды.

а) Қ_Т_Б

Қызметкер	Телефон	Б_нөмері
Иванов	3721	109
Петров	4328	111
Сидоров	4328	111
Егоров	4328	111

б) Қ_Т_Б

Қызметкер	Телефон	Б_нөмері
Иванов	3721	109
Петров	4328	111
Сидоров	-	111
Егоров	-	111

5.3-сурет. Артық қосарланушылық

Осы жағдайдан шығудың мүмкін әдісі 5 мысалдың 3-суретінде келтірілген. Мұнда екі қатынастар Қ_Б және Б_Т көрсетілген, ол Қ_Т_Б бастапқы қатынастарын декомпозициялау жолымен алынған. Алғашқысы қызметкерлер орналасқан бөлме нөмерлері туралы ақпаратты, ал екіншісі – әрбір бөлменің телефон нөмерлері туралы ақпаратты қамтиды. Енді, егер Петровты қызметтен шығарып, деректер қорындағы ол туралы барлық мәліметтерді өшірсе де, бұл 111-бөлме телефон нөмері туралы ақпаратты жоғалтпайды.

Қ_Б

Қызметкер	Б_нөмері
Иванов	109
Петров	111
Сидоров	111
Егоров	111

Б_Т

Б_нөмері	Телефон
109	3721
111	4328

5.3-сурет. Артық қосарланушылықты жою

Қ_Т_Б қатынастарын екі Қ_Б және Б_Т қатынастарға декомпозициялау қатынастарды нормалаудың негізгі процедурасы болып табылады.

Деректердің артық қосарланушылығы қатынастар кортежін өңдеу кезінде Э. Кодд "қатынастарды жаңартудағы ауытқушылықтар" деп атаған мәселелер туындатады. Ол қайсыбір қатынастар үшін өшіру, қосу немесе редактрлеу кезінде мәселелер туындайтынын көрсетті.

Ауытқушылықтар деп ДҚ-да қайшылықтар тудыратын немесе деректерді өңдеуді недәуір қиындататын жағдайларды айтамыз.

Ауытқушылықтардың негізгі үш түрін бөліп көрсетеді: түрлендірулер (немесе өңдеу) кезіндегі ауытқушылықтар, өшіру кезіндегі ауытқушылықтар және жаңа мәлімет қосу кезіндегі ауытқушылықтар.

Түрлендірулер кезіндегі ауытқушылықтар кезінде бір мәліметтің мәнінің өзгеруі нәтижесінде барлық кестелерде және қайсыбір басқа кестелердегі сәйкес жазбаларда өзгерістер болуы мүмкін.

Ендеше, мысалы, 111-бөлмедегі телефон нөмерін (5.2а-сурет) өзгерту барлық Қ_Т_Б кестені қарап және кестелер жазбаларындағы Петров, Сидоров және Егоровқа қатысты Б_нөмері өрістеріне өзгерістер енгізуді талап ететін жалғыз факт.

Өшіру кезіндегі ауытқушылықтар кезінде кестелерден мәліметтерді өшіру нәтижесінде өшірілген деректермен тікелей байланысты басқа ақпараттардың өшіп қалуы мүмкін.

Қарастырылған Қ_Т_Б кестесінде қызметкер Иванов туралы жазбаны (мысалы, жұмыстан шығу немесе кезекті демалысқа кетуі) өшіру 109-бөлмеде орнатылған телефон нөмері туралы ақпараттың жойылуына апарып соқтырады.

Жаңа мәлімет қосу кезіндегі ауытқушылықтар кестелер толық емес кезде оған жаңа жазба кірістіру жағдайында туындайды.

Мысалы, Қ_Т_Б кестесіне жаңа қызметкер қосу операциясы. Әлбетте, бұл кестеде тек қана бөлме және телефон нөмері туралы мәліметті сақтау дұрыс емес (егер ол бөлмеде ешқандай қызметкер орналаспаса). Сонымен қатар, егер Қ_Т_Б кестеде Қызметкер өрісі кілттік өріс болып табылса, онда кілттік өрістердің мәні анықталмағандықтан, қызметкердің фамилиясы көрсетілмеген толық емес жазбаларды сақтауға жол беруге болмайды.

Жаңа мәлімет қосу кезіндегі ауытқушылықтар болуы жағдайының екінші мысалы - кестеге жаңа қызметкерді қосу. Мұндай жағдайда жаңа қызметкермен бір бөлмеде отыратын ең болмағанда бір қызметкердің телефон нөмері және сәйкес бөлме нөмерін тексеру керек. Егер бір бөлмеде отыратын қызметкерлердің телефон нөмерлері әртүрлі болса, онда тіпті түсініксіз жағдай туындайды (бөлмеде бірнеше телефон бар немесе телефон нөмерлерінің бірі қате жазылған).

Бастапқы қатынастарды қалыптастыру

ДҚ жобалау барлық объектілерді, деректер қорына қосылатын мәліметтерді және олардың атрибуттарын анықтаудан басталады. Содан соң атрибуттар бір кестеге жинақталады – Бастапқы қатынас.

Мысалы. Бастапқы қатынастарды қалыптастыру.

Факультеттің оқу бөлімі үшін оқытушылар туралы ДҚ құрылсын. ДҚ жобалаудың бірінші кезеңінде тапсырыс берушімен (оқу бөлімінің бастығы) келісім нәтижесінде қордағы мәліметтер, олар қалай қолданылады және оның жүзеге асуы нәтижесінде тапсырыс беруші қандай ақпараттарды алғысы келетіні анықталуы керек. Нәтижесінде ДҚ қатынастарында болуы керек атрибуттар және олардың арасындағы байланыстар орнатылады. Атрибуттардың атауларын атап өтейік және оларға қысқаша сипаттамалар берейік:

Аты-жөні – оқытушы фамилиясы және аты-жөнінің бірінші әріптері. Оқытушылардың фамилиясы және аты-жөнінің бірінші әріптері қайталанбайтын болсын.

Қызметі – оқытушының атқаратын қызметі.

Жалақысы – оқытушы жалақысы.

Жұмыс өтілі – оқытушылық жұмыс өтілі.

Y_Жұмыс өтілі – жұмыс өтіліне үстеме.

Кафедра – оқытушы жұмыс жасайтын кафедра нөмері.

Пән – оқытушының оқытатын пәнінің атауы.

Тобы – оқытушы сабақ өтетін топтар нөмері.

Сабақ түрі – оқытушының оқу тобында жүргізетін сабақ түрі.

Қатынастарға қойылатын талаптардың бірі - барлық қатынастар атрибуттарының мәні қарапайым болуы керек. Қарастырылып отырған қатынаста әрбір атрибут кортеж, сонымен қатар қарапайым болуы керек. Мысалы, ОҚЫТУШЫ бастапқы қатынастары 5.4-суретте келтірілген.

ОҚЫТУШЫ

Аты-жөні	Қыз-меті	Жала-қысы	Жұм. өтілі	Y_ ж.өтілі	Каф	Пән	Тобы	Саб.түрі
Иванов И.М.	Оқыт.	500	5	100	25	СУБД	256	Практ
Иванов И.М.	Аға оқыт.	500	5	100	25	ПЛ/1	123	Практ
Петров М.И.	Аға оқыт.	800	7	100	25	СУБД	256	Дәріс
Петров М.И.	Аға оқыт.	800	7	100	25	Паскаль	256	Практ
Сидоров Н.Г.	Оқыт.	500	10	150	25	ПЛ/1	123	Дәріс
Сидоров Н.Г.	Оқыт.	500	10	150	25	Паскаль	256	Дәріс
Егоров В.В.	Оқыт.	500	5	100	24	ПЭВМ	244	Дәріс

5.4 - сурет. ОҚЫТУШЫ бастапқы қатынастары

Көрсетілген қатынас келесі ОҚЫТУШЫ (Аты-жөні, Қызметі, Жалақысы, Жұмыс өтілі, Y_Жұмыс өтілі, Кафедра, Пән, Тобы, Сабақ түрі) сызбасына ие.

ОҚЫТУШЫ бастапқы қатынасында деректердің артық қосарланушылығы бар, олар өңдеу кезіндегі ауытқушылықтар салдары болып табылады. Айқын және айқын емес артықтықты ажыратады.

Айқын артықтық мысалы: ОҚЫТУШЫ қатынастарында бірнеше топтарға сабақ жүргізетін оқытушылар туралы деректер сәйкесінше бірнеше рет қайталанады. Мысалы, ОҚЫТУШЫ қатынастарында барлық деректер Иванов бойынша екі рет қайталанады. Сондықтан, егер Иванов И.М. аға оқытушы болса, онда бұл факт екі жолда да көрсетілуі керек. Әйтпесе бұл деректердің қайшылығына апарып соқтырады.

Айқын емес артықтық ОҚЫТУШЫ қатынастарында барлық оқытушылардың бірдей жалақысында көрінеді және бірдей жұмыс өтілі үшін жалақыға бірдей үстеме қосылған. Сондықтан, егер қызметі үшін жалақыларын 500-ден 510-ға өзгертсек, онда бұл мән Сидоровтан басқа барлық оқытушылардың жалақысын өзгертеді де деректер қорында қайшылықтар пайда болады.

Артық қатынастарды құралдар көмегімен шығарып тастау қатынастарды нормалау нәтижесі болып табылады, оны неғұрлым жан-жақты қарастырайық.

5.2. Қалыпты түр әдісі

ДҚ жобалау ақпараттық жүйелердің өміршеңдік кезеңінің бір кезеңі болып табылады. ДҚ жобалау процесіндегі негізгі міндет оның қатынастарын қалыпқа келтіру мәселесі болып табылады. Төменде қарастырылатын қалыпты түр әдісі реляциялық ДҚ жобалаудың классикалық әдісі болып табылады. Бұл әдіс реляциялық деректер қоры теориясында қатынастар атрибуттары арасындағы тәуелділік ұғымына негізделген.

Атрибуттар арасындағы тәуелділіктер

Қатынастар атрибуттары арасындағы тәуелділіктердің негізгі түрлерін қарастырайық: функциональдық, транзитивтік және көпмәнді.

Функционалдық тәуелділік түсінігі базалық түсінік болып табылады, өйткені оның негізінде тәуелділіктердің барлық қалған түрлері анықталады.

Егер әрбір А мәніне нақты бір В мәні сәйкес келсе, онда В атрибуты А атрибутынан ***функционалды тәуелді*** болады. Математикалық тұрғыда В-ның А-дан функционалдық тәуелділігі А→В түрінде жазылады. Бұл барлық кортеждерде А атрибутының бірдей мәніне В атрибутының бірдей мәні сәйкес келетінін білдіреді. Ескере кетейік, А және В құрама болуы мүмкін – екі және одан да көп атрибуттардан тұрады.

5.4-суреттегі қатынастарда атрибуттар арасындағы функционалдық тәуелділіктерді Аты-жөні→Кафедра, Аты-жөні→Қызметі, Қызметі→Жалақысы және басқа көрсетуге болады. Қатынастарда функционалдық тәуелділіктің бар болуын қатынастар кортеждері ұсынған ақпараттар анықтайды. 5.4-суреттегі қатынастарындағы кілт құрама болып табылады және Аты-жөні, Пән, Тобы атрибуттарынан тұрады.

Функционалдық өзара тәуелділік. Егер А→В және В→А түріндегі функционалдық тәуелділік бар болса, онда А және В арасында өзара бірмәнді сәйкестік немесе функционалдық өзара тәуелділік бар. А және В атрибуттары арасындағы функционалдық өзара тәуелділікті А↔В немесе В↔А деп белгілейміз.

Мысалы. Бір-бірінен функционалды тәуелді екі атрибуттан тұратын қайсыбір қатынас берілсін. Бұл төлқұжат сериясы не төлқұжат номері (N) және төлқұжат иесінің фамилиясы, аты және әкесінің аты (Аты-жөні). Аты-жөні өрістерінің төлқұжат нөмерінен функционалды тәуелділігі N өрістерінің мәні Аты-жөні өрістерінің мәнін бірмәнді анықтайтынын білдіреді. N өрістерінің бір мәніне Аты-жөні өрістерінің жалғыз ғана мәні сәйкес келеді. Осы жағдайда кері тұжырым да дұрыс: Аты-жөні өрістерінің бәр мәніне N өрістерінің тек бір ғана мәні сәйкес келеді. Бұл мысалда екі адамның фамилиясы, аты және әкесінің аты толық сәйкес келуі мүмкін емес.

Кілттік емес атрибуттың кілттің құрама бөлігінен тәуелділігі ***ішінара тәуелділік*** (***ішінара функциональды тәуелділік***) деп аталады. Қарастырылған қатынаста Қызметі атрибуты кілттік бөлім болып саналатын АТЫ-ЖӨНІ атрибутынан функционалды тәуелді. Сондықтан, Қызметі атрибуты кілттік қатынастардан ішінара тәуелді.

Бұған балама нұсқа ***толық функционалдық тәуелділік*** болып табылады. Онда кілттік емес атрибут кілттің барлық құрама бөлігінен тәуелді. Біздің мысалда Сабақ түрі атрибуты кілттің құрамынан толық функционалды тәуелді.

Егер А, В, С атрибуттары үшін А→В және В→С шарты орындалса, бірақ кері тәуелділік болмаса, онда С атрибуты А атрибутынан ***транзитивті (транзитивті тәуелділік)*** тәуелді болады. 5.4-суреттегі қатынастарда транзитивті тәуелділік бар:

Аты-жөні→Қызметі→Жалақысы

Атрибуттар арасында көпмәнді тәуелділік орын алуы мүмкін.

Егер R қатынастарында әрбір А мәніне В мәндер жиыны сәйкес келсе немесе басқа R атрибутымен байланысқан болса, онда В атрибуты А атрибутынан ***көпмәнді тәуелді*** болады.

Көпмәнді тәуелділік «бірден көпке» (1:М), «көптен бірге» (М:1) немесе «көптен көпке» (М:М) болуы мүмкін, сәйкесінше белгіленеді: А⇒В, А⇐В және А⇔В.

Мысалы, оқытушы бірнеше пәннен сабақ беретін болсын, ал әрбір пәнді бірнеше оқытушы жүргізуі мүмкін, онда Аты-жөні⇔Пән тәуелділігі орындалады. Ендеше, 7.2-кестелерден көретініміз: оқытушы Иванов И.М. екі пәннен сабақ береді, ал ДҚБЖ пәнін екі оқытушы жүргізеді: Иванов И.М. және Петров М.И.

Ескерту. Жалпы жағдайда екі атрибуттар арасында мынадай тәуелділіктер болуы мүмкін: 1:1, 1:М, М:1 және М:М. Атрибуттар арасындағы тәуелділік ауытқушылықтардың себебі болып табылады, сондықтан қатынастарды атрибуттардың тәуелділігімен бірнеше қатынастарға бөлуге тырысады. Нәтижесінде 1:1, 1:М, М:1 және М-М (3.3-бөлім) түріндегі байланыстары бар байланысқан қатынастар (кесте) жиынтығы пайда болады. Кестелер арасындағы байланыстар әртүрлі қатынастардың атрибуттары арасындағы тәуелділіктерді бейнелейді.

Өзара тәуелсіз атрибуттар. Егер екі немесе бірнеше атрибуттардың ешқайсысы басқа атрибуттардан функционалды тәуелсіз болса, онда олар өзара тәуелсіз деп аталады.

Екі атрибуттың арасында тәуелділіктің болмауын, яғни А атрибутының В атрибутынан тәуелсіздігін келесі түрде жазамыз: А¬→В. А¬→В және В¬→А жағдайын А¬=В түрінде жазуға болады.

Атрибуттар арасында тәуелділіктерді анықтау

Атрибуттар арасында тәуелділіктерді анықтау қалыпты формалар әдісімен ДҚ жобалауды орындау үшін қажет.

Функционалдық тәуелділіктердің бар болуын анықтаудың негізгі әдісі – атрибуттар семантикасын мұқият талдау. Әрбір қатынастар үшін атрибуттар арасында функционалдық тәуелділіктердің қайсыбір жиыны бар болады, бірақ әрқашан емес. Сонымен қатар, егер қайсыбір қатынаста бір немесе бірнеше функционалдық тәуелділіктер бар болса, онда бұл қатынаста басқа функционалдық тәуелділіктер шығаруға болады.

Мысалы. R қатынасы R(А1, А2, А3) сызбасымен және сандық мәндермен келесі кестемен берілсін.

А1	А2	А3
12	21	34
17	21	34
11	24	33
13	25	31
15	23	35
14	22	32

R қатынасында функционалдық тәуелділіктің бар екені белгілі: A1→ A2 және A2→ A3.

Бұл қатынасты талдап, онда басқа да тәуелділіктердің бар екендігін көруге болады:

A1→ A3, A1A2→ A3, A1A2A3→ A1A2, A1A2→ A2A3 және т.с.с.

Оны келесі түрде жазуға болады:

A2¬→A1, A3¬→A1 және т.б.

A1 атрибутының A2 (A2¬→A1) атрибутынан тәуелді болмауы A2 (21) атрибутының бір ғана мәніне A1 (12 және 17) атрибутының әртүрлі мәндері сәйкес келетіндігімен түсіндіріледі. Басқаша айтқанда, функционалдық емес көпмәнділік орын алған.

R қатынастарындағы барлық қолданыстағы функционалдық тәуелділіктерді ескере отырып, функционалдық тәуелділіктердің толық жиынын аламыз, оны F^+ деп белгілейік.

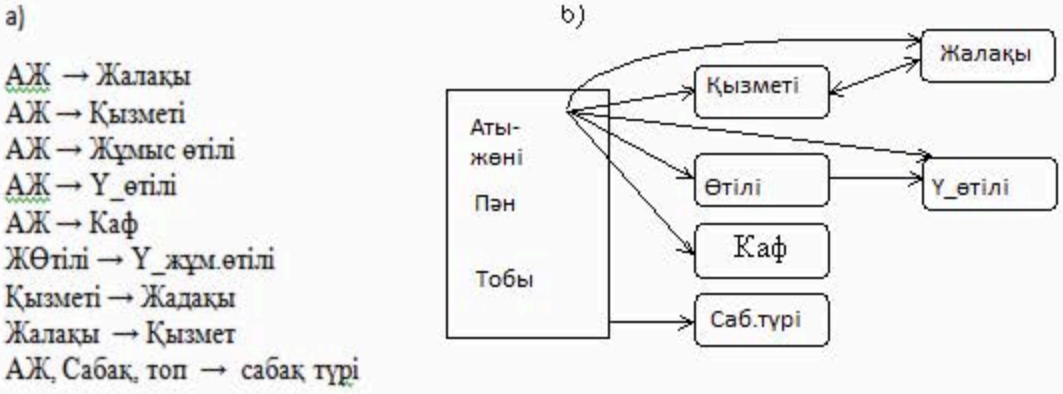
Осылайша, соңғы мысал үшін F = (A1→ A2, A2→ A3) бастапқы жиыны, ал F^+ = (A1→ A2, A2→ A3, A1→ A3, A1A2→ A3, A1A2A3→ A1A2, A1A2→ A2A3, ...) толық жиын.

F жиынынан F^+ жиынын тұрғызу үшін функционалдық тәуелділіктерді шығару ережелерін (немесе аксиомаларын) білу керек.

8 негізгі аксиома бар: рефлексивтілік, толтыру, транзитивтік, кеңейту, жалғастыру, псевдотранзитивтік, біріктіру және декомпозиция. Аталған аксиомалар барлық функционалдық тәуелділікті алуды қамтамасыз етеді, яғни олардың жиынтығын шығару процедурасына қарай «функционалды толық» деп есептеуге болады. Аксиомалар мазмұны және сәйкес мысал 1-қосымшаларда келтірілген.

5.4-суретте келтірілген ОҚЫТУШЫ қатынастарында атрибуттар арасындағы тәуелділіктерді анықтайық. Сонымен қатар, осы қатынаста орындалатын келесі шарттарды ескерейік: бір оқытушы бір топта сабақтың бір түрін жүргізеді (лекция немесе практикалық сабақ).

Қатынастарды талдау нәтижесінде 5.5-суретте көрсетілген атрибуттар арасындағы тәуелділіктерді аламыз.



5.5-сурет. Атрибуттар арасындағы тәуелділіктер

Қарастырылған мысал үшін осы функционалдық тәуелділіктерді анықтау келесі тұжырымдарға апарады.

Факультет оқытушыларының фамилиясы, аты және әкесінің аты бірегей. Әрбір оқытушыға оның жұмыс өтілі бірмәнді сәйкес келеді, яғни Аты-жөні→Жұмыс өтілі функционалдық тәуелділігі орындалады. Кері тұжырым қате, өйткені әртүрлі оқытушылардың бірдей жұмыс өтілі болуы мүмкін.

Әрбір оқытушы жұмыс өтілі үшін үстеме алады, Аты-жөні→Д_Жұмыс өтілі функционалдық тәуелділігі бар; бірақ кері функционалдық тәуелділік болмайды, өйткені бірнеше оқытушылардың бірдей үстеме алуы мүмкін.

Әрбір оқытушының қызметі (оқытушы, аға оқытушы, доцент, профессор) белгілі, бірақ бірнеше оқытушылардың бірдей қызмет атқаруы мүмкін, яғни Аты-жөні→Қызметі функционалдық тәуелділігі бар, ал кері функционалдық тәуелділік жоқ.

Әрбір оқытушы тек бір ғана кафедраның қызметкері болып табылады. Сондықтан Аты-жөні→Кафедра функционалдық тәуелділігі орындалады. Басқа жағынан, әрбір кафедрада көптеген оқытушылар бар, сондықтан кері қарай функционалдық тәуелділік жоқ.

Әрбір оқытушыға нақты жалақы сәйкес келеді, ол бірдей қызмет атқаратын барлық оқытушылар үшін бірдей, яғни Аты-жөні→Жалақысы және Қызметі→Жалақысы тәуелділіктері орындалады. Әртүрлі қызметтер үшін бірдей жалақылар жоқ сондықтан Жалақысы→Қызметі функционалдық тәуелділік орынды.

Бір оқытушы бір топта әртүрлі пәндерден әртүрлі сабақ өткізуі мүмкін. Оқытушы өткізетін сабақ түрін пәннің атын және топтарды көрсетпей анықтау мүмкін емес, Аты-жөні, Пәні, Тобы→Сабақ түрі функционалдық тәуелділігі бар. Шындығында, Петров М.И. 256-топта дәріс оқиды және практикалық сабақтар жүргізеді. Бірақ, дәрісті ол ДҚБЖ пәні бойынша оқиды, ал практикалық сабақты Паскаль бойынша жүргізеді.

Біз Аты-жөні, Пәні және Тобы атрибуттары арасындағы тәуелділіктерді көрсетпедік, себебі олар құрама кілт құрастырады және бастапқы қатынастарды нормалау процесінде ескерілмейді.

Барлық функционалдық тәуелділіктер анықталғаннан кейін оларды ОҚЫТУШЫ қатынастарында (5.4-сурет) бастапқы деректермен келісушіліктерге тексеру керек.

Мысалы, Қызметі='оқытушы' және Жалақысы=500 барлық кортеждерде әрқашан бір-бірінмен сәйкес келеді, яғни Қызметі.↔Жалақысы функционалдық тәуелділігі бекітіледі. Осылайша қалған функционалдық тәуелділіктерді деректердің шектеулерін ескере отырып, қарастыру керек

Қалыпты формалар

ДҚ жобалау процесінде қалыпты формалар әдісін қолдану итерациялық болып табылады және қатынастардың бірінші қалыпты формаларынан қайсыбір ережелер бойынша неғұрлым жоғары ретті қалыпты формаларға өтуінен тұрады. Әрбір келесі қалыпты форма функционалдық тәуелділіктердің қайсыбір түрін шектейді, қатынастармен орындалатын операциялар кезінде сәйкес ауытқушылықтарды жояды және алдыңғы қалыпты форманың қасиеттерін сақтайды.

Қалыпт формалардың келесі тізбегін бөліп көрсетеді:

- бірінші қалыпты форма (1ҚФ);
- екінші қалыпты форма (2ҚФ);
- үшінші қалыпты форма (3ҚФ);
- күшейтілген үшінші қалыпты форма немесе Бойса-Кодда қалыпты формасы (БКҚФ)
- төртінші қалыпты форма (4ҚФ);
- бесінші қалыпты форма (5ҚФ).

Бірінші қалыпты форма. Егер оның барлық атрибуттары қарапайым болса (жалғыз мәні болса), онда 1-қатынасқа орналасады. Бастапқы қатынас осылайша құрылады.

Қатынастарды келесі қалыпты формаға ауыстыру "жоғалтуларсыз декомпозиция" әдісімен жүзеге асады. Мұнда бастапқы қатынасқа және қатынастарға декомпозиция нәтижесінде алынған сұраныстар бірдей нәтиже береді.

Әдістің негізгі операциясы кескіндеу болып табылады. Оны мысалмен түсіндірейік. R қатынастарында (A,B,C,D,E,...) C→D функционалдық тәуелділігін жою оның келесі қалыпты формаға ауысуына жол берсін. Бұның шешімін табу үшін R қатынастарды екі жаңа R1(A,B,C,E,...) және R2(C,D) қатынастарға декомпозициялаймыз. R2 қатынасы R қатынастарының C және D атрибуттарына проекциясы болып табылады.

ОҚЫТУШЫ бастапқы қатынасын иллюстрациялау үшін әдіс қолданамыз; Аты-жөні, Пәні, Тобы құрама кілті бар және 1ҚФ орналасқан, себебі оның барлық атрибуттары қарапайым.

Бұл қатынаста 5.1-суретке сәйкес Жұмыс өтілі, Y_Жұмыс өтілі, Кафедра, Қызметі, Жалақысы атрибуттарының кілттен тәуелді бөлімдерін бөліп көрсетуге болады – көрсетілген атрибут Аты-жөні атрибутынан ішінара функционалды тәуелді.

Бұл ішінара функционалды тәуелділіктен келесілер туындайды:

1. Қатынастарда деректердің анық және анық емес артық қосарланушылығы бар, мысалы:

- бірнеше топтарда не/немесе әртүрлі пәндер бойынша сабақ жүргізетін оқытушылардың жұмыс өтілі, қызметі және жалақысы туралы мәліметтердің қайталануы;
- бірдей қызметі үшін жалақы туралы немесе бірдей жұмыс өтіліне үстеме туралы мәліметтердің қайталануы.

2. деректердің артық қосарланушылығының салдары оларды өңдеу мәселесі болып табылады. Мысалы, оқытушы Иванов И.М. қызметіндегі өзгерістер осы оқытушы туралы мәліметтерді қамтитын барлық қатынастар кортежін көруді және оларға өзгерістер енгізуді талап етеді.

Артық қосарланушылық 2ҚФ ауыстыру кезінде жойылады.

Екінші қалыпты форма. Егер форма 1ҚФ орналасқан болса және әрбір кілттік емес атрибут алғашқы кілттен (құрамынан) функционалды толық тәуелді болса, онда қатынас 2ҚФ орналасады.

Ішінара тәуелділікті жою және 2ҚФ қатынастарға ауысу үшін кескіндеу операциясын пайдаланып, оны келесі түрде бірнеше қатынастарға жіктеу қажет:

- алғашқы кілттен ішінара функционалдық тәуелділікте орналасқан атрибуттары жоқ проекция тұрғызу;
- алғашқы кілтті қамтитын бөлімдер және осы бөліктерден тәуелді атрибуттарды кескіндеу.

Нәтижесінде 2ҚФ формада екі R1 және R2 қатынастар (5.6-сурет) алынды.

а)

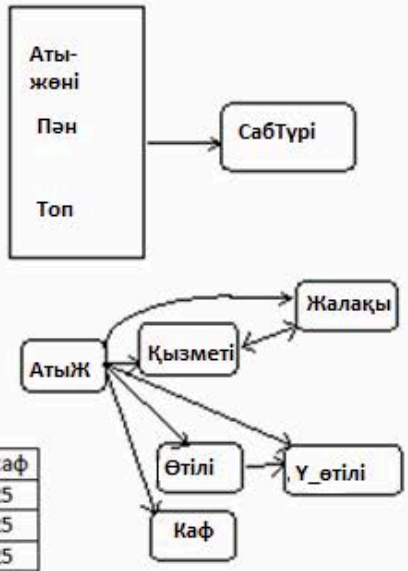
R1

Аты-жөні	Пән	Топ	Саб. түрі
Иванов И.М.	ДҚБЖ	256	Практика
Иванов И.М.	ПЛ/1	123	Практика
Петров М.И.	ДҚБЖ	256	Дәріс
Петров М.И.	Паскаль	256	Практика
Сидоров Н.Г.	ПЛ/1	123	Дәріс
Сидоров Н.Г.	Паскаль	256	Дәріс
Егоров В.В.	ДЭЕМБ	244	Дәріс

R2

Аты-жөні	Қызметі	Жал	Ж. өтілі	Ү_өтілі	каф
Иванов И.М.	Оқытушы	500	5	100	25
Петров М.И.	Аға оқыт	800	7	100	25
Сидоров Н.Г.	Оқытушы	500	10	150	25
Егоров В.В.	Оқытушы	500	5	100	24

б)



5.6-сурет. ДҚ қатынастары 2ҚФ формада

R1 қатынастарында алғашқы кілт құрама болып табылады және Аты-жөні, Пәні, Тобы атрибуттарынан тұрады. Ескере кетейік, берілген кілт R1 қатынастарында, әрбір оқытушы бір топта бір пән бойынша дәріс оқуы немесе практикалық сабақтар жүргізуі мүмкін деген болжам арқылы алынған. R2 қатынастарындағы кілт Аты-жөні.

R1 және R2 қатынастарын зерттеу нәтижесі 2ҚФ-ға өту R2-кестеден айқын артықтық деректерді – оқытушылар туралы мәліметтері бар қайталанатын жолдарды алып тастауға мүмкіндік беретінін көрсетеді. R2 қатынасында бұрынғыдай деректердің анық емес қосарланушылығы бар.

Қатынастарды бұдан әрі жетілдіру үшін оны 3ҚФ түрлендіру қажет.

Үшінші қалыпты форма.

1-анықтама. Егер форма 2ҚФ орналасқан болса және әрбір кілттік емес атрибут алғашқы кілттен транзитивті емес тәуелді болса, онда қатынас 3ҚФ орналасады.

Бұған балама анықтама бар.

2-анықтама. Егер барлық кілттік емес қатынастар атрибуттары өзара тәуелсіз және алғашқы кілттен толық тәуелді болса ғана қатынас 3ҚФ орналасады.

Бұл тұжырымның дұрыстығын дәлелдеу қиын емес. Шындығында, кілттік емес атрибуттардың алғашқы кілттен толық тәуелділігі берілген қатынастың 2ҚФ формада орналасатынын білдіреді. Атрибуттардың өзара тәуелсіздігі (жоғарыда келтірілген анықтама бойынша) қатынастарда қайсыбір атрибуттар арасындағы тәуелділіктердің, сонымен қатар олардың арасында транзитивті тәуелділіктің де болмауын білдіреді. Осылайша, екінші анықтама бірінші анықтамаға келеді.

Егер R1 қатынастарында транзитивтік тәуелділік болмаса, онда олар R2 қатынастарында бар:

Аты-жөні→Қызметі→Жалақысы,

Аты-жөні→Жалақысы→Қызметі,

Аты-жөні→Жұмыс өтілі→Ү_жұмыс өтілі

Транзитивтік тәуелділік, сонымен қатар, қатынастарда ақпараттардың артық қосарланушылығын тудырады. Оларды жояйық. Осы үшін транзитивті тәуелділіктердің себебі болып табылатын атрибуттарға кескіндеу операциясын пайдаланып, R2 қатынасын өзгертеміз. Сонымен қатар, әрқайсысы 3ҚФ формада орналасқан R3, R4

және R5, қатынастарын аламыз (5.3а-сурет). Графикалық тұрғыдан бұл қатынастар 5.6б-суретте көрсетілген. Ескере кетейік, R2 қатынасын басқаша түрлендіруге болады, нақтырақ айтқанда: R3 қатынастарында Қызметі атрибуты орнына Жалақысы атрибутын алу.

а)

R3

Аты жөні	Қызмет	Өтіл	Каф
Иванов И.М.	Оқыт	5	25
Петров М.И.	Аға оқыт	7	25
Сидоров Н.Г.	Оқыт	10	25
Егоров В.В.	Оқыт	5	24

R4

Қызмет	жалақы
Оқыт	500
Аға оқыт	800

R5

Жұм өтілі	Y_ Өтіл
5	100
7	100
10	150

б)

5.6-сурет. ДҚ қатынастары 3ҚФ формада

Іс жүзінде қатынастар сызбасының 3ҚФ тұрғызу көбінесе жеткілікті болып табылады және осымен реляциялық ДҚ жобалау процесі аяқталады. Шындығында, біздің мысалда қатынастарды 3ҚФ келтіру артық қосарланушылықты жойды.

Егер қатынастарда кілттік атрибуттар арасында кілттік емес атрибуттардан тәуелділік бар болса, онда күшейтілген 3ҚФ ауысу қажет.

Күшейтілген 3ҚФ немесе Бойса-Кодда қалыпты формасы (БКҚФ).

Егер қатынас 3ҚФ формада орналасса және онда кілттік емес атрибуттардан тәуелділік болмаса, онда ол БКҚФ формасында орналасады.

Бізде мұндай тәуелділік жоқ, сондықтан жобалау процесі осымен аяқталады. Жобалау нәтижесі келесі кестелерден тұратын ДҚ болып табылады: R1, R3, R4, R5. Алынған ДҚ-да деректердің қосарланушылығы бар, бірақ артық қосарланушылық жоқ.

Төртінші қалыпты форма.

Жаңа ЖОБАЛАР қатынастары мысалын қарастырайық, оның сызбасы келесі түрде берілсін: ЖОБАЛАР (Жоба_номері, Қызметкер_коды, Қызметкер_тапсырмасы). Атрибуттардың барлық жиынтығы алғашқы кілттік қатынастар болып табылады: Жоба_номері, Қызметкер_коды және Қызметкер_тапсырмасы.

Қатынастарда жоба номері, әрбір жоба үшін – қызметкер-орындаушы кодтары тізімі, сонымен қатар әрбір жобада қарастырылған тапсырмалар тізімі болады. Қызметкерлер бірнеше жобаға қатыса алады және әртүрлі жобаларда бірдей тапсырмалар болуы мүмкін. Қайсыбір жобаға қатысатын әрбір қызметкер бұл жоба бойынша барлық тапсырмаларды орындайды (бұл жобалау әрқашан дұрыс болмайды, бірақ біздің мысал үшін тиімді).

Тапсырманың мұндай берілуінде қатынастардың жалғыз мүмкін кілті Жоба_номері, Қызметкер_коды, Қызметкер_тапсырмасы құрама атрибуттары болып табылады. Ол, әрине, алғашқы кілттік қатынастар болады. Бұдан шығатыны, ЖОБАЛАР қатынасы БКҚФ формасында орналасады.

Бастапқы ақпарат бұл қатынаста келесі түрде берілсін:

ЖОБАЛАР

Жоба номері	Қызметкер коды	Қызметкер тапсырмасы
001	05	1
001	05	2
001	05	3
004	02	1
004	02	2
004	03	1
004	03	2
004	05	1
004	05	2
007	06	1

ЖОБАЛАР қатынастарының негізгі кемшіліктері: қайсыбір қызметкерді жобаға енгізу/шығару кезінде жобада қанша тапсырма болса сонша кортеждерді қатынастарға қосу/шығару керек. Қайсыбір қатынастарға қызметкер туралы мәліметті енгізу немесе жою кортежде қайталанатын мәндер болғандықтан, бірнеше элементар операцияларды орындауды талап етеді.

Келесі сұрақтар туындауы мүмкін: «Кортежде қызметкерлер кодының қайталанатын мәнін сақтаудың қажеті қанша? Әрбір жобалау бойынша және әрбір қызметкер-орындаушы үшін барлық тапсырмаларды қарастыру қажет па? Ақпаратты жоба тапсырмаларына енгізу кезінде негізгі кестеден қайталанатын мәндерді алып тастауға болмайды ма?»

Ескере кетейік, бұл кестеде ақпараттардың қосарланушылығынан туындайтын ауытқушылықтар. Ауытқушылықтардың болу себебі қайсібір атрибуттар арасындағы тәуелділіктердің (олардың көпмәнді тәуелділік екендігін көреміз) бар болуы деп есептейік.

Шындығында, ЖОБАЛАР қатынастарында келесі екі көпмәнді тәуелділік бар:

Жоба_номері⇒Қызметкер_коды
Жоба_номері⇒Қызметкер_тапсырмасы

Ерікті R(A, B, C) қатынасында бір уақытта көпмәнді тәуелділік A⇒B және A⇒C болуы мүмкін. Бұны A⇒B|C түрінде белгілейік.

Бұдан былайғы ЖОБАЛАР қатынасына ұқсас қатынастарды қалыпқа келтіру келесі теоремаға негізделген.

Фейджина теоремасы (Fagin R.). Егер A⇒B|C тәуелділігі бар болса, онда R(A, B, C) қатынасын R1(A, B) және R2(A, C) қатынастарға мәндерін жоғалтпай жобалауға болады.

Мәндерін жоғалтпай жобалау дегеніміз қатынастарды декомпозициялау тәсілі, мұнда бастапқы қатынас алынған қатынастармен *табиғи байланысу* арқылы толық және артық емес түрде қалпына келеді (3.6-бөлімді қараңыз).

Мәндерін жоғалтпай жобалауды мысал ретінде түсіндірейік.
Қарапайым R(A, B, C) қатынасы келесі түрде берілсін:

R

A	B	C
К	15	1
К	15	2
Л	10	1
М	20	1
М	20	2
М	20	3

R1 және R2 атрибуттарға сәйкесінше A, B және A, C кескіндеулер тұрғызамыз. Олар келесі түрде болады:

R1

A	B
К	15
Л	10
М	20

R2

A	C
К	1
К	2
Л	1
М	1
М	2
М	3

Бинарлық қатынастар *байланысы* операциясының нәтижесі R1(A, B) және R2(A, C) атрибуты бойынша A болып табылады, A, B және C атрибуттарымен тернарлық қатынас нәтижесі R1 және R2 қатынастарын байланыстыру жолымен 1:M типі бойынша мәндерді сәйкестендіру негізінде A атрибуты болады (3.3-бөлім).

Ендеше, (к, 15) және {(к, 1), (к, 2)} кортеждерді байланыстыру {(к, 15, 1), (к, 15, 2)} кортеждерін береді.

R1(A, B) және R2(A, C) байланыстыру бастапқы R(A, B, C) қатынасын тудыратынын көру қиын емес. R қатынастарында артық кортеждер жоқ, жоғалтулар да жоқ.

Төртінші қалыпты формалар анықтамасы. Егер көпмәнді тәуелділік A⇒B орындалса, ал барлық қалған R атрибуттары A атрибутынан функционалды тәуелді болса, онда R қатынасы төртінші қалыпты формада (4ҚФ) орналасады.

Жоғарыда келтірілген ЖОБАЛАР қатынасын екі қатынастар түрінде көрсетуге болады: Жобалар-Қызметкерлер және Жобалар-Тапсырмалар. Бұл қатынастар келесі құрылымға ие:

Жобалар-Қызметкерлер (Жоба_номері, Қызметкер_коды).
Алғашқы кілттік қатынастар: Жоба_номері, Қызметкер_коды.

Жобалар-Тапсырмалар (Жоба_номері, Қызметкер_тапсырмасы).
Алғашқы кілттік қатынастар:Жоба_номері , Қызметкер_тапсырмасы

Сәйкес кестелер құрамы мына түрге келеді:

Жобалар-Қызметкерлер

Жоба номері	Қызметкер коды
001	05
004	02
004	03
004	05
007	06

Жобалар-Тапсырмалар

Жоба номері	Қызметкер тапсырмасы
001	1
001	2
001	3
004	1
004	2
007	1

Осы қатынастардың екеуі де 4ҚФ формада орналасқанын және аталған кемшіліктері жоқ екендігін оңай көруге болады. Қызметкер кодында атрибут мәндері қосарланушылығы жойылды. Бұл қатынастарды өз бетіңізше жалғастырып көріңіз және ЖОБАЛАР қатынасы алынатынына көз жеткізіңіз.

Жалпы жағдайда, кез-келген қатынасты бастапқы қалпына келтіру мүмкін емес. Біздің жағдайда қалпына келтіру мүмкін, себебі қайсыбір жобаға қатысатын әрбір қызметкер бұл жоба бойынша барлық тапсырмаларды орындады (байланыстық қатынастардың 1:М принципі). Қызметкерлердің өзі бірнеше жобаға қатысты және әртүрлі жобаларда бірдей тапсырмалар болуы мүмкін.

Бесінші қалыпты форма.

Барлық алдыңғы қатынастарды қалыпқа келтіру сызбасы нәтижесі екі жаңа қатынастар болады. Кей жағдайда мұны жасау мүмкін емес немесе алынатын қатынастар жағымсыз қасиеттерге ие. Бұл жағдайда бастапқы қатынастарды саны екіден көп қатынастарға декомпозициялайды.

Қызметкерлер-Бөлімдер-Жобалар қатынасын қарастырайық, онда келесі тақырыптар бар: Қызметкерлер-Бөлімдер-Жобалар (Қызметкер_коды, Бөлім_коды, Жоба_номері). Алғашқы кілттік қатынастар барлық атрибуттарды іске қосады: Қызметкер_коды, Бөлім_коды және Жоба_номері. Бұл қатынаста бір қызметкер бірнеше бөлімдерде жұмыс жасауы мүмкін, сонымен қатар ол әрбір бөлімде бірнеше жобаға қатысуы мүмкін. Бір бөлімде бірнеше қызметкерлер жұмыс жасауы мүмкін, бірақ әрбір жобаны тек қана бір қызметкер орындайды. Атрибуттар арасында функционалдық және көпмәнді тәуелділіктер жоқ болады.

Қызметкерлер-Бөлімдер-Жобалар

Қызметкер коды	Бөлімі коды	Жоба номері
01	ПХ	036
02	ӘК	004
03	ӘӨ	004
04	ӘК	019
05	СБ	001
05	СБ	004
06	ӘӨ	007
08	ЕО	013
09	ЕО	014
10	ӨМ	013

Бұл қатынас ойдан шығарылған НИИЧАВО – Стругацких А.Б. «Дүйсенбі сенбіден басталады» әңгімесінен алынған ғылыми-зерттеу Институты сиқыршылық және арбаушылық ғылыми бөлімшесінің деректер қорының бөлімі болып табылады. Бөлімдер коды былай белгіленген: ӘК – Әкімшілік, ЕО – Есептеу орталығы, СБ – Сызықтық бақыт, ПХ – Перзентхана, ӨМ – Өмірдің мәні, ӘӨ – Әмбебап өзгерістер.

Қызметкерлер-Бөлімдер-Жобалар қатынастарының құрылымынан оның 4ҚФ формада орналасатынын көреміз. Дегенмен, бірнеше кортеждерде атрибуттар мәндері қайталануы мүмкін болғандықтан, қатынастарда ауытқушылықтар болуы мүмкін. Мысалы, қызметкер бірнеше бөлімдерде жұмыс жасауы мүмкін, қызметкерді жұмыстан шығарғанда оны бастапқы кестелердің бірнеше жазбаларынан тауып, өшіру талап етіледі.

Байланысқа тәуелділікке анықтама берейік.

Егер X, Y, ... , Z өз проекциясында R байланыстыру жолымен жоғалтуларсыз бастапқы қалпына келсе, онда R(X, Y, ... , Z) қатынасы *(X, Y, ... , Z) түрінде белгіленетін ***байланысқа тәуелділікті*** қанағаттандырады. Байланыстыруға тәуелділік функционалды және көпмәнді тәуелділіктердің жалпы жағдайы болып табылады.

Бесінші қалыпты формалар анықтамасы. Егер R қатынасында кез-келген байланыстыруға тәуелділік R қатынасында қайсыбір мүмкін кілттің бар болуынан шықса, онда R қатынасы 5ҚФ формада (немесе қалыпты кескіндеу-байланыстыру – PJ/NF формасында) орналасады.

Қызметкерлер-Бөлімдер-Жобалар қатынастары атрибуттарының құрамын қалыптастырайық:

СО={ Қызметкер_коды, Бөлім_коды }
СП={ Қызметкер_коды, Жоба_номері }
ОП={ Бөлім_коды, Жоба_номері }.

Егер Қызметкерлер-Бөлімдер-Жобалар қатынасын СО, СП және ОП атрибуттарына кескіндесек, онда осы кескін байланысы бастапқы қатынасты беретінін көрсетейік. Бұл біздің жағдайда байланыстыруға тәуелділік *(СО, СП, ОП) бар екенін білдіреді. Атрибуттарға кескіндеуді сәйкесінше Қызметкерлер-Бөлімдер, Қызметкерлер- Жобалар және Бөлімдер - Жобалар деп атайық.

Қызметкерлер-Бөлімдер

Қызметкер коды	Бөлім коды
01	ПХ
02	ӘК
03	ӘӨ
04	ӘК
05	СБ
06	ӘӨ
08	ЕО
09	ЕО
10	ӨМ

Қызметкерлер- Жобалар

Қызметкер коды	Жоба номері
01	036
02	004
03	004
04	019
05	001
05	004
06	007
08	013
09	014
10	013

Бөлімдер - Жобалар

Бөлімі коды	Жоба номері
ӘК	004
ӘК	019
ЕО	013
ЕО	014
СБ	001
СБ	004
ПХ	036
ӨМ	013
ӘӨ	004
ӘӨ	007

Біз бұрын орындаған екі проекцияны байланыстырудан ізделінді нәтиже алдық. Үш (немесе бірнеше) проекциядан тұратын қатынастарды қалпына келтіру үшін барлық жұптық байланыстыруларды (себебі олардың қайсысы «жақсырақ» екендігі туралы ақпарат жоқ) алу керек, содан соң олармен жиындардың қиылысуы операциясын орындайды. Тексерейік.

Жоғарыда келтірілген үш қатынастардан келесі түрдегі жұптық байланыс аламыз:

***(СО, СП)**

Қызметкер коды	Бөлім коды	Жоба номері
01	ПХ	036
02	ӘК	004
03	ӘӨ	004
04	ӘК	019
05	СБ	001
05	СБ	004
06	ӘӨ	007
08	ЕО	013
09	ЕО	014
10	ӨМ	013

***(СП, ОП)**

Қызметкер коды	Бөлім коды	Жоба номері
01	ПХ	036
02	ӘК	004
02	СБ	004
02	ӘӨ	004
03	ӘК	004
03	СБ	004
03	ӘӨ	004
04	ӘК	019
05	СБ	001
05	ӘК	004
05	СБ	004
05	ӘӨ	004
06	ӘӨ	004
06	ӘӨ	007
08	ЕО	013
08	ЕО	014
09	ЕО	013
09	ЕО	014
10	ӨМ	013

*(СО, ОП)

Қызметкер коды	Бөлім коды	Жоба номері
01	ПХ	036
02	ӘК	004
02	ӘК	019
03	ӘӨ	004
03	ӘӨ	007
04	ӘК	004
04	ӘК	019
05	СБ	001
05	СБ	004
06	ӘӨ	004
06	ӘӨ	007
08	ЕО	013
08	ВЦ	014
09	ВЦ	013
09	ВЦ	014
10	СЖ	013

Барлық қатынастардың қиылысуы бастапқы Қызметкерлер - Бөлімдер – Жобалар қатынасын беретінін көру қиын емес.

Ескерту.

Бастапқы қатынастарды оның проекциясынан қалпына келтірудің басқа да әдістері бар. Ендеше, Қызметкерлер-Бөлімдер-Жобалар қатынастарын қалпына келтіру үшін Қызметкер_коды атрибуты бойынша Қызметкерлер-Бөлімдер және Қызметкерлер-Жобалар қатынастарын біріктіру керек, содан соң алынған қатынасты құрама (Бөлім_коды, Жоба_номері) атрибутты бойынша Бөлімдер-Жобалар қатынасымен біріктіру керек.

Қызметкерлер-Бөлімдер, Қызметкерлер-Жобалар және Бөлімдер-Жобалар қатынастары 5ҚФ формада орналасқан. Бұл форма соңғы белгілі форма болып табылады. Оны алудың шарты жеткілікті түрде тривиальды емес және сондықтан ол іс жүзінде көп қолданылмайды. Сонымен қатар оның кемшіліктері көп.

02 кодты қызметкердің қандай жобаларды орындайтынын білу керек болсын. Бұл үшін Қызметкерлер-Бөлімдер қатынастарынан 02 кодты қызметкердің ӘК бөлімінде жұмыс жасайтынын табамыз, ал Бөлімдер-Жобалар қатынастарынан ӘК бөлімінде 004 және 019 жобалар орындалатынын көреміз. Ал бұл, 02 кодты қызметкердің 004 және 019 жобаларын орындауы керек екендігін білдіреді. Бірақ, 02 кодты қызметкердің 019 жобаны орындауы туралы ақпарат жалған (бастапқы Қызметкерлер-Бөлімдер-Жобалар қатынасынан қараңыз).

Ондай қайшылықтарды негізгі қатынастардың барлық проекцияларын бірлесе қарастыру жолымен ғана жоюға болады. Сондықтан, проекцияларды байланыстырудан кейін оларды қиылыстыру операциясын орындау керек. Әрине, бұл – кемшілігі. Ескере кетейік, кемшіліктердің бар болуы аталған қалыпты формадан мүлдем бас тартуды талап етпейді. Тек қана кемшіліктерді және олардың салдарын ескеру қажет. Кейбір тапсырмалардың қойылымында кемшіліктер орын алмайды.

Іс жүзінде әдетте ДҚ құрылымы сәйкес 3ҚФ немесе БКҚФ шектеледі. Сондықтан, қатынастарды қалыпты формаға келтіру бастапқы қатынастардан атрибуттар арасындағы келесі тәуелділіктерді жоюды жобалайды:

- кілттік емес атрибуттардың кілттік атрибуттардан ішінара тәуелділіктері (2ҚФ талаптарын қанағаттандыру);
- кілттік емес атрибуттардың кілттік атрибуттардан транзитивті тәуелділіктері (3ҚФ талаптарын қанағаттандыру);
- кілттік атрибуттардың (кілттің құрамындағы атрибуттар) кілттік емес атрибуттардан тәуелділігі (БКҚФ талаптарын қанағаттандыру).

Код қалыпты форма әдісінен басқа, кішігірім ДҚ жобалау үшін басқа әдістер де қолданылады, мысалы, ER-диаграмма («Мән-байланыс» әдісі) әдісі. Бұл әдіс үлкен ДҚ жобалау кезінде қолданылады, ДҚ жобалау құралдарына негізделген. ER-диаграмма әдісі келесі бөлімде қарастырылады.

ER-диаграмма әдісінің соңғы кезеңіндегі жобалау нәтижесі олардың БКҚФ формада болуын тексереді. Бұл кезең қалыпты форма әдісін қолданып орындалуы мүмкін. Жобалау аяқталғаннан кейін ДҚ көмегімен ДҚБЖ құрылады.

5.3. Құрылымдарды құруға ұсыныстар

Алдыңғы бөлім материалдарын жалпылау үшін неғұрлым маңызды ұсыныстар келтірейік, оларды ескермеу деректер қорында деректерді өңдеу кезінде ауытқушылықтарға соқтыруы мүмкін. Екі мәселені қарастырайық: қатынастарды (кестелер) қалай құруға болады және оларды қалай байланыстыруға болады.

Мәндік кестелер құрған кездегі *негізгі ереже* – бұл «әрбір болмысқа – жеке кесте» («әрбір жанұяға – жеке пәтер» ұраны сияқты).

Мәндік кестелер өрістері екі түрлі болуы мүмкін: *кілттік* және *кілттік емес*. Кестеге кілттер кірістіру, практикалық тұрғыдан, барлық реляциялық ДҚБЖ-да кесте жазбаларындағы мәндердің кілт бойынша бірегейлігін қамтамасыз ету, кестелердегі жазбаларды өңдеуді жылдамдатуға мүмкіндік береді, сонымен қатар кілттік өрістердің мәні бойынша жазбаларды автоматты түрде сұрыптауды орындайды.

Әдетте *қарапайым* кілтті анықтау жеткілікті, *құрама* кілт аз қолданылады. Кестелер құрама кілтті болуы мүмкін, мысалы, қызметкерлер (фамилиясы, аты және әкесінің аты) тізімі кестесі, онда фамилиясы бірдей қызметкерлер кездесуі мүмкін. Қайсыбір ДҚБЖ-да пайдаланушыларға автоматты түрде құрылатын кілттік өрісті нөмерлеуді (Access бағдарламасында – бұл «санауыш» типті өрісі) анықтау ұсынылады, олар кестелердегі жазбалардың бірегейлік мәселелерін шешуді жеңілдетеді.

Кей жағдайда мәндік кестелерде объектілердің сипаттамасын анықтау өрістері бар. Егер кестеде осы өрістер бойынша бірнеше қайталанушылықтар орын алса және бұл ақпаратты көлемі үлкен болса, онда оларды жеке кестеге («әрбір болмыстар – жеке кесте» ережесін сақтап) жазған дұрыс. Сонымен қатар, егер қасиеттер өзара байланысты болса, онда қосымша кесте құрамыз.

Жалпы жағдайда соңғы ұсыныстарды келесі түрде жазуға болады: мәндер туралы ақпаратты кестелердің кілттік өрістері өзара тәуелсіз болатындай және кілттен толық тәуелді болатындай (үшінші қалыпты формалар анықтамасын қараңыз) беру керек.

Мәндік кестелерді өңдеу процесінде келесілерді ескеру керек. Кірістіру (қосу) және өзгерту оңай, бірақ өшіру кезінде – кесте байланыстарынан барлық оған сілтемелерді жою керек, әйтпесе кестелер байланысы дұрыс емес болады. Көптеген заманауи ДҚБЖ мұндай операциялардың орындалуына жол бермейді.

Мәндік байланыстарды ұйымдастыру

Кестелер байланысы жазбасы мәндер арасындағы байланысты көрсетуге, олардың сәйкес мәндер кестесінде орналасуы туралы ақпаратқа арналған.

Әдетте бір кесте байланысы екі мәннің өзара байланысын сипаттайды. Себебі мәндік кестелердің қарапайым жағдайда бір кілттік өрісі бар, ал екі кесте байланысын

қамтамасыз ету үшін кестеде байланыстар туралы бірегей жазбалардың екі кілті болуы керек. Кесте обьектісі ретінде және кілтсіз байланыс кестесін құру мүмкін, бірақ онда жазбалар бірегейлігін бақылау функцияларын пайдаланушының өзі атқарады.

Неғұрлым күрделі байланыстарды (бинарлық емес) бинарлық байланысқа келтіру керек. N объектілердің өзара байланысын сипаттау үшін кестеден N-1 байланыстары талап етіледі. Транзитивті байланыс болмауы керек. Байланыстардың артықтығы қайшылықтарға (мысалы, Қызметкерлер-Бөлімдер, Қызметкерлер -Жобалар және Бөлімдер - Жобалар қатынастары) апарады.

Кестелер байланысы мәндер сипаттамаларын қамтымауы керек, әйтпесе ауытқушылықтар туындайды. оларды жекелеген мәндер кестесінде сақтаған дұрыс.

Кесте байланысы көмегімен бірнеше маманданған байланыстар түрін сипаттауға болады – сызықтық байланыс немесе әлсіз байланыс. Сызықтық байланыстар мысалы ретінде неғұрлым жоғары ретті қайсыбір басқа болмыстарға (түйіндерден тұратын жүйелер; компоненттерден тұратын дәрілер; металлдар құймасы және т.б.) жатуын көрсететін қатынасты есептеуге болады. Бұл жағдайда байланысты сипаттау үшін бір кестенің байланысы жеткілікті.

Кестелер байланысымен жұмыс жасау кезінде кестелерден кез-келген байланыс жазбасы оңай жойылуы мүмкін екенін ескеру керек, себебі болмыстар қайсыбір уақытта байланыссыз болуы мүмкін. Кестелерге жазбалар қосу немесе өзгерту кезінде қолданыстағы объектілерге сілтемелердің дұрыстығын бақылау керек, өйткені объектілерсіз байланыстың болуы мүмкін емес. Көптеген заманауи ДҚБЖ объектілерге сілтемелердің дұрыстығын бақылайды.

5.4. Тұтастықты қамтамасыз ету

Тұтастық дегеніміз деректер қорының қасиеті, ол оның толық, қайшылықсыз және пәндік облысты адекватты бейнелейтін ақпараттары бар екенін білдіреді.

Физикалық және *логикалық* тұтастықты ажыратады. Физикалық тұтастық деректерге физикалық қатынас құруда деректердің жоғалмағандығын білдіреді. Логикалық тұтастық деректер қорында логикалық қателіктердің болмауын білдіреді, оларға жататындар: ДҚ немесе оның объектілерінің құрылымының бұзылуы, объектілер арасында орнатылған байланыстарды өшіру немесе өзгерістер енгізу және т.б. Бұдан былай логикалық тұтастықты қарастырамыз.

ДҚ тұтастығын сүйемелдеу тұтастықты және оның қалпына келтіру жағдайында қордағы табылған қайшылықтарды тексеруді іске қосады. ДҚ тұтастығы ДҚ-на *тұтастыққа шектеу қою* көмегімен шарттар түрінде беріледі, оларды қорда сақталатын деректер қанағаттандыруы тиіс.

Тұтастыққа шектеу қою арасында шектеулердің екі негізгі түрін бөліп көрсетуге болады: қатынастардың *атрибуттары мәндеріне шектеу* және қатынастар кортеждеріне *құрылымдық шектеулер.*

Қатынастардың ***атрибуттары мәндеріне шектеу*** мысалы ретінде атрибуттарда бос немесе қайталанатын мәндерге жол беруге болмайтын талабын, сонымен қатар атрибуттар мәндері берілген аралықта жататындығын тексеруді атауға болады. Ендеше, жазбалардағы кадрлар туралы қатынастарда Туған_күні атрибуты мәні Қабылдау_уақыты атрибутының мәнінен аспауы керек.

Атрибуттар мәндерін бақылауды жүзеге асырудың неғұрлым ыңғайлы құралдары қайсыбір ДҚБЖ-да бар *сақталатын процедуралар мен триггерлер* болады.

Құрылымдық шектеулер *мәндер тұтастығы* және *сілтемелер тұтастығы* талаптарын анықтайды. Қатынастарда келтірілген әрбір болмыстар үлгісіне тек қана бір кортеж сәйкес келеді. *Мәндер тұтастығын* талап ету қатынастардың кез-келген

кортежі осы қатынастардың кез-келген басқа кортежінен өзгеше болуы керектігінен тұрады, яғни кез-келген қатынастың алғашқы кілті болуы керек.

Сілтемелер тұтастығы талабын қалыптастыру *сыртқы кілттің* түсінігімен тығыз байланысты. Ескере кетейік, сыртқы кілттер қатынастар арасындағы байланыстар (ДҚ кестесі) үшін керек. Сонымен қатар, егер кілт басқа қатынастардың *алғашқы кілті* болып табылса, онда бір қатынастардың (түпкі) атрибуты берілген қатынастардың ***сыртқы кілті*** деп аталады. Сыртқы кілт анықталған қатынас осы атрибут алғашқы кілт болып табылатын қатынасқа сілтеме жасайды.

Сілтемелер тұтастығы талабы бойынша сыртқы кілттің әрбір мәні үшін түпкі кестелер ішкі кестеден алғашқы кілттік мәні сәйкес келетін жолды табу керек. Мысалы, егер R1 қатынастарында (5.7-сурет) кафедра қызметкерлері туралы мәліметтер болса, ал бұл қатынастардың алғашқы кілттік атрибуты Қызметі болып табылса, онда бұл қатынаста әрбір қызметі атрибуты үшін сәйкес жалақы болуы керек.

5.7-сурет. Сыртқы кілттің көмегімен қатынастар байланысы

Көптеген заманауи ДҚБЖ-да ДҚ тұтастығын бақылауды қамтамасыз ету құралдары бар.

Бақылау сұрақтары және тапсырмалар

1. Деректердің құрылымын жобалау әдістерін атаңыз.
2. Артық қосарланушылықты түсіндіріңіз.
3. Ауытқушылықтардың негізгі түрін атаңыз және сипаттаңыз.
4. ДҚ жобалау кезінде бастапқы қатынас қалай қалыптасады?
5. Айқын және айқын емес артықтыққа мысал.
6. Қатынастар атрибуттары арасындағы тәуелділіктердің негізгі түрін атаңыз.
7. Функционалды және ішінара функционалды тәуелділіктерге мысал келтіріңіз.
8. Тәуелді атрибуттары бар қатынастарға мысал келтіріңіз.
9. Қалыпты формаларды сипаттаңыз.
10. Бірінші қалыпты формаларға анықтама беріңіз.
11. Екінші қалыпты формаларға анықтама беріңіз.
12. Үшінші қалыпты формаларға анықтама беріңіз.
13. Күшейтілген үшінші қалыпты формаларға анықтама беріңіз.
14. 4ҚФ талаптарын мысал арқылы түсіндіріңіз.
15. 5ҚФ талаптарын мысал арқылы түсіндіріңіз.
16. Мәндік кестелер құрудың негізгі ережесін айтыңыз.
17. Мәндік байланыстарды ұйымдастыру бойынша ұсыныстар айтыңыз.
18. ДҚ-ның физикалық және логикалық тұтастығына анықтама беріңіз.
19. Мәндерді шектеулер және құрылымдық шектеулерге мысал келтіріңіз.

20. Сыртқы және алғашқы кілт ұғымын түсіндіріңіз.

21. Автопарк жұмысын ұйымдастыруға арналған, ақпаратты көру және редактрлеуге рұқсат беретін ДҚ сызбасын құрыңыз. ДҚ жүргізушілер (транспорттық құралдар категориясы, жүргізушілік жұмыс өтілі, көлігі және т.б.) туралы, сонымен қатар автотұрақ машиналары (маркасы, шыққан жылы, техникалық жағдайы және т.б.) туралы мәліметтерді қамтысын. Кесте құрылымы мен олардың арасындағы өзара байланыстарды ұсыныңыз және түсіндіріңіз.

6. МӘН-БАЙЛАНЫС ӘДІСІ

Мән-байланыс әдісі «ER-диаграмма» әдісі деп те аталады: біріншіден, ER – *Essence (мағынасы) және Relation (байланыс)* сөзінен шыққан, екіншіден, әдіс ER-типті диаграмма қолдануға негізделген.

6.1. Әдістің негізгі ұғымдары

Мән-байланыс әдісінің негізгі түсініктері:

- мағынасы;
- болмыстар атрибуты;
- болмыстар кілті;
- мәндер арасындағы байланыс;
- байланыстар деңгейі;
- болмыстар үлгісі жататын санат;
- ER-үлгі диаграммасы;
- ER-типті диаграммалар.

ДҚ-да сақталатын объект туралы ақпарат *мағынасы* болып саналады. Байланыс атаулары, қағидаға сәйкес, *қолданыстағы* байланыстар болады, мысалы: Оқытушы, Пәні, Кафедра, Тобы.

Болмыстар туралы қасиеттер *атрибут* болып саналады. Бұл түсінік қатынастардағы атрибут түсінігіне ұқсас. Ендеше, Оқытушы болмыстар атрибуттары оның Фамилиясы, Қызметі, Жұмыс өтілі (оқытушылық) және т.с.с. болуы мүмкін.

Болмыстар кілті – болмыстар үлгілерін сәйкестендіру үшін қолданылатын атрибут немесе атрибуттар жиыны. Анықтамадан көргеніміздей, *болмыстар кілті* түсінігі қатынастар кілті ұғымына ұқсас.

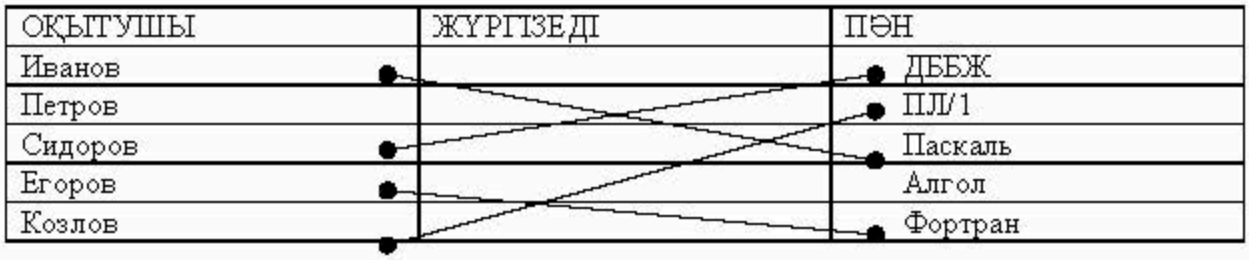
Екі немесе бірнеше болмыстар арасындағы байланыс осы болмыстар арасында атрибуттар тәуелділігін жобалайды. Байланыстар атауы әдетте *етістікпен* ұсынылады. Мәндер арасындағы байланыс мысалдары келесі түрде болсын: Оқытушы сабақ жүргізеді (Иванов «Деректер қоры» пәнінен сабақ береді), Оқытушы *топта сабақ береді* (Иванов 256-топта), Оқытушы кафедрада жұмыс жасайды (Иванов 25-кафедрада).

Келтірілген болмыстар және байланыстарды анықтау толық емес, бірақ іс жүзінде қолданылады. Бір ДҚ жобалау нәтижесінде бірнеше ДҚ алынуы мүмкін екенін ескеру керек. Ендеше, әртүрлі екі жобалаушы бір мәселені әртүрлі көзқараспен қарастыра отырып, болмыстар және байланыстардың әртүрлі жинаған алуы мүмкін. Сонымен қатар, екі нұсқа да қажеттілікті қанағаттандырса, онда олардың жақсысын таңдаған жөн.

Болмыстар көрінісі, болмыстар үлгісі және олардың арасындағы байланысты жобалау көрнекілігін және қолайлылығын арттыру мақсатында графикалық тұрғыдан келесі құралдар қолданылады:

- *ER-үлгі диаграммасы*,
- *ER-типті диаграммалар* немесе *ER-диаграммалар*.

6.1-суретте Оқытушы және Пәні болмыстары үшін *Жүргізеді* байланысымен ER-үлгі диаграммасы келтірілген.



6.1-сурет. ER-үлгі диаграммасы

ER-үлгі диаграммасы, қай нақты пәнді (ДҚБЖ, ПЛ/1 және т.с.с.) оқытушылардың қайсысы жүргізетінін көрсетеді. 6.2-суретте ER-үлгі диаграммасына сәйкес қарастырылған ER-типті диаграмма келтірілген.

6.2-сурет. ER-типті диаграмма

ДҚ жобалаудың алғашқы кезеңінде болмыстар кілттерін құраушы атрибуттарды бөліп көрсетеді.

ER-типті диаграммаларды талдау негізінде жобаланған ДҚ қатынастары қалыптасады. Сонымен қатар, өз кезегінде, ER-үлгілер диаграммасын талдау негізінде анықталатын сәйкес болмыстардың *мәндік байланыстар деңгейі* және *олардың жататын класы* ескеріледі.

Байланыстар деңгейі 1:1, 1:M, M:1, M:M типті мәндер арасындағы байланыстардың сипаттамасы болып табылады.

Болмыстар үлгісі жататын санат (БҮЖС) болуы мүмкін: *міндетті* және *міндетті емес*.

Егер болмыстардың барлық үлгілері қарастырылатын байланыстарға міндетті түрде қатысса, онда болмыстар үлгісі жататын санат *міндетті* болып табылады. Кері жағдайда болмыстар үлгісі жататын санат *міндетті емес* болып табылады.

Әрбір аталған типтік байланыстарда болмыстар үлгісі жататын санатты түрлендіре отырып, болмыстар үшін ER-типті диаграмманың бірнеше нұсқасын алуға болады. Олардың мысалын қарастырайық.

1-мысал. 1:1 типті байланыстар және міндетті емес болмыстар үлгісі жататын санат. 6.1-суретте келтірілген байланыстар деңгейі диаграммасында мәндер арасындағы байланыс 1:1, ал болмыстар үлгісі жататын санат екі болмыстарда да міндетті емес. Шындығында, суреттен келесілерді көруге болады:

- әрбір оқытушы бір пәннен сабақ береді, ал әрбір пәнді бір ғана оқытушы жүргізеді (байланыстар деңгейі 1:1);
- қайсыбір оқытушылар бірде-бір пәннен сабақ бермейді және бірде-бір оқытушы жүргізбейтін пәндер бар (болмыстар үлгісі жататын санат екі болмыстар үшін де міндетті емес).

2-мысал. 1:1 типті байланыстар және міндетті болмыстар үлгісі жататын санат. 6.3-суретте келтірілген диаграммадағы мәндер арасындағы байланыстар деңгейі 1:1, ал болмыстар үлгісі жататын санат екі болмыстар үшін де міндетті.

а) ER көшірмелері

ОҚЫТУШЫ	ЖҮРГІЗЕДІ	ПӘН АТАУЫ
Иванов		ДББЖ
Петров		ПЛ/1
Сидоров		Паскаль
Егоров		Алгол
Козлов		Фортран

б) ER типті

6.3-сурет. 1:1 байланыстары үшін диаграммалар және екі болмыстар үшін де міндетті БҮЖС

Бұл жағдайда әрбір оқытушы бір пәнді жүргізеді және әрбір пәнді бір ғана оқытушы жүргізеді.

Міндетті емес болмыстар үлгісіне жататын болмыстардың бірі санатты екі аралық нұсқа болуы мүмкін.

Ескертулер.

- ER-типті диаграммаларда байланыстардың *міндетті түрде* қатысатын болмыстар үлгісі блок ішіндегі нүкте арқылы белгіленеді (6.3б-сурет).
- Байланыстарда болмыстар үлгісі *міндетті емес түрде* қатысса болмыстар блогына қосымша блок салынбайды, ал нүкте байланыстар сызығында орналасады (6.2-сурет).
- Байланыстар сызығындағы символдар байланыстар деңгейін көрсетеді.

Қайсыбір болмыстарға сәйкес келетін әрбір блок астында оның кілті асты сызылып көрсетіледі. Кілттік атрибуттардағы көпнүкте болмыстардың басқа атрибуттарының болуы мүмкіндігін білдіреді, бірақ олардың біреуі де оның кілті болуы мүмкін емес. Бұл атрибуттар қатынастарды қалыптастырғаннан кейін беріледі.

Іс жүзінде байланыстар деңгейі және болмыстар үлгісі жататын санат ДҚ жобалау кезінде зерттеу облысының ерекшелігімен анықталады. 1:М немесе М:1 деңгейлі байланыстар нұсқасы мысалдарын қарастырайық.

3-мысал. 1:М типті байланыстар.

Әрбір оқытушы бірнеше пәндерді жүргізуі мүмкін, бірақ әрбір пәнді бір оқытушы ғана жүргізеді.

4-мысал. М:1 типті байланыстар.

Әрбір оқытушы бір пәннен сабақ береді, бірақ әрбір пәнді бірнеше оқытушылардың жүргізуі мүмкін.

1:М немесе М:1 типті байланыстар мысалының болмыстар үлгісі жататын санаттарымен ерекшеленетін бірнеше нұсқасы болуы үмкін. Міндетті болмыстар үлгісі жататын санатты «О» символымен, ал міндетті емес – «Н» символымен белгілейік, онда 1:М типті байланыстар нұсқаларын шартты түрде былай көрсетуге болады: О-О, О-Н, Н-О, Н-Н. М:1 типті байланыстар үшін 4 ұқсас нұсқа бар.

5-мысал. 1:М типті байланыстардың Н-О нұсқасы.

Әрбір оқытушы бірнеше пәндерді жүргізуі немесе ешбір пәнді жүргізбеуі мүмкін, бірақ әрбір пәнді бір оқытушы ғана жүргізеді (6.4-сурет).

а) ER көшірмелері

ОҚЫТУШЫ	ЖҮРГІЗЕДІ	ПӘН АТАУЫ
		ДББЖ
Иванов		ПЛ/1
Петров		Паскаль
Сидоров		Алгол
Егоров		Фортран
Козлов		C++
		Java

б) ER типті

6.4-сурет. 1:М типті байланыстар Н-О нұсқасы үшін диаграммалар

Осылайша басқа нұсқалар диаграммасын тұрғызу оңай.

6-мысал. М:М типті байланыстар.

Әрбір оқытушы бірнеше пәндерді жүргізуі мүмкін, ал әрбір пәннен бірнеше оқытушы сабақ береді.

Басқа байланыс типтері сияқты, М:М типті байланыстар үшін болмыстар үлгісі жататын санаттармен ерекшеленетін 4 нұсқа болуы мүмкін.

7-мысал. М:М типті байланыстар және болмыстар үлгісі жататын санат О-Н нұсқасы.

Әрбір оқытушы бірден кем емес пәндерден сабақ береді дейік, ал пәнді бірнеше оқытушының жүргізуі мүмкін. Сонымен қатар, ешкім жүргізбейтін пәндер бар. Осы жағдайға сәйкес диаграмма 6.5-суретте келтірілген.

ОҚЫТУШЫ	ЖҮРГІЗЕДІ	ПӘН АТАУЫ
		ДББЖ
Иванов		ПЛ/1
Петров		Паскаль
Сидоров		Алгол
Егоров		Фортран
Козлов		C++
		Java

б) ER типті

6.5-сурет. М:М типті байланыстар және О-Н нұсқасы үшін диаграммалар

Болмыстар және олардың арасындағы байланысты анықтау, сонымен қатар оларды ER-типті диаграмма негізінде қалыптастыру мән-байланыс әдісінің алғашқы кезеңдерінде орындалады. Әдістің жүзеге асу кезеңдерін қарастырайық.

6.2. Жобалау кезеңдері

Деректер қорын жобалау процесі бұрын қабылданған шешімдерді қайта қарастыру үшін алдыңғы кезеңге қайтуға мүмкіндік беретін және келесі кезеңдерді іске қосатын итерациялық процесс болып табылады:

1. Болмыстарды жіктеу және олардың арасындағы байланыс.
2. Барлық болмыстар және олардың байланысын ескере отырып ER-типті диаграмма тұрғызу.
3. Әрбір қатынастар және ER-типті диаграмма үшін қолданатын алғашқы кілтті көрсете отырып, бастапқы қатынастар жинағын қалыптастыру.
4. Қатынастарға кілттік емес атрибуттар кірістіру немесе қосу.
5. Бастапқы қатынастарды Бойса-Кодда қалыпты формасына, мысалы, қалыпты форма әдісі көмегімен келтіру.
6. Келесі жағдайларда ER-диаграммаларды қайта қарау:

- қайсыбір қатынастар Бойса-Кодда қалыпты формасына келтірілмейді;
- қайсыбір атрибуттар бастапқы қатынастардың логикалық құрылған орындарында орналаспайды.

ER-диаграммаларды түрлендіргеннен кейін жобалаудың алдыңғы кезеңдерін қайталап орындау (1-кезеңге оралу) жүзеге асады.

Жобалау кезеңдерінің бірі қатынастарды қалыптастыру кезеңі болып табылады. ДҚ сызбасының алғашқы нұсқасын құрайтын бастапқы қатынастарды қалыптастыру процесін қарастырайық.

Жоғарыда қарастырылған мысалдарда Жүргізеді байланысы әрқашан екі болмыстарды байланыстырады және сондықтан *бинарлық* болып табылады. Қарым-қатынасты қалыптастыру ережесіне сәйкес ER-типті диаграмма тек бинарлық байланыстарға тән. Сондықтан, байланыс туралы сөз болғанда, «бинарлық» сөзі бұдан былай жазылмайды.

6.3. Қарым-қатынасты қалыптастыру ережесі

Қарым-қатынасты қалыптастыру ережесі келесілерге негізделген:

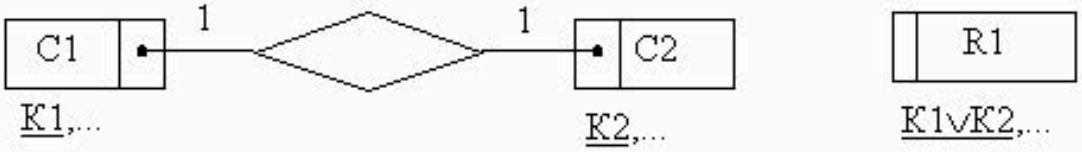
- мәндер арасындағы байланыстар деңгейі (1:1, 1:M, M:1, M:M);
- болмыстар үлгісі жататын санаттағы болмыстар үлгісі (міндетті және міндетті емес).

ER-типті диаграмма негізінде қатынастарды қалыптастырудың алты ережесін қарастырайық.

1:1 байланыстары үшін қатынастарды қалыптастыру

1-ереже: Егер бинарлық байланыстар деңгейі 1:1 және болмыстар үлгісі жататын санат екі болмыстар үшін де міндетті болса, онда бір қатынас қалыптасады. Бұл қатынастардың алғашқы кілті екі болмыстардың кез-келгенінің кілті болуы мүмкін.

6.6-суретте 1-ереже негізінде құрылған ER-типті диаграмма және қатынас келтірілген.



6.6-сурет. 1-ереже үшін диаграмма және қатынас

Суретте келесі белгілеулер қолданылады:

С1, С2 – 1-ші және 2-ші болмыстар;

К1, К2 – сәйкесінше бірінші және екінші болмыстар кілттері;

R1 – бірінші және екінші болмыстар негізінде қалыптасқан 1-қатынас;

К1∨К2,..., қалыптасқан қатынастар кілті К1 немесе К2 болуы мүмкін екенін білдіреді.

Бұл және басқа ережелерді «Оқытушы пәнді жүргізеді» байланыстарының әртүрлі нұсқаларын қарастырып тексереміз. Оқытушы мағынасы ОЖН (оқытушының жеке номері), Аты-жөні (фамилия, аты және әкесінің аты), Жұмыс өтілі (оқытушының жұмыс өтілі) атрибуттарымен сипатталсын. Пәні мағынасы сәйкесінше атрибуттарымен ПК (пәндер коды), Сағаттар (пәнді оқуға бөлінген сағаттар) сипатталсын. Онда екі болмыстар туралы ақпаратты қамтитын қатынастар сызбасы және байланыстар деңгейі 1:1 болатын, ал БҰЖС барлық болмыстар үшін міндетті болатын жағдай үшін қатынас 6.7-суреттегідей болады.

ОҚЫТУШЫ_ПӘН(пн, аты-жөні, өтілі, пкоды, сағаттар)

ОҚЫТУШЫ_ПӘН

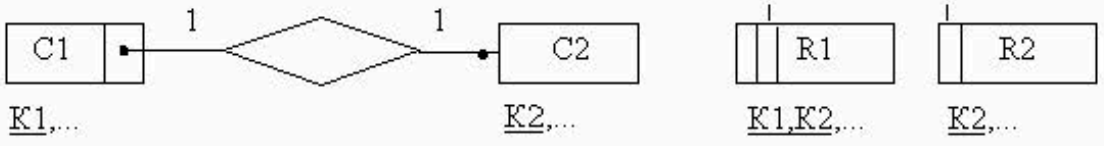
Пән нөмері	Аты-жөні	Өтілі	Пән коды	Сағаттар
П1	Иванов	5	К1	62
П2	Петров	7	К2	74
П3	Сидоров	10	К3	102
П4	Егоров	5	К4	80

6.7-сурет. 1-ереже бойынша алынған сызба және қатынас

Қалыптасқан қатынас оқытушылар, пәндер және олардың арасындағы байланыстар туралы толық ақпаратты қамтиды. Ендеше, оқытушы Иванов тек қана К1 кодты пәнді жүргізеді, ал К1 пәнін тек қана Иванов жүргізеді (1:1 байланысы). Бұл қатынаста, бірде-бір пәннен сабақ бермейтін оқытушылар және ешкім жүргізбейтін пәндер жоқ болғандықтан, бос өрістер (барлық болмыстар үшін БҰЖС міндетті) болмайды. Осылайша, осы жағдайда бір қатынастар жеткілікті. Алғашқы кілт ретінде ОЖН бірінші қатынастар кілті және ПК екінші қатынастар кілті таңдалуы мүмкін.

2-ереже: Егер байланыстар деңгейі 1:1 және болмыстар үлгісі жататын санат біреуі міндетті, ал екіншісі – міндетті емес болса, онда әрбір болмыстарда сәйкес болмыстардың кілті болып табылатын алғашқы кілт бойынша қатынас қалыптасады.

6.8-суретте ER-типті диаграмма және 2-ереже бойынша қалыптасқан қатынас келтірілген.



6.8-сурет. 2-ереже үшін диаграмма және қатынастар

Ережелердің дұрыстығына көз жеткізу үшін келесі мысалды қарастырайық. 6.9-суретте оқытушылар және пәндер туралы ақпаратты қамтитын бастапқы қатынас келтірілген. Ол ұсынатын нұсқада Оқытушы болмыстар класы міндетті болып табылады, ал Пәні болмыстары міндетті емес. Сонымен қатар, барлық кортеждерде бірде-бір оқытушы жүргізбейтін пәндер туралы ақпараттары бар "---" (бос өрістер) бар.

ОҚЫТУШЫ_ПӘН

Пән нөмері	Аты-жөні	Өтілі	Пән коды	Сағаттар
П1	Иванов	5	К1	62
П2	Петров	7	К2	74
П3	Сидоров	10	К3	102
---	---	---	К4	80

6.9-сурет. Бастапқы қатынас

Бұл жағдайларда 2-ережені қолданып, 6.10-суретте көрсетілгендей екі қатынастарға бөлуге болады.

ОҚЫТУШЫ (пн, аты-жөні, өтілі, пк)

ПН	Аты-жөні	өтілі	пкоды
П1	Иванов	5	К1
П2	Петров	7	К2
П3	Сидоров	10	К3

ПӘН (пк, сағаттар)

ПК	Сағаттар
К1	62
К2	74
К3	102
К4	80

6.10-сурет. 2-ереже бойынша алынған қатынастар

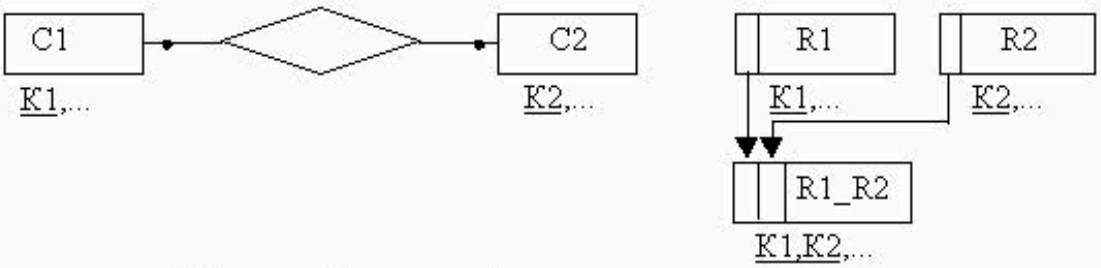
Нәтижесінде қатынастарда деректерді жоғалтпай біз бос өрістен құтылдық. Қатынаска ПК атрибутын – Оқытушы болмыстарына (міндетті БҰЖС) сыртқы кілт ретінде сәйкес келетін Пәні болмыстар кілтін (міндетті емес БҰЖС) қосып, біз қатынастарды байланыстырдық (6.11-сурет).

6.11-сурет. Сыртқы кілт бойынша қатынастар байланысы

Нақтырақ айтқанда, біз қатынастарды байланыстыру үшін жағдай жасадық. Бұл байланыстыру деректер қорымен жұмыс жасау кезінде, мысалы, бір уақытта оқытушы туралы және ол жүргізетін пәндер (сағаттар) туралы деректер алуға мүмкіндік береді.

3-ереже: Егер байланыстар деңгейі 1:1 және екі болмыстар үшін де болмыстар үлгісі жататын санат міндетті емес болып табылса, онда үш қатынастар қолдану қажет. Екі қатынастар кілттері осы қатынастарда алғашқы кілттер болатын байланыстырылатын болмыстарға сәйкес келеді. Үшінші қатынас екеуінің арасындағы жалғастырушы болып табылады, сондықтан оның кілті байланыстырылатын қатынастардың кілттік атрибутын біріктіреді.

6.12-суретте 3-ереже бойынша қалыптасқан ER-типті диаграмма және қатынас келтірілген.



6.12-сурет. 3-ереже үшін диаграмма және қатынастар

6.8-суретте үш қатынастарды қолдану қажеттілігін көрсететін, екі байланыстырылатын болмыстар үшін міндетті емес түрдегі БҰЖС кезіндегі қатынастар мысалы келтірілген.

а) бір қатынас

ОҚЫТУШЫ_ПӘН

ПІІ	Аты-жөні	Отілі	Пән коды	Сағаттар
П1	Иванов	5	К1	62
П2	Петров	7	---	---
П3	Сидоров	10	К3	102
---	--	---	К4	80

б)Екі қатынас

ОҚЫТУШЫ_ПӘН

ОҚЫТУШЫ (пн, аты-жөні, өтілі, пк)

ПН	Аты-жөні	өтілі	пкоды
П1	Иванов	5	К1
П2	Петров	7	
П3	Сидоров	10	К3

ПӘН (пк, сағаттар)

ПК	Сағаттар
К1	62
К2	74
К3	

в) үш қатынас

ОҚЫТУШЫ (пн, аты-жөні, отілі, пк)

ПН	Аты-жөні	өтілі	шкоды
П1	Иванов	5	К1
П2	Петров	7	
П3	Сидоров	10	К3

ЖҮРГІЗЕДІ

ПН	ПК
П1	К1
П3	К2

ПӘН (пк, сағаттар)

ПК	Сағаттар
К1	62
К2	74
К3	

6.13-сурет. 3-ереже үшін қатынастар нұсқалары

Қарастырылған жағдайда бір қатынасты қолдану бұл қатынаста бос өрістердің болуына апарады (6.13а-сурет). Екі қатынастарды қолданған кезде (6.13б-сурет) бізге қатынасқа әрбір оқытушы қай пәнді жүргізеді және керісінше мәліметтерді жоғалтпау үшін кілттер кірістіруге (қосу) тура келді. Сонымен қатар, тағы да бос өрістер пайда болды.

3-ереже бойынша үш қатынастарды қолдану арқылы шешім табылды (6.13в-сурет). Объектілік қатынастар (болмыстар атрибуттарымен) барлық оқытушылар және пәндер туралы толық ақпаратты қамтиды. Байланыстырушы «Жүргізеді» қатынасы пәндерді оқытатын оқытушылар туралы және оқытушылар оқытып жүрген пәндер туралы деректерді қамтиды. Сонымен қатар, онда 1:1 байланыстар орнатылған. Бұл қатынас

осы жағдайда екі болмыстар үшін де тек қана кілттік атрибуттарды (бірақ осы байланысты сипаттайтын басқа атрибуты да болуы мүмкін) қамтиды. Мысалы, оқытушы пәнді жүргізетін семестр нөмері.

Ендеше, ER-диаграмм негізінде қатынастар қалыптастыруға мүмкіндік беретін тұжырымдалған үш ережелер 1:1 типті байланыстар деңгейі бар нұсқаларға арналған. Мәндер арасындағы байланыстар деңгейі 1:М нұсқалар үшін ұқсас екі ереже қалыптастырайық.

1:М байланыстар үшін қатынастарды қалыптастыру

Егер екі болмыстар С1 және С2 1:М байланысты болса, онда мағынасы бойынша С1 бірбайланысты (1-байланысты), ал С2 – көпбайланысты (М-байланысты) деп аталады. Қатынастарды қалыптастыру кезінде анықтаушы фактор, байланыстардың осы түрімен байланысқан болмыстар үлгісі жататын санат М-байланысты болмыстар болып табылады. Ендеше, егер болмыстар үлгісі жататын санат М-байланысты болмыстар міндетті болса, онда ережелерді қолдану нәтижесінде екі қатынас аламыз, егер міндетті емес болса – үш қатынастар аламыз. Болмыстар үлгісі жататын санат бірбайланысты болмыстарда нәтижеге әсер етпейді.

Бұған көз жеткізу үшін, 6.4-суретте келтірілген диаграммаға сәйкес келетін Оқытушы_Пәні (6.14-сурет) қатынасын қарастырайық, яғни байланысы 1:М типті, болмыстар үлгісі жататын санат М-байланысты болмыстар үшін міндетті, 1-байланысты болмыстар үшін міндетті емес жағдайы.

ОҚЫТУШЫ_ПӘН

ПН	АТЫ-ЖӨНІ	ӨТІЛІ	ПКОДЫ	САҒАТТАР
П1	Иванов	5	К1	62
П1	Иванов	5	К2	74
П2	Петров	7	К4	80
П3	Сидоров	10	К5	96
П3	Сидоров	10	К6	120
П4	Егоров	5	К3	102
П4	Егоров	5	К7	89
П5	Козлов	8	---	---

6.14-сурет. Бастапқы қатынас

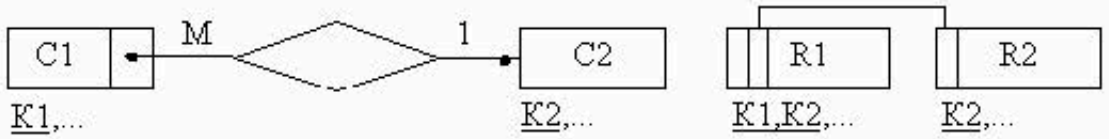
Оқытушы_Пәні (6.14-сурет) қатынасымен келесі мәселелер байланысты:

- бар кортеждерде бос өрістер бар (оқытушы оқытпайтын пәндер),
- кортеждерде бірнеше пәндерді жүргізетін оқытушылар туралы мәліметтермен деректердің артық қосарланушылығы (оқытушының жұмыс өтілі қайталанады) бар.

Егер болмыстар үлгісі жататын санат 1-байланысты болмыстар міндетті (ешбір пәнді жүргізбейтін оқытушылар жоқ) болса, онда бос өрістер жойылатын еді, бірақ оқытушы атрибуттарында қайталанатын деректер сақталып қалады. Аталған мәселелерді жою үшін қатынастарды келесі ереже бойынша қалыптастыру керек.

4-ереже: Егер мәндер арасындағы байланыстар деңгейі 1:М (немесе М:1) және болмыстар үлгісі жататын санат М-байланысты болмыстар міндетті болса, онда екі қатынастар (әрбір болмысқа біреуден) қалыптастыру жеткілікті. Сонымен қатар, осы қатынастардың алғашқы кілті олардың болмыстарының кілттері болады. Сондай-ақ, қатынаста 1-байланысты болмыстарға кілт (сыртқы кілт) М-байланысты болмыстарға сәйкес келетін атрибут ретінде қосылады.

6.15-суретте 4-ереже бойынша қалыптастырылған ER-типті диаграмма және қатынастар келтірілген.



6.15-сурет. 4-ереже үшін диаграмма және қатынастар

4-ережеге сәйкес 6.14-суреттегі қатынасты екі қатынастарға (6.16-сурет) түрлендіреміз.

ОҚЫТУШЫ

ПН	АТЫ-ЖӨНІ	ӨТІЛІ
П1	Иванов	5
П2	Петров	7
П3	Сидоров	10
П4	Егоров	5
П5	Козлов	8

ПӘН

ПК	САҒАТТАР	ПН
К1	62	П1
К2	74	П1
К3	102	П4
К4	80	П2
К5	96	П3
К6	120	П3
К7	89	П4

6.16-сурет. 4-ереже бойынша алынған қатынастар

6.16-суреттен көрсетілгендей, бос өрістер және ақпараттардың қосарланушылығы жойылды. Оқытушылардың қайсысы қай пәнді жүргізетіні туралы мәліметтер ОЖН кілтін енгізу нәтижесінде жоғалған жоқ.

Келесі ережелерді енгізу және бекіту үшін мысалдар қарастырайық.

Мысалы. Мәндер арасындағы байланыс 1:М, ал болмыстар үлгісі жататын санат М-байланысты болмыстар міндетті емес болсын.

Болмыстар үлгісі жататын санат 1-байланысты болмыстар сонымен қатар, міндетті емес болсын. Бірақ шарт емес, өйткені анықтаушы болмыстар үлгісі жататын санат М-байланысты болмыстар болып табылады. Бұл жағдайда бір қатынастарды қолдану қандай нәтиже беретінін көрейік (6.17-сурет).

ОҚЫТУШЫ_ПӘН

ПН	АТЫ-ЖӨНІ	ӨТІЛІ	ПКОДЫ	САҒАТТАР
П1	Иванов	5	К1	62
П1	Иванов	5	К2	74
П2	Петров	7	К4	80
--	---	---	К5	96
П3	Сидоров	10	К6	120
П4	Егоров	5	К3	102
П4	Егоров	5	К7	89
П5	Козлов	8	---	---

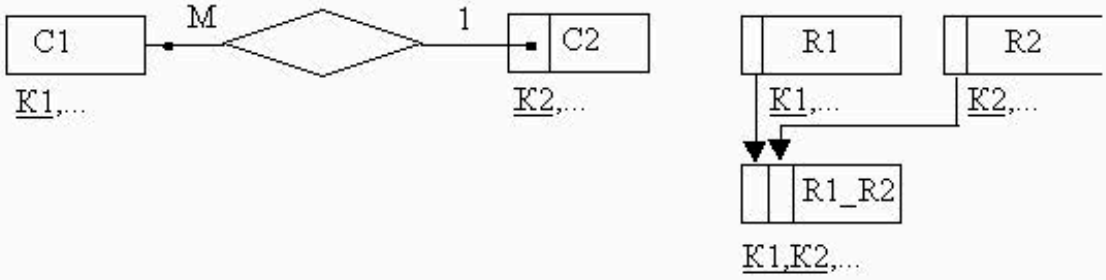
6.17-сурет. Бастапқы қатынас

Келтірілген қатынастармен келесі мәселелер байланысты:

1. Кортеждерде келесі деректерден тұратын бос өрістер бар:
   а) пәндерді жүргізбейтін оқытушылар туралы деректер;
   б) оқытушылар жүргізбейтін пәндер туралы деректер.
2. Бірден көп пәндерді жүргізетін оқытушылар туралы деректердің артық қосарланушылығы бар.

Болмыстар үлгісі жататын санат 1-байланысты болмыстар жоғалуы 1 а) жағдайында барлық мәселелерді жою үшін келесі ереже бойынша үш қатынастарға көшу керек.

5-ереже. Егер байланыстар деңгейі 1:М (М:1) және болмыстар үлгісі жататын санат М-байланысты болмыстар міндетті емес болып табылса, онда үш қатынастар қалыптастыру қажет (6.18-сурет). Екі қатынастар байланыстырылатын болмыстарға сәйкес келеді, олардың кілттері осы қатынастардың алғашқы кілттері болады. Үшінші қатынас алғашқы екі қатынас арасындағы байланыстырушы болып табылады (оның кілті байланыстырылатын қатынастар кілттік атрибуттарын құрайды).



6.18-сурет. 5-ереже үшін диаграмма және қатынас

Қарастырылып отырған қатынасқа 5-ереже қолдану нәтижесінде деректер (6.12-сурет) үш қатынас бойынша жіктеледі (6.19-сурет).

ОҚЫТУШЫ

ПН	Аты-жөні	Өтілі
П1	Иванов	5
П2	Петров	7
П3	Сидоров	10
П4	Егоров	5
П5	Козлов	8

ЖҮРГІЗЕДІ

ПН	ПК
П1	К1
П1	К2
П2	К4
П3	К6
П4	К3
П4	К7

ПӘН

ПК	Сағаттар
К1	62
К2	74
К3	102
К4	80
К5	96
К6	120
К7	89

6.19-сурет. 5-ереже бойынша алынған қатынастар

Осылайша, көрсетілген мәселелердің шешімін таптық. «Жүргізеді» қатынасындағы кілт құрама болып табылады және екі қатынас арасындағы байланыстырушы кілттік атрибутты іске қосады. Практикалық тұрғыдан, жағдайларда байланыстырушы қатынас және басқа сипаттайтын байланыс атрибуты тұруы мүмкін.

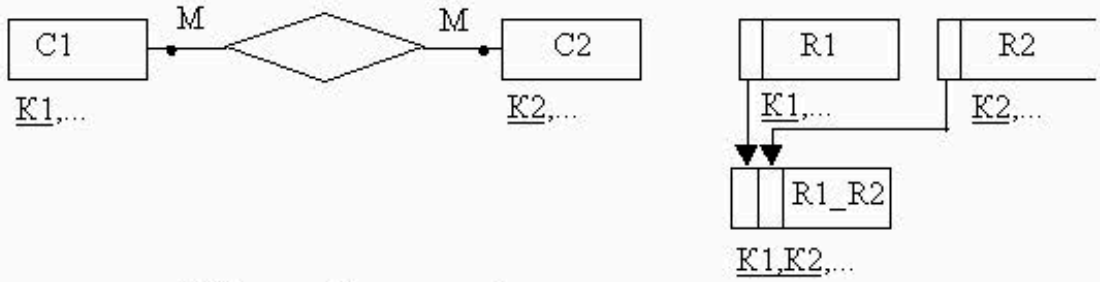
4-ереже мен 5-ереже арасында таңдау кезінде анықтаушы фактор болмыстар үлгісі жататын санат М-байланысты болмыстар болып табылатынын айта кету керек.

М:М түріндегі байланыстар үшін қатынастарды қалыптастыру

Екі болмыстар арасында М:М түріндегі байланыстар кезінде болмыстардан тәуелсіз үш қатынастар үлгісі жататын санат кез-келген болмыстар болсын. Бұл жағдайда бір немесе екі қатынастарды қолдану бос өрістерді немесе деректердің артық қосарланушылығын жоймайды.

6-ереже. Егер байланыстар деңгейі М:М түрінде болса, онда болмыстардың жататын санатынан тәуелсіз үш қатынастар қалыптасады. Екі қатынастарға байланыстырылатын болмыстар сәйкес келеді және олардың кілттері осы қатынастардың алғашқы кілті болады. Үшінші қатынас алғашқы екеуінің арасындағы байланыстырушы болып табылады, ал оның кілті байланыстырылатын қатынастардың кілттік атрибуттарын біріктіреді.

6.19-суретте 6-ереже бойынша құрылған ER-типті диаграмма және қатынастар келтірілген. Біз қарастырған нұсқада болмыстар үлгісі жататын санат Н-Н, бірақ, 6-ережеге сәйкес ол ерікті болуы мүмкін.



6.20-сурет. 6-ереже үшін диаграмма және қатынастар

6-ережені 6.5-суретте келтірілген мысалға пайдалануға болады. Онда байланыстар деңгейі М:М болады, болмыстар үлгісі жататын санат Оқытушы болмыстары үшін міндетті, ал Пәні болмыстары үшін міндетті емес. Бұл мысалға сәйкес бастапқы қатынас 6.21-суретте көрсетілген.

ОҚЫТУШЫ_ПӘН

ПН	АТЫ-ЖӨНІ	ӨТІЛІ	ПКОДЫ	САҒАТТАР
П1	Иванов	5	К1	62
П1	Иванов	5	К2	74
П2	Петров	7	К4	80
--	---	---	К3	96
П3	Сидоров	10	К6	120
П4	Егоров	5	К3	102
П4	Егоров	5	К7	89
П5	Козлов	8	К5	96

6.21-сурет. Бастапқы қатынас

6-ережені қолдану нәтижесінде үш қатынастар алынады (6.22-сурет).

ОҚЫТУШЫ

ПН	Аты-жөні	Өтілі
П1	Иванов	5
П2	Петров	7
П3	Сидоров	10
П4	Егоров	5
П5	Козлов	8

ЖҮРГІЗЕДІ

ПН	ПК
П1	К1
П1	К2
П2	К4
П3	К6
П4	К3
П4	К7

ПӘН

ПК	Сағаттар
К1	62
К2	74
К3	102
К4	80
К5	96
К6	120
К7	89

6.22-сурет. 6-ереже бойынша алынған қатынастар

Осындай нәтижелер болмыстар үлгісі жататын санатпен ажыратылатын болмыстардың басқа үш нұсқасы үшін де алынады. Қалыптастырылған ережелерді мән-байланыс әдісімен ДҚ жобалау мысалдарында қарастырайық.

6.4. Оқу бөлімі деректер қорын жобалау мысалы

Сипатталған ережелерді қалыпты форма әдісін меңгеру кезінде қарастырылған мысалға (0-бөлім) пайдалануға болады. Ескере кетейік, ондағы оқу бөлімдері туралы ДҚ келесі мәліметтерді қамтыды:

Аты-жөні – оқытушы фамилиясы және аты-жөнінің бірінші әріптері. Оқытушылардың фамилиясы және аты-жөнінің бірінші әріптері қайталанбайтын болсын.

Қызметі – оқытушының атқаратын қызметі.

Жалақысы – оқытушы жалақысы.

Жұмыс өтілі – оқытушылық жұмыс өтілі.

Ү_Жұмыс өтілі – жұмыс өтіліне үстеме.

Кафедра – оқытушы жұмыс жасайтын кафедра номері.

Пән – оқытушының оқытатын пәнінің атауы.

Тобы – оқытушы сабақ өтетін топтар номері.

Сабақ түрі – оқытушының оқу тобында жүргізетін сабақ түрі.

Оқытушы бастапқы қатынасы 5.4-суретте келтірілген.

Жобалау кезінде 6.2-бөлімде сипатталған кезеңдерді қамтимыз.

Жобалаудың бірінші кезеңі – болмыстарды және олардың арасындағы байланысты жіктеу.

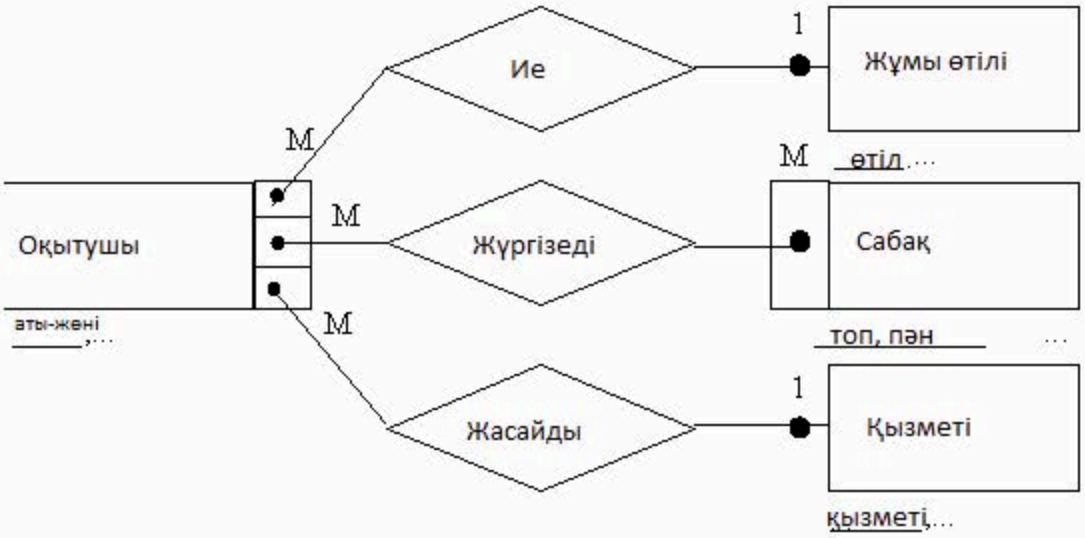
Келесі болмыстарды атап өтейік:

- Оқытушы (Кілт – Аты-жөні),
- Занятие (Кілт – Тобы, Пән),
- Жұмыс өтілі (Кілт – Жұмыс өтілі),
- Қызметі (Кілт – Қызметі).

Мәндер арасындағы байланыстар:

- Оқытушы **ИЕ** Жұмыс өтілі,
- Оқытушы **ЖҮРГІЗЕДІ** Сабақ,
- Оқытушы **ЖАСАЙДЫ** Қызметі.

Екінші жобалау кезеңі – барлық болмыстар және олардың арасындағы байланысты ескере отырып ER-типті диаграммалар тұрғызу. Қарастырылған мысал үшін ER-типті диаграмма 6.4.1-суретте келтірілген.



6.4.1-сурет. ER-типті диаграмма

Байланыс М:1 типті, өйткені бірнеше оқытушылардың бірдей жұмыс өтілі болуы мүмкін. Оқытушы міндетті болмыстар үлгісі жататын санатқа ие, себебі әрбір оқытушының өз жұмыс өтілі бар. Жұмыс өтілі міндетті емес болмыстар үлгісі жататын санатқа ие, өйткені жұмыс өтілінің ондай мәні бірнеше оқытушыларда болуы мүмкін.

Жүргізеді байланысы М:М типті, өйткені оқытушы бірнеше пәнді жүргізуі мүмкін, ал әрбір пәнді бірнеше оқытушы жүргізуі мүмкін. Оқытушының оқу тобында пәндердің бірі бойынша жүргізетін сабағы дәріс немесе практикалық сабақ болуы мүмкін. Берілген байланыстарда болмыстардың екеуі де БҰЖС міндетті, бірде-бір пәнді оқытпайтын оқытушы және бірде-бір оқытушы жүргізбейтін пән жоқ деп есептейміз.

ЖАСАЙДЫ байланысы М:1 типті, өйткені әрбір оқытушы қандай да бір қызмет иесі және бірнеше оқытушылардың қызметі бірдей болуы мүмкін. Оқытушы міндетті болмыстар үлгісі жататын санатқа ие, өйткені әрбір оқытушының қызметі бар. Қызметі міндетті емес БҰЖС. Себебі: кафедрада, мысалы, профессор қызметі болмауы, яғни осы қызметті атқаратын оқытушы болмауы мүмкін.

Үшінші жобалау кезеңі – ER-типті диаграммаларды пайдаланып, әрбір қатынастар үшін алғашқы кілтті көрсете отырып, бастапқы қатынастар жиынтығын қалыптастыру.

ER-типті диаграммалар негізінде талдау (6.4.1-сурет) көмегімен, жоғарыда қалыптасқан алты ереже арқылы қатынастары келесі сызбалармен берілген бастапқы қатынастар жинағын аламыз.

ИЕ байланысы 4-ереже шарттарын қанағаттандырады, соларға сәйкес екі қатынастар аламыз:

1. Оқытушы (Аты-жөні, Жұмыс өтілі, ...) – Жұмыс өтілі кілттік атрибуты қосылды.
2. Жұмыс өтілі (Жұмыс өтілі, ...).

Жүргізеді байланысы 6-ереже шарттарын қанағаттандырады, соларға сәйкес үш қатынастар аламыз:

1. Оқытушы (Аты-жөні, Жұмыс өтілі, ...).
2. Сабақ (Тобы, Пәні, ...).
3. Жүргізеді (Аты-жөні, Тобы, Пәні, ...).

Жасайды байланысы 4-ереже шарттарын қанағаттандырады, соларға сәйкес екі қатынастар аламыз:

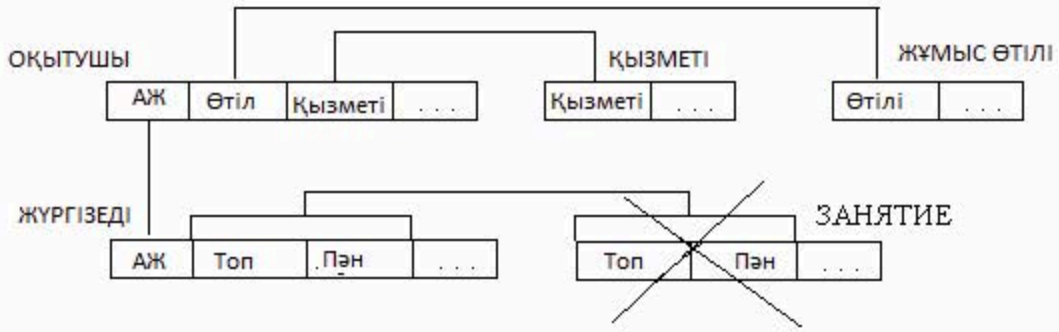
1. Оқытушы (Аты-жөні, Жұмыс өтілі, Қызметі, ...) Қызметі кілттік атрибуты қосылды.
2. Қызметі (Қызметі, ...).

Үшінші жобалау кезеңі – кілттік емес атрибуттар кірістіру немесе қосу және олардың бастапқы қатынастардан Бойса-Кодда қалыпты формалар талаптарын қанағаттандыратын шарттармен тағайындалуы.

Кілттік емес атрибуттарды кірістіргеннен кейін қатынастар сызбасы келесі түрге келеді:

Оқытушы (Аты-жөні, Жұмыс өтілі, Қызметі, Кафедра),
Жұмыс өтілі (Жұмыс өтілі, Y_жұмыс өтілі),
Сабақ (Тобы, Пәні),
Жүргізеді (Аты-жөні, Тобы, Пәні, Сабақ түрі),
Қызметі (Қызметі, Жалақысы).

Қатынастарды анықтағаннан кейін олардың Бойса-Кодда қалыпты формалар талаптарына сәйкестігін тексеру керек. Біздің жағдайда біз қалыпты формалар әдісімен жобалау кезіндегі қатынастарды алдық. Деректер қорының алынған сызбасы 6.4.2-суретте келтірілген.



6.4.2-сурет. Деректер қоры сызбасы

Ескертулер.

- Біздің мысалда Сабақ қатынасы, кілттік атрибуттардан (Тобы, Пәні) басқа атрибуттарға ие емес. Сондықтан, ол Жүргізеді қатынастарындағы ақпараттардан басқа қосымша ақпараттарды қамтымайды. Шындығында, бұл қатынас Тобы, Пәні атрибуттарын іске қосады. Сондықтан, Сабақ қатынасын қалыптасатын ДҚ сызбасынан алып тастау керек (6.2-суретте ол сызылып тасталған). Егер басқа атрибуттар болса,

мысалы Семестр атрибуты, онда Сабақ қатынасы Сабақ (Тобы, Пәні, Семестр), ДҚ-на кіретін еді.

- Жобалаудың соңғы кезеңінде қатынастардағы қосарланушылық ақпараттар талданады. Сонымен қатар, әрбір қатынастардың бірнеше кортеждері қарастырылуы мүмкін. Артықтық болған кезде сәйкес бөлімдерді қайтадан жобалайды (ER-диаграмма) немесе қалыпты форма әдісін қолданып қатынастарды декомпозициялайды. Түрлендірулердің соңғы нәтижесі Бойса-Кодда қалыпты формасындағы қатынастар жиынтығы болып саналады.
- ДҚ жобалаудағы қарастырылған ережелер практикалық тұрғыдан көптеген жағдайларда қолданылады. Басқа нақты үлгілерді тұрғызу қосымша жобалауларды қажет етуі мүмкін. Жеке жағдайда, *бинарлыққа* қарағанда неғұрлым жоғары ретті, мысалы, үш болмыстарды байланыстыратын *үштік* байланысты пайдалану қажеттілігі пайда болуы мүмкін.

Бақылау сұрақтары және тапсырмалар

1. Мән-байланыс әдісінің негізгі ұғымдарын атап шығыңыз.
2. Болмыстар кілті түсінігін сипаттаңыз.
3. ER-үлгі диаграммасы және ER-типті диаграммалар нені көрсетеді?
4. Мәндер арасындағы байланыстар деңгейі нені анықтайды?
5. Болмыстар үлгісі жататын санат қандай болуы мүмкін?
6. Мәндер арасындағы байланыстар деңгейі 1:1 және міндетті болмыстар үлгісі санатты екі болмыстардың ER-үлгі диаграммасына мысал келтіріңіз.
7. ER-типті диаграммаларда байланыстар деңгейі қалай белгіленеді?
8. 1:M типті байланыстар H-O нұсқасы үшін ER-үлгі диаграммасына мысал келтіріңіз.
9. Деректер қорын жобалау кезеңдерін атаңыз.
10. 1:1 байланыстары үшін қатынастарды қалыптастыру қалай жүзеге асады?
11. Егер байланыстар деңгейі 1:1 және болмыстар үлгісі жататын санат екі болмыстар үшін де міндетті емес болып табылса, онда қатынастарды қалыптастырудың 3-ережесін тұжырымдаңыз.
12. Мәндер арасындағы байланыстар 1:M (M:1) деңгейі үшін қатынастарды қалыптастырудың ережесін тұжырымдаңыз.
13. Қарым-қатынасты қалыптастыру ережесі үшін M:M түріндегі байланыстарды түсіндіріңіз.

7. ЖОБАЛАУДЫ АВТОМАТТАНДЫРУ ҚҰРАЛДАРЫ

Бөлімде CASE-құралдар құру және ақпараттық жүйелерді сүйемелдеудің жалпы сипаттамасы беріледі. Бағдарламалық жүйелердің өміршеңдік кезең үлгілері қарастырылады, құрылымдық және объектілі-негізделген жобалау үлгілері суреттеледі. CASE-құралдарды жіктеу беріледі және неғұрлым танымал жүйелер сипаттамасы келтіріледі.

7.1. Негізгі ұғымдар

Жобалауды автоматтандыру және ақпараттық жүйелерді өңдеу үшін 70-80 жылдары құрылатын жүйелер және қабылданған техникалық шешімдерді сипаттау әдістерін қолдануды білдіретін *құрылымдық әдістеме* кеңінен қолданылды. Сонымен қатар, графикалық тұрғыдан, сызбалар және диаграммалар көмегімен ақпараттық жүйелердің әртүрлі нұсқаларын сипаттау құралдары қолданылды. Ақпараттық жүйелерді қолмен құру кезінде графикалық тұрғыдан ондай үлгілерді құру және қолдану қиынырақ.

Аталған жағдайлар ақпараттық жүйелерді құру және сүйемелдеудің CASE-технологиясын жүзеге асыратын CASE-құралдар деп аталатын бағдарламалық-техникалық құралдарды қолданудың бір себебі болды. Құрылымдық әдістемеден басқа, заманауи CASE-құралдар қатарында объектілі-бағытталған жобалау әдістемесі қолданылады.

CASE (Computer Aided Software Engineering) термині сөзбе-сөз компьютердің бағдарламалық қамтамасыз ету көмегімен құру деп аударылады. Қазіргі уақытта бұл термин ақпараттық жүйелерді өңдеуді автоматтандыруды білдіретін неғұрлым кең мағынаға ие.

CASE-құралдар ақпараттық жүйелерді құру және/немесе сүйемелдеудің төмендегі процестерін сүйемелдейтін бағдарламалық құралдарды қамтиды: талаптарды талдау және қалыптастыру, деректер қорын және қосымшаны жобалау, кодты іздестіру, тестілеу, сапаны қамтамасыз ету, жобаны басқару.

CASE-жүйелерді алдын-ала нақты функционалды тағайындалуы бар және біртұтас бағдарламалық өнім шеңберінде орындалатын CASE-құралдар жинағы ретінде анықтауға болады.

Әдетте ***CASE-технология*** пәндік облысты көрнекі түрде құратын, өңдеудің барлық кезеңдерінде оның үлгісін талдайтын және ақпараттық жүйелерді сүйемелдейтін және пайдаланушылар үшін қосымша құратын құрал-саймандық құралдары бар ақпараттық жүйелерді жобалау әдістемесі ретінде анықталады.

CASE-индустрия әртүрлі бағыттағы жүздеген фирмалар мен компанияларды біріктіреді. Практикалық тұрғыдан барлық маңызды шетелдік бағдарламалық жобалар CASE-құралдарды қолданумен жүзеге асады, ал қолданыстағы бағдарламалық дестелердің жалпы саны 500-ден астам.

CASE-жүйелер және құралдардың негізгі мақсаты кодтау және өңдеу (тестілеу, құжаттандыру және басқа) кезеңдерін жобалауды бағдарламалық қамтамасыз ету, сонымен қатар бағдарламалық жүйелер құру процесін автоматтандыру немесе *инжиниринг* (ағылшынша engineering – құру).

Соңғы уақытта барлық бағдарламаларды құру қайсыбір жүйелердің бастапқы нұсқасынан басталады. Бұл нұсқа ретінде арнайы осы үшін құрылғандай *түпнұсқа* немесе ескірген және жүйелердің жаңа талаптарын қанағаттандармайтын нұсқа қолданылуы мүмкін. Соңғы жағдайда бағдарламалық жүйелер туралы білімдерді қалпына келтіріп, оларды бұдан әрі пайдалану мақсатында қайтадан өңдеу – *реинжиниринг* қолданылады.

Қайтадан құру бағдарламалық жүйелердің бағдарламалық кодын зерттеу жолымен бастапқы үлгілерді тұрғызуға апарып соқтырады. Үлгіні алдымен жетілдіреміз, содан соң оларды қайтадан өңдеуге көшеміз. Бұл жобалау кезінде жиі қолданылады. Осы принциптердің неғұрлым танымалдарының бірі «кайтадан жобалау» – Round Trip Engineering (RTE) принципі болып табылады.

Заманауи CASE-жүйелер тек қана құрумен шектелмейді, олар көбінесе қайтадан өңдеуді қамтамасыз етеді. Бұл қосымшаны құруды недәуір жеңілдетеді және олардың сапасын арттырады.

Құрылымдық және объектілі-бағытталған жүйелер позициясына өзгерістер енгізу динамикасы келешегі бар CASE-жүйелер объектілі-бағытталған жүйелер болады деп болжауға мүмкіндік береді. Кемшіліксіз келешегі бар CASE-жүйелерге қойылатын талаптарды қарастырайық.

Толық функционалдық объектілі-бағытталған жүйелер талдау және модельдеу, жобалау, өңдеу (жүзеге асуы) тапсырмаларын шешуі, сонымен қатар, негізгі үш тапсырманы жүзеге асыруды қамтамасыз ететін тиімді инфрақұрылымы болуы керек.

Талдау және модельдеу тапсырмаларын шешу үшін келесі мүмкіндіктерді қамтамасыз етуі тиіс бағдарлама құрушының біріктірілген ортасы болғаны жөн:

- Жүйелерде динамикалық модельдеу;
- Диаграммалардың барлық жиынтығына динамикалық және келісілген түзетулер енгізу;
- Диаграммаларға және құжаттандыруға түсіндірме жазбалар қосу;
- Құжаттарды автоматты түрде дайындау;
- Әртүрлі көріністер және жүйелердің жабық қолданылмайтын топтарын құру;
- Әртүрлі жазба шарттарын сүйемелдеу (бұл талап байланыстарда модельдеудің ортақ тіліне көшу кезінде бәсеңдейді).

Жобалау процесін жүзеге асыру келесі мүмкіндіктердің бар болуын жобалайды:

- қолданбалы есептің (әдетте объектілі-бағытталған бизнес-үлгілер) негізгі үлгілерін және оның негізгі ережелерін (бизнес-ереже) анықтау;
- элементтерді (объектілер және ережелер) жобалаудың сақтау, іздеу және таңдау құралдарымен қамтылған кітапханалар көмегімен жобалау процесін сүйемелдеу;
- пайдаланушы интерфейсін құру және кеңінен таралған бағдарламалық интерфейстерді (OLE, OpenDoc, HTML/Java кітапханаларға қатынас құру және т.с.с. стандарттарды) сүйемелдеу;
- әртүрлі үлестірімді клиент-сервер қосымшасын құру.

CASE-жүйелер құрамындағы өңдеу құралдары қосымшаны өңдеудің алдыңғы кезеңдерінің нәтижесі бойынша қосымшаны тұрғызуды қамтамасыз етуі керек. Өңдеу құралдарына қойылатын ең жоғары талаптардың бірі орындалуға дайын бағдарламаларды іздестіру деп есептеуге болады.

Жақын болашақтың CASE-жүйелері кластар, атрибуттар және әдістер үшін қаңқалық дайын нұсқа құруға, C++ немесе Smalltalk типті объектілі-бағытталған бағдарламалау тілінде немесе клиент-серверлік өнімдер (PowerBuilder, Forte, VisualAge, VisualWorks және т.б.) бағдарламалары тілінде дайын қосымшаларды пайдалануға мүмкіндік беруі керек. Сүйемелдеуші тілдерге Java тілі де қосылады.

Болашақтағы CASE-жүйелердің инфрақұрылымы ядросы кодты іздеуге жауап беретін және бағдарламалар үлгілері мен кодтары арасында сәйкестікті қамтамасыз ететін репозиторий болуы керек. Репозиторий негізін объектілі-бағытталған ДҚ құрайды, бірақ бұрын осы үшін реляциялық ДҚ қолданылды. Басқа маңызды инфрақұрылым функциялары нұсқаларды бақылау және топтық құру кезінде жүйе бөліктерін басқару функциялары болады.

7.2. Үлгілердің өміршеңдік кезеңі

Үлгілердің өміршеңдік кезеңі (ӨК) ақпараттық жүйелерді бағдарламалық қамтамасыз етудің (АЖ БҚЕ) автоматтандырылған жобалау кезінде маңызды роль атқарады. Бұл әрбір CASE-жүйелердің әрқайсысы нақты бір АЖ БҚЕ ӨК үлгісіне негізделгендігімен (сүйемелдейтіндігімен) түсіндіріледі.

АЖ БҚЕ өміршеңдік кезеңі БҚЕ құру туралы шешім қабылдаған сәттен басталатын және ол толық жүзеге асқан кезде аяқталатын үзіліссіз процесс болып саналады.

БҚЕ ӨК негізгі нормативті құжаты халықаралық ISO/IEC 12207 (International Organization of Standardization – стандарттау бойынша халықаралық ұйымдастыру /International Electrotechnical Commission – электротехника бойынша халықаралық комиссия) стандарты болып табылады. Онда БҚЕ құрған кезде орындалуы керек процестер, әрекеттер және міндеттерді құрайтын ӨК құрылымы анықталады.

БҚЕ ӨК үлгісі ретінде ӨК кезінде процестер, әрекеттер және міндеттердің орындалу тізбегі және өзара байланыстарын анықтайтын құрылым қарастырылады. Неғұрлым кең таралған БҚЕ ӨК үлгілері: каскадты, аралық бақылаумен және спираль тәріздес.

Каскадты және аралық бақылаумен үлгілері БҚЕ өміршеңдік кезеңінің келесі кезеңдерін қамтиды: талдау, жобалау, жүзеге асыру, енгізу және сүйемелдеу.

Каскадты үлгі өміршеңдік кезеңнің аталған кезеңдерін қатаң тізбектелген түрде жүзеге асыруды жобалайды. Мұндай үлгілердің *ерекшеліктері*: әрбір кезеңде құжаттауды қалыптастыру және жұмысты аяқтау уақыты мен сәйкес шығынды жоспарлау мүмкіндігі. Үлгілердің *кемшілігі* БҚЕ құрудың нақты процесіне сәйкессіздігі болып табылады, әдетте ол қатаң сызбамен шектелмейді және қабылданған шешімді нақтылау немесе қайта қарау үшін алдыңғы кезеңдерге қайтуды талап етеді.

Аралық бақылау үлгісі өміршеңдік кезеңді БҚЕ құру және қолданудың нақты процесіне жақындатады. Каскадты үлгілерден айырмашылықтары: ол түзетулер енгізу үшін өміршеңдік кезеңнің әрбір кезеңінен кез-келген алдыңғы кезеңге қайтуға рұқсат береді. Сонымен қатар, БҚЕ жоғары сенімділігін қамтамасыз етеді, бірақ өңдеу кезеңінің ұзақтығы артады.

Өміршеңдік кезеңнің *спираль тәріздес үлгісі* (7.1-сурет) алдыңғы үлгілердің кемшіліктерін жоюға мүмкіндік береді. Онда алдыңғы кезеңдер: талдау және жобалау кезеңдеріне баса назар аударылады. Техникалық шешімдердің жүзеге асуы түпнұсқалар құру көмегімен тексеріледі.

7.1-сурет. Өміршеңдік кезеңнің спираль тәріздес үлгісі

Өміршеңдік кезеңнің спираль тәріздес үлгісінің сызбасында қандай да бір кезеңде жұмыстың толық аяқталмауы келесі кезеңге өтуге мүмкіндік береді. Аяқталмаған жұмыс спиральдың келесі орамында орындалуы мүмкін.

7.3. Құрылымдық жобалау үлгілері

Ақпараттық жүйелерді талдау және жобалаудың құрылымдық тәсілдемесі оның иерархиялық құрылым түрінде бейнеленуі және бөлшектерге жіктелуін жалпылама қарастырудан тұрады. Иерархияның жоғары деңгейінде әдетте жүйелерді функционалдық сипаттау ұсынылады.

Құрылымдық талдау және жобалау кезінде көрнекілікті арттыру үшін ақпараттық жүйелер және деректердің арасындағы қатынастарды графикалық бейнелену функциялары қолданылады.

Графикалық көрініс үлгілері мен диаграммаларының неғұрлым кеңінен таралғандары:

- мән-байланыс диаграммасы немесе ER-диаграммалар – Entity-Relationship Diagrams (ERD) деректер қоры сызбасының көрнекі түрде көрінісі үшін қолданылады(6-бөлім);
- деректер ағымы диаграммалары – Data Flow Diagrams (DFD) жүйелер үлгілерін иерархиялық сипаттау үшін қолданылады;
- құрылымдық талдау және жобалау әдісі – Structured Analysis and Design Technique (SADT), қызметкер үшін объектінің функционалды үлгілерін тұрғызу;
- енгізу-өңдеу-шығару құрылымын сипаттау сызбасы – Hierarchy plus Input-Processing-Output (HIPO) бағдарламамен жүзеге асатын функцияларды және деректердің ағымы сипаттау үшін қолданылады;
- Варнье-Орра диаграммалары жүйелердің иерархиялық құрылымын оны кішігірім құрама бөліктерге, процестерге бөлу және әрбір процесс үшін деректердің ағымы көрсету арқылы қолданылады.

Аталған үлгілер ақпараттық жүйелердің сипаттамасын береді. Кеңінен қолданылатын мән-байланыс диаграммалары 6-бөлімде қарастырылған. Маңызды және CASE-құралдарда жиі қолданылатын диаграммалар және DFD және SADT үлгілерін қысқаша қарастырайық.

Деректер ағымы диаграммалары

Деректер ағымы диаграммалары DFD деректердің ағымын модельдеу әдістемесінің негізін қалайды, онда жүйелер үлгісі түрлендіру процесін сипаттайтын деректер ағымы диаграммасы иерархиясы ретінде құрылады.

Жоғары деңгейдегі иерархия диаграммалары немесе контекстік диаграммалар негізгі процестерді анықтайды. Оларда неғұрлым төменгі деңгейдегі диаграммалар көмегімен декомпозиция орындалады, элементар процестерге дейін.

Деректер ағымы диаграммасының негізгі компоненттері:

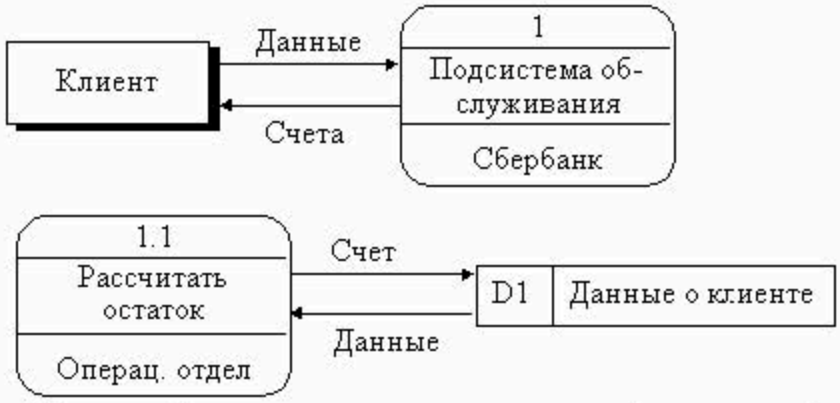
- *сыртқы болмыстар* – Түпнұсқа немесе ақпараттық ағымды (деректердің ағымын) тудыратын немесе қабылдайтын ақпараттарды тұтынушылар;
- *жүйелер/ішкі жүйелер* - қабылданған ақпаратты өзгертетін және жаңа ағымдар тудыратын жүйелер;
- *процестер* - енгізілген деректер ағымын нақты бір алгоритмге сійкес шығарылатын деректер ағымына түрлендіретін процестер;
- *деректерді жинақтауыштар* - ақпараттарды сақтауға арналған абстрактілі құрылғы, оны жинақтауышта сақтауға және қайсыбір уақытта шығарып алуға болады;

• *деректердің* ағымдары - қайсыбір байланыстыру арқылы түпнұсқадан қабылдаушыға жіберілетін ақпаратты анықтауға арналған.

Графикалық блоктардың типтік жинағын сипаттайық, олар әдетте DFD компонент-терін белгілеу үшін қолданады. Нақты CASE-құралдарда және жүйелерде бұл жинақтардың қайсібір айырмашылықтары болуы мүмкін.

Сыртқы мағынасы көлеңкелі тіктөртбұрышпен белгіленген. *Жүйелер және ішкі жүйелер* келесі өрістері бар тіктөртбұрышпен бейнеленген: номері, анықтаулармен және толықтырулармен атауы және жобалаушы аты. *Процесс* үшбұрыш арқылы келесі өрістермен бейнеленген: номері, аты және физикалық жүзеге асуы (қай бөлімше, бағдарлама немесе құрылғы қандай процесті орындайтынын көрсетеді). *Деректердің жинақтауышы* оң (немесе оң және сол) жақ сызығы жоқ тіктөртбұрышпен бейнеленген: идентификатор (D әрпі және санмен) және аты (сақталатын деректерді көрсетеді). *Деректердің ағымы* бағыттауышы бар сызықпен көрсетілген, ол ағымның бағытын көрсетеді және оның құрамын бейнелейтін атауы бар.

Деректер ағымы диаграммаларының мысалы аталған компоненттермен 7.1-суретте келтірілген.



7.1-сурет. Деректер ағымы диаграммасы фрагменттері

Деректер ағымы диаграммасын тұрғызу контексті диаграммалар тұрғызудан басталады. Қарапайым ақпараттық жүйелер жағдайында бір контексті диаграммамен шектелуге болады.

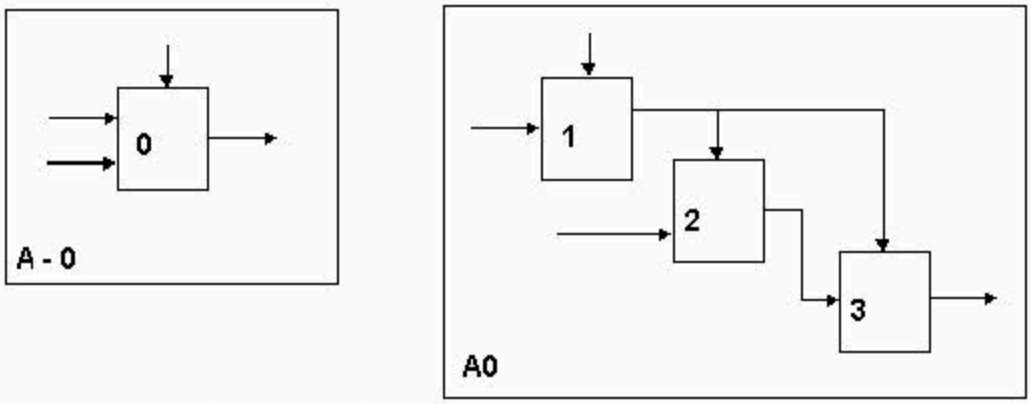
Функционалдық модельдеу әдістемесі

Функционалдық модельдеу әдістемесі SADT функционалды үлгілер объектісін немесе зерттеу облысын тұрғызу үшін қолданылады. Соңғысы функционалдық құрылымын объектісінің орындайтын әрекеттерін және олардың арасындағы байланыстарды сипаттайды.

Функционалдық үлгі диаграммалар, мәтін фрагменттері және глоссарийлердің бір-біріне сілтемелерінен тұрады. Диаграммаларда АЖ функциялары және олардың арасындағы блоктар және доғалар түріндегі өзарабайланыстар (интерфейстер) көрсетіледі. Доға мен блоктың байланысқан жері интерфейс типін анықтайды. Басқарушы ақпарат жоғарыдан, өңделетін ақпарат – блоктың сол жағынан, шығарылатын ақпарат – блоктың оң жағынан көрсетіледі, операцияны орындайтын механизм (адам, бағдарлама немесе құрылғы) блоктың төменгі жағынан доғамен ұсынылады (7.2-сурет).

7.2-сурет. Функциональдық блок және интерфейс доғасы

SADT әдістемесін қолданған кезде ақпараттық жүйелер үлгілерін тұрғызу үшін бөлшектеу деңгейлерін біртіндеп жалғастыру орындалады. 7.3-суретте жүйелердің бастапқы блогын үш құраушы компоненттерге декомпозициялау көрсетілген. Блоктардың әрқайсысы бастапқы функциялардың ішкі функцияларын анықтайды.



7.3-сурет. Диаграммалар декомпозициясы

Жалпы жағдайда АЖ функциональдық үлгісі күрделі объектіні құрамдас компоненттерге жіктейтін, құжаттамалары бар диаграммалар сериясы болып саналады. Диаграммалардағы блоктар номерленеді. Диаграммалар иерархиясында диаграммалар немесе блоктардың орналасуын көрсету үшін диаграмма номері қолданылады. Мысалы, А32 белгілеуі А3 диаграммасындағы 2-блокты көрсетеді. Өз кезегінде, А03 диаграммасы А0 диаграммасындағы 3-блок бөлшектенуі.

Диаграммаларда функциональды үлгілер SADT тізбегі және уақыт анық көрсетілмесе, онда кері байланыстар, итерациялар, процестер және уақыт бойынша функцияларды доға көмегімен белгілеуге болады.

Функциональдық модельдеу әдістемесінің негізгі қасиеті функциялар арасындағы байланыстардың бейнеленуінің барлық мүмкін типтері болып табылады. байланыстардың өсу реті бойынша келесі жеті типін атап көрсетуге болады:

- *кездейсоқ байланыстар*, функциялар арасында байланыс аз немесе жоқ екендігін білдіреді;
- логикалық *байланыстар*, деректер және функциялар элементтердің бір класы немесе жинағына жататынын, бірақ олардың арасында функциональдық қатынастардың жоқ екендігін білдіреді;
- *уақытша байланыстар* уақыт бойынша байланысты функцияларды ұсынады, деректер бір уақытта қолданылатын немесе функциялар параллель қамтылатын жағдай;
- *процедуралық байланыстар* функциялардың топтастырыл-ғандығын білдіреді, өйткені олар цикл немесе процестің бірдей бөлімдерінде қатар орындалады;

- *коммуникациялық байланыстар* функциялардың топтастырыл-ғандығын білдіреді, өйткені олар бірдей кіріс деректерін қолданады және/немесе бірдей шығыс деректерін тудырады;
- *тізбектелген байланыстар* себеп-салдарлық тәуелділікті белгілеу үшін қолданылған – бір функциялардың шығыс деректері басқа функциялардың кіріс деректері болады;
- *функциональдық байланыстар* функциялардың барлық элементтерінің тек бір ғана функцияның орындалуына әсер ететін жағдайын білдіреді.

7.4. Объектілі-бағытталған үлгілер

Объектілі-бағытталған жобалау үлгілері мүмкіндіктері бойынша көрініс үлгісіне ұқсас, бірақ айырмашылықтары да бар. Объектілі-бағытталған технологиялардың кең таралғандығы көптеген белгілі үлгілерді жақындастырды. Үлгілердің көптүрлілігі жобалаушыларға үлгілерді таңдау және әртүрлі жобалармен жұмыс жасау кезінде ақпараттармен алмасуда қиындықтар тудырады. Осы байланыстарды белгілі мамандар Г.Буч, Д.Рамбо және И.Джекобсон Rational Software Corporation фирмасының қолдауымен қарастырып, UML (Unified Modeling Language) деп аталатын модельдеу тілін ұсынды.

UML жалпы сипаттамасы

UML бағдарламалық жүйелер материалдарын мамандандыруға, құрастыруға және құжаттауға, сонымен қатар бизнес және басқа бағдарламалық емес жүйелерді модельдеуге арналған модельдеудің ортақ тілі болып саналады. UML құру негізінде үш неғұрлым кеңінен таралған үлгілер жатыр:

- Booch, Гради Буч (Grady Booch) автор фамилиясымен аталған;
- OMT (Object Modeling Technique – объектілерді модельдеу әдісі);
- OOSE (Object-Oriented Software Engineering – объектілі-бағытталған жобалауды бағдарламалық қамтамасыз ету);

Өңдеу және қабылдаудың соңғы кезеңінде UML стандартына OMG (Object Management Group – объектіні басқару тобы) консорциумы үлкен үлес қосты. UML-ді объектілі-бағытталған *модельдеу стандарты* ретінде анықтауға болады. Ол визуальды (графикалық) модельдеудің әдістерін іске қосады. Қазіргі уақытта UML сүйемелдейтін көптеген аспаптық құралдар бар, солардың ішінде ерекше атауға болатындары: Rational Rose, Select Enterprise, Platinum және Visual Modeler.

UML диаграммалары түрлері

UML көмегімен құрылған ақпараттық жүйелер жобасы 8 түрлі диаграммаларды (diagrams) қамтуы мүмкін:

- пайдалану жағдайлары (use case),
- кластар (class),
- жағдайлар (statechart),
- белсенділіктер (activity),
- ілесулер (sequence),
- ынтымақтастық (collaboration),
- компоненттер (component),
- орналасу (deployment).

Жағдайлар, белсенділіктер, ілесулер және *ынтымақтастық* диаграммалары өңделетін ақпараттық жүйелердің өзгерістерін сипаттау үшін диаграммалар жиынтығын құрастырады. Сонымен қатар, соңғы екеуі ақпараттық жүйелердің *объектілерінің өзара*

әрекетін сипаттауды қамтамасыз етеді. *Компоненттер* және *орналасу* диаграммалары ақпараттық жүйелердің физикалық жүзеге асуын сипаттайды.

Диаграммалардың әрқайсысы элементтедің белгілі түрінен тұруы мүмкін. Элементтердің мүмкін түрлері және олардың арасындағы қатынастар диаграммалар түрінен тәуелді. Диаграммалардың көрсетілген түріне неғұрлым жан-жақты сипаттама береміз.

Пайдалану жағдайлары диаграммалары АЖ қызметін сипаттайды. Әрбір функциональдылық пайдалану жағдайлары түрінде бейнеленеді. Жағдай – бұл пайдаланушының жүйелермен өзара әрекеті, ол келесілерді орындайды:

- пайдаланушы функциясын сипаттайды,
- әртүрлі деңгейде бөлшектеуді көрсетеді,
- нақты мақсатқа жетуді қамтамасыз етеді.

Жағдай доға түрінде бейнеленеді, ол "актер" (actors) деп аталатын әдеттегі пайдаланушылармен байланыстырады. Актер өзара әрекеттесетін жүйелерден тысқары кезкелген деген мағынаны білдіреді, мысалы адам, құрылғы, басқа жүйелер. Жағдай актерге жүйелердің ұсынатындарын сипаттайды, ақпараттық жүйелермен актердің сұқбаттасуы кезінде транзакциялар жинағын анықтайды. Диаграммада бір актер бейнеленеді, бірақ пайдаланушылардың актер ретінде ұсынатындары бірнеше болуы мүмкін. Пайдалану жағдайлары диаграммаларының абстракция деңгейі жоғары емес және АЖ-ге қойылатын функциональдық талаптарды анықтауға мүмкіндік береді.

Кластар диаграммалары кластардың *статистикалық* құрылымын сипаттайды. Кластар диаграммалары құрамына келесі элементтер кіреді: кластар, объектілер және олардың арасындағы қатынастар. Класс үш бөлімнен тұратын тіктөртбұрышпен ұсынылады: класс аты, атрибуты және операциялары. Ұқсас белгілеулер объектілер үшін де қолданылады, айырмашылығы - класс атына объект аты қосылады және барлық жазудың асты сызылады.

Қосымша ақпараттарды (абстрактылық операциялар және кластар, стереотиптер, жалпы және дербес әдістер, интерфейстер, және т.с.с.) бейнеленуге рұқсат беріледі. Ассоциация немесе кластар арасындағы статистикалық байланыстар байланыстырушы сызықтар түрінде бейнеленеді, онда ассоциацияның қуатын, бағытын, атауын, және мүмкін шектеулерді көрсетуге болады. Ассоциацияның қасиеттерін, мысалы, агрегация қатынасын (құрамы басқа кластар бөлігі болатын) көрсетуге болады. Агрегация қатынасы ромб түрінде бейнеленеді. Жалпылау қатынасы үшбұрыш және байланыстырушы сызықтар түрінде бейнеленеді, олар мұрагерлік кластардың иерархиясын көрсетуге мүмкіндік береді.

Жағдайлар диаграммалары объектінің уақыт бойынша өзгерісін сипаттайды, объектіге барлық мүмкін өзгерістер енгізуді көрсетеді. Жағдайлар диаграммалары объектінің өзгерісін және кластардың операцияларын сипаттау үшін қолданылады. Диаграмманың бұл түрі бір класс немесе объекті жағдайының өзгерістерін сипаттайды. Объектінің әрбір жағдайы бұрыштары дөңгеленген үшбұрыш түрінде ұсынылады, онда жағдай атауы және берілген уақыт аралығындағы объекті атрибуттарының мүмкін мәні беріледі. Бір жағдайдың жүзеге асуы (мысалы, хабарлама алу немесе сигнал қабылдау) бағыттауыш арқылы көрсетіледі. Онда жағдай атауы мен сыртқы жағдай сұранысына жауап ретінде объектімен жүзеге асатын әрекет көрсетеледі.

Белсенділіктер диаграммалары жағдайлар диаграммаларының жеке жағдайы. Әрбір жағдай қайсібір операцияларды орындайды және келесілерге өтуді жүзеге асырады. Сонымен қатар, процедуралық, синхронды басқару орындалады.

Белсенділіктер диаграммалары үшін жағдайлар диаграммасына ұқсас белгілеулер қолданылады, бірақ жағдай туралы мәліметтер жоқ және параллель алгоритмдердің жүзеге асуы үшін келесі кезеңге өтудің "синхронизациясы" символы қосылған.

Белсенділіктер диаграммалары көбінесе кластардың операцияларын сипаттау үшін қолданылады.

Ілесулер диаграммасы жөнелтілетін хабарламалардың уақытша тізбегін (өзара әрекеті динамикасы), хабарлама реті, түрі және атын анықтайды. Диаграммада өзара әрекеттесуге тікелей қатысатын объектілер бейнеленеді және басқа объектілермен статистикалық ассоциациясы көрсетілмейді.

Ілесулер диаграммасы екі өлшемге ие. Біріншісі – солдан оңға қарай, өзара әрекеттесуге қатысатын объектілерді бейнелейтін вертикальды сызықтар түрінде. Сызықтардың жоғарғы бөлімі объект классы аты немесе объект үлгісі атын қамтитын үшбұрышпен толықтырылады. Екінші өлшемі – вертикальды уақытша ось. Жөнелтілетін хабарлама хабарлама атымен бағыттауыш түрінде бейнеленеді және пайда болған уақыт бойынша реттелген.

Ынтымақтастық диаграммалары жүйелер объектілерінің қайсыбір нәтиже алу үшін өзара әрекетін сипаттайды. Нәтиже алу дегеніміз әрекетті орындаудың толық аяқталуы.

Ынтымақтастық диаграммалары өзара әрекеттесуге қатысатын объектілерді үшбұрыш түрінде бейнелейді, онда объект аты, оның класы және атрибуттардың мүмкін мәндері болады. Ассоциацияның объектілері арасында байланыстырушы сызықтар түрінде бейнеленеді. Ассоциацияның аты және берілген ассоциациядағы объектілер ролі көрсетілуі мүмкін. Динамикалық байланыстар (хабарламалар ағымдары) объектілер арасындағы байланыстырушы сызықтар түрінде беріледі, үстіңгі жағында бағыты мен хабарлама аты жазылады.

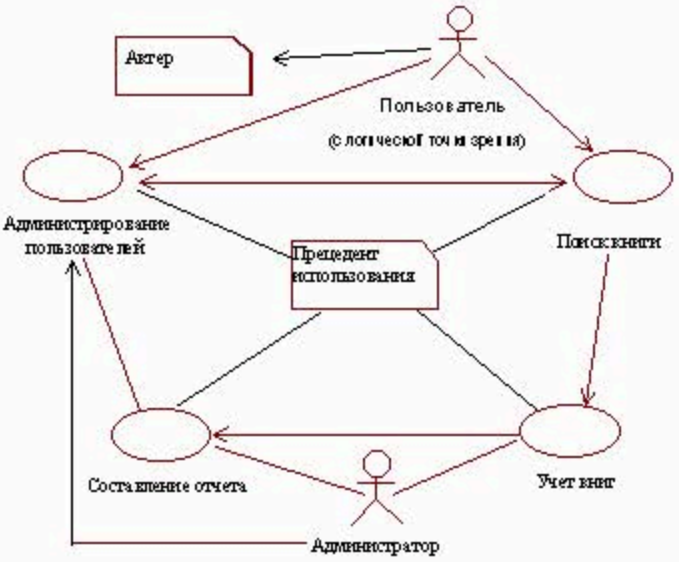
Компоненттер диаграммасы бағдарламалық компоненттер арасындағы тәуелділікті тағайындау жолымен құрылатын жүйелер архитектурасын анықтау үшін қолданылады: бастапқы, бинарлық және/немесе орындалатын кодты. Көптеген ортада өңдеу модулі, немесе компоненті файлғы сәйкес келеді. Модульдерді байланыстыратын үзік сызықты бағыттауыштар қатынастардың өзара тәуелділігін (компиляциядағы сияқты) көрсетеді.

Орналасу диаграммалары компоненттер, процестер және объектілер конфигурациясы тапсырмалары үшін қолданылады. Сонымен қатар, олар жүйелердің жұмыс жасауына қатынасатын аппараттық құрылғылардың физикалық тәуелділігін көрсетеді.

UML диаграммалары мысалы

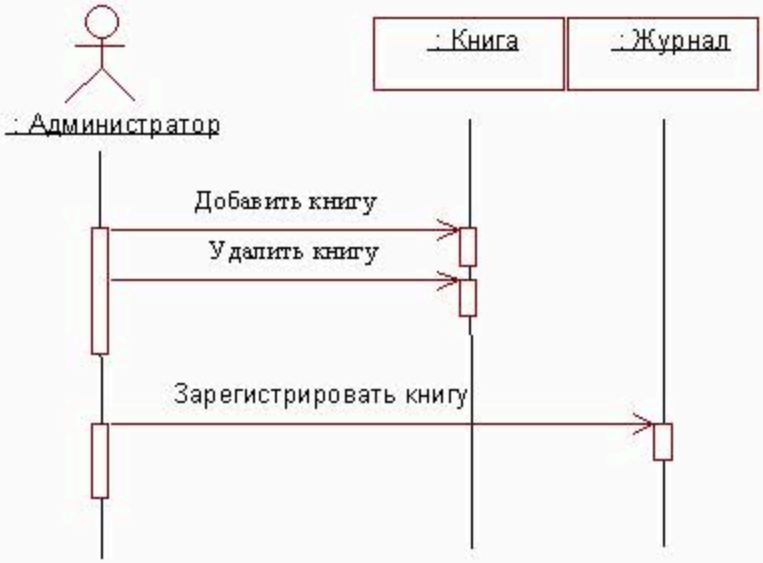
Неғұрлым көрнекі бейнелену алу үшін UML диаграммаларын қарастырайық. Мысалы, объектілі үлгі сипатталған Rational Rose 98 қатынасы. Зерттеу облысы ретінде кітапхана жұмысын сипаттауды қолданамыз, онда клиенттерден әртүрлі баспахана өнімдеріне сұраныстар жасалған және олардың кітапхана қорына қайтарылғандығы туралы ақпараттар тіркелген.

Пайдалану жағдайлары диаграммалары мысалы 7.1-суретте келтірілген. Диаграммада талдау кезінде жүзеге асатын ақпараттық жүйелердің функциялары ерекшеленіп келтірілген: пайдаланушыларды басқару (Administrative Client); кітаптарды тіркеу (Administrative Books); есеп берулер құру (Report) және кітапты іздеу (Find Book).



7.1-сурет. Пайдалану жағдайлары диаграммалары

Ілесулер диаграммалары мысалы 7.2-суретте келтірілген. Келтірілген диаграмма объектілердің уақыт бойынша өзгерісін сипаттайды. Ол объектілер және объектілер жөнелткен хабарламалар тізбегін көрсетеді.



7.2-сурет. Ілесулер диаграммасы

Ескере кетейік, АЖ үлгілерін тұрғызудан бастап оны бағдарламалық өңдеуге дейін жобалаудың қажетті кезеңі болып табылады. Жақсы құрылған үлгілерде тапсырыс берушілер, пайдаланушылар және бағдарлама құрушылар арасында конструктивті өзара әрекет қалыптасқан. UML диаграммалары құрылатын жүйелер үшін архитектуралық шешімдердің айқын бейнеленуін қамтамасыз етеді.

7.5. CASE-құралдарды жіктеу

CASE-құралдарды жіктеу кезінде келесі белгілер қолданылады:

- өміршеңдік кезеңі кезеңдеріне бейімделу;

- функциональдық толықтығы;
- қолданатын үлгілер түрі;
- ДҚБЖ-дан тәуелсіздік деңгейі;
- ұйғарымды платформалар.

CASE-құралдардың жіктелуін неғұрлым жиі қолданылатын белгілер бойынша қарастырайық.

Өміршеңдік кезеңі кезеңдеріне бейімделуі бойынша CASE-құралдардың келесі негізгі типтерін бөліп көрсетеді:

- зерттеу облысы үлгілерін тұрғызу және талдауға арналған *талдау құралдары*, мысалы: Design/IDEF (Meta Software) және BPwin (Logic Works);
- маманданған жобалар құруды жүзеге асыратын *талдау және жобалау құралдары*, мысалы: Vantage Team Builder (Cayenne), Silverrun (Silverrun Technologies), PRO-IV (McDonnell Douglas) және CASE талдау жасаушы (МакроПроджект);
- негізгі ДҚБЖ деректер қоры үшін сызбасын құру және өңдеуді жүзеге асыратын *деректер қорын жобалау құралдары*, мысалы: ERwin (Logic Works), S-Designor (SPD), DataBase Designer (ORACLE);
- *қосымшаны өңдеу құралдары*, мысалы: Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Centura) және Delphi (Borland).

Функциональдық толықтығы бойынша CASE-жүйелер және құралдарды шартты түрде келесі типтерге жіктеуге болады:

- өміршеңдік кезеңнің бір немесе бірнеше кезеңдерінің дербес тапсырмаларын шешуге арналған жүйелер, мысалы, ERwin (Logic Works), S-Designor (SPD), CASE талдау жасаушы (МакроПроджект) және Silverrun (Silverrun Technologies);
- АЖ өміршеңдік кезеңін сүйемелдейтін және ортақ репозиториймен байланысты *біріктірілген жүйелер*, мысалы, Vantage Team Builder (Cayenne) және Designer/2000 жүйелері мен қосымшаны өңдеу Developer/2000 (ORACLE) жүйесі.

Қолданатын үлгілер түрі бойынша CASE-жүйелерді шартты түрде үш түрге бөлуге болады: құрылымдық, объектілі-бағытталған және біріктірілген.

Тарихи тұрғыда алғашқы құрылымдық CASE-жүйелер құрылымдық және модульдік бағдарламалау әдістеріне, құрылымдық талдау және синтезге негізделген, мысалы, Vantage Team Builder (Cayenne) жүйелері.

Объектілі-бағытталған әдістер және CASE-жүйелер 90-шы жылдардан бастап жаппай қолданыла бастады. Олар өңдеу уақытын азайтты, сонымен қатар АЖ жұмыс жасау сенімділігі және тиімділігін арттырды. Объектілі-бағытталған CASE-жүйелер мысалдары: Rational Rose (Rational Software) және Object Team (Cayenne).

Біріктірілген құрал-саймандық құралдар бір уақытта құрылымдық және объектілі-бағытталған әдістерді сүйемелдейді, мысалы: Designer/2000 (ORACLE).

ДҚБЖ-дан тәуелсіздік деңгейі бойынша CASE-жүйелерді екі топқа бөлуге болады:

- тәуелсіз жүйелер;
- ДҚБЖ-ға кіріктірілген.

Тәуелсіз CASE-жүйелер нақты ДҚБЖ құрамына кірмейтін жүйелер түрінде қамтамасыз етіледі. Әдетте олар ODBC интерфейсі арқылы деректер қорының бірнеше форматын сүйемелдейді. Санынан тәуелсіз CASE-жүйелерге жатады: S-Designor (SDP, Powersoft), ERwin (LogicWorks) және Silverrun (Computer Systems Advisers Inc.).

Кіріктірілген CASE-жүйелер әдетте ДҚБЖ құрамына кіретін деректер қоры форматын сүйемелдейді. Сонымен қатар, деректер қорының басқа форматтарын да сүйемелдеуі мүмкін. Мысалы: ORACLE ДҚБЖ құрамына кіретін Designer/2000.

Неғұрлым кең таралған CASE-жүйелерді қарастырайық.

7.6. Құрылымдық типті жүйелер

CASE-жүйелердің құрылымдық типтерін қарастыру кезінде екі негізгі топтарды атауға болады: тәуелсіз және кіріктірілген жүйелер.

Тәуелсіз жүйелер

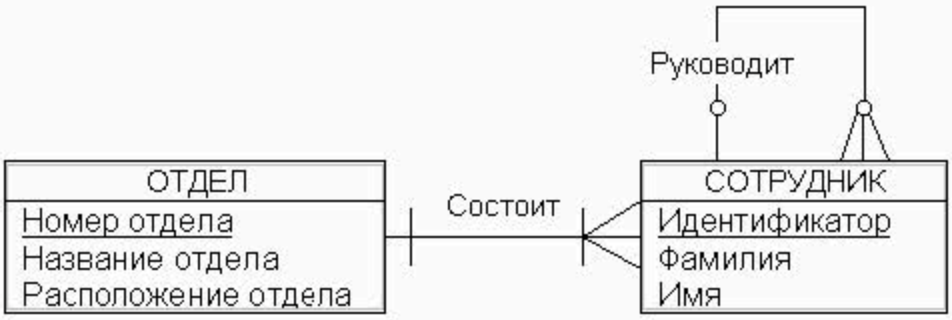
Тәуелсіз CASE-жүйелердің құрылымдық типіне кең таралған S-Designor (фирмы SDP, приобретенной Powersoft) өнімдерін, ERwin (LogicWorks) және Silverrun (Computer Systems Advisers Inc.) дестелерін жатқызуға болады.

S-Designor графикалық тұрғыдан құрал-жабдық болып саналады. S-Designor 6.0 нұсқасынан бастап өнім PowerDesigner 6.0 атауымен шығарылады.

ДҚ құрылымын S-Designor көмегімен құру кезінде тұжырымдамалық деректер үлгісі (ТДҮ) қалыптасады, ол кейінірек физикалық деректер үлгісіне (ФДҮ) өзгертіледі.

Тұжырымдамалық деректер үлгісін сипаттау үшін MS Windows стилінде графикалық интерфейстің қолайлы құралдары ұсынылады. *Тұжырымдамалық деректер үлгісі* ER-үлгілер түріндегі деректер қоры сызбасы болып саналады.

Мағынасы үшбұрышпен бейнеленеді, оның ішіне атрибуттар орналасқан. Мағынасын бірмәнді бейнелейтін атрибуттарды (болмыстар идентификаторы) астын сызу арқылы бөліп көрсетеді. Мәндік байланыстар сәйкес үшбұрыштарды жалғастыратын сызықтармен бейнеленеді. Байланыс түрі (1:1, 1:M, M:1, M:M) және болмыстар тәуелділігі сызықтармен белгіленеді. Егер байланыс үшін барлық элементтер болмыстар болса, онда сызық сызылып тасталады, кері жағдайда – сызық орнына дөңгелек салынады. Мысалы, тұжырымдамалық үлгілер түріндегі болмыстар диаграммалары 7.1-суретте келтірілген.



7.1-сурет. Тұжырымдамалық үлгілер мысалы

Тұжырымдамалық деректер үлгісін тұрғызған кезде кестелер бағаналары мәндеріне шектеулер қоюға болады (ең кіші және ең үлкен мәні, үнсіз келісіммен қабылданатын мән және мүмкін мәндер тізімі).

Егер тұжырымдамалық деректер үлгісінде мәндер арасында M:M түріндегі байланыс болса, онда физикалық үлгілер тұрғызған кезде автоматты түрде қосымша кесте құрылады. оның тағайындалуы – қатынастарды қалыпқа келтіру. Кестелер бағаналары болмыстар байланыстарына қатысатын идентификаторлар болады. Алғашқы кілт жаңа кестелер біріктіреді, алғашқы кілттер бағанасы екі бастапқы байланысқан кесте. M:M түріндегі байланыс үшін тұжырымдамалық деректер үлгісінен физикалық деректер үлгісіне көшу мысалы 7.2-суретте келтірілген. «<pk>» түріндегі символдық құрылым бұл бағананың (өріс) кілттік болып табылатынын білдіреді.

7.2-сурет. Функциональдық үлгілерге көшу мысалы

Қарастырылған жүйелер ODBC интерфейсі арқылы жұмыс жасап тұрған ДҚБЖ серверіне қосылып, деректер қорын құруға немесе ДҚ құрылымын құру үшін SQL-операторларының мәтіндік файлын (дестесін) дайындауға мүмкіндік береді. SQL-операторлар файлы бұдан соң қайсібір ДҚБЖ-да өңделеді, нәтижесінде қажетті ДҚ құрылады.

S-Designor көптеген ДҚБЖ интерфейстеріне ие, олар Oracle, Ingress, Informix, Sybase, SQL Server, Access және Paradox.

S-Designor жүйелері Windows ортасында жұмыс жасайды және PowerBuilder, TeamWindows және Progress сияқты басқа аспаптық бағдарламаларды өңдеу құралдарын пайдалану мүмкіндігімен қамтамасыз етеді. PowerBuilder аспаптық жүйелері үшін S-Designor дестесі қосымшаны автоматтты түрде іздестіруді орындайды.

ДҚ құрылымын модельдеу құралдары көмегімен S-Designor жүйелері тура (функциональды үлгілерге) және кері (тұжырымдамалық үлгілерге) көшуді жүзеге асырады. «Кері қарай жобалау» ақпараттық үлгілері талдау есептерін шешу және ААЖ функцияларын кеңейту үшін қажет.

S-Designor жүйелері топтық өңдеуді сүйемелдейді. Жобаланған ақпараттық жүйелердің деректер үлгісі ішкі үлгілерге жіктелуі мүмкін, олардың әрқайсысымен бағдарлама құрушы жұмыс жасайды. Ішкі деректер үлгісі ыңғайлылық деректер қоры үшіннда сақталады. Үлгілерді сақтау үшін кез-келген SQL-ДҚБЖ қолданылуы мүмкін. S-Designor жүйелерінде топтық жұмысты басқару құралдарында құпия сөзбен қорғау бар.

ERwin тұжырымдамалық модельдеу деректер қоры жүйелері болып саналады. ERwin жүйелері ДҚ сызбасын жобалауды жүзеге асырады. Функциональдық мүмкіндіктері бойынша ERwin жүйелері S-Designor жүйелеріне ұқсас. ERwin жүйелерінде ДҚБЖ байланысы тікелей ұйымдастырылады, S-Designor жүйелерінен айырмашылықтары - сыртқы файлдарды қолдануы.

Silverrun ашық жүйелер болып саналады, оны әртүрлі фирмалар өнімдерімен бірлесе қолданамыз. Ол ДҚБЖ интерфейстері емес, оған келесілер жатады: DB2, Informix, Ingress, Oracle, Progress, SQLBase, SQLServer. Silverrun жүйелерінің қолдану облысы – бизнес-класс ақпараттық жүйелердің құрылымдық әдістемесін құрал-саймандық сүйемелдеу. Бұл жүйелер жобалаудың алғашқы кезеңдеріне негізделген. Ол үлгілердің тәуелсіз екі түрін тұрғызуға мүмкіндік береді: функциональдық және ақпараттық. Функциональдық үлгілер түрінде пайдаланушылар-тапсырыс берушілерге негізделген деректер ағымы диаграммасы DFD қарастырылады. Silverrun жүйелері үш негізгі ішкі жүйелерден тұрады: деректер ағымын диаграммасын тұрғызу модулі және екі мән-байланыс түріндегі диаграмма тұрғызу модулі: тұжырымдамалық үлгілер ERX

(Entity Relationship eXpert) модулі және реляциялық үлгілер RDM (Relational Data Modeler) модулі.

DFD-диаграммалар тұрғызу құралдары келесі мүмкіндіктерді береді: DFD нотация түрін таңдау; диаграмма элементтерінің сыртқы түрін өзгерту; ережелер жиынын таңдау және т.б.

Қорыта айтқанда, Silverrun жүйелері өздерінің мүмкіндіктері бойынша S-Designor және Erwin жүйелеріне жақын. Жүйелер әртүрлі ортада топтық өңдеуді сүйемелдейді және Windows, OS/2, Macintosh және Solaris платформаларында жұмыс жасауы мүмкін.

Designer/2000 жүйелері

ORACLE фирмасының Designer/2000 CASE-жүйелері кіріктірілген болып табылады және Oracle ДҚБЖ-да қолданылады. CASE-технология негізінде жүзеге асқан ORACLE өнімдері келесілерді құрайды:

- Жобалаудың құрылымдық әдістемесі;
- қолданбалы жүйелердің өміршеңдік кезеңінің барлық кезеңдерін сүйемелдеу;
- «клиент-сервер» технологиясына бейімделу;
- жобалау кезінде барлық ақпараттарды сақтау үшін орталықтандырылған деректер қорының (репозиторийлер) бар болуы;
- көптеген пайдаланушылардың репозиториймен бір уақытта жұмыс жасау мүмкіндігі;
- өңдеу кезеңдері арасында бір-біріне көшуді автоматтандыру;
- жобалауды автоматтандыру және қосымша құру (құжаттаулар құру, маманданулардың тексеру, бағдарламаларды автоматты түрде іздестіру және т.б.).

Designer/2000 жүйелері қолданбалы жүйелерді өңдеудің келесі кезеңдерін сүйемелдейді: қызмет ұйымдастыруды модельдеу және талдау, зерттеу облысының тұжырымдамалық үлгілерін өңдеу, қосымшаны жобалау және бағдарламаларды синтездеу.

Модельдеу және талдау кезеңінің сүйемелдеу құралдары технологиялық және ұйымдастырушылық процестердің көрнекі үлгілерін және меңгеру мен жетілдіру үшін ұйымдастыру құрылымын құрады. Сонымен қатар, мультимедиа құралдары, дыбыстық сүйемелдеу, видео және анимация кеңінен қолданылады.

Қызметті ұйымдастыру үлгісі ішінара процестерді сипаттайтын диаграммалар жиынтығы түрінде ұсынылады. Диаграммалар стандартты элементтерден құрылады, олардың негізгілері: базалық процесс, процесс қадамы, сақтау қоймасы, ағымдар, ұйымдастырушылық бірліктер және оқиғалар.

Тұжырымдамалық модельдеу кезеңінде зерттеу облысы зерттеу облысы ерекшеліктерін, шығарылатын есептердің ерекшеліктерін, ақпараттық қажеттіліктер және ресурстар, технологиялық шектеулер және т.б. сипаттайтын үлгілерден құрылады. Үлгілердің екі түрі қолданылады: ақпараттық (қолданыстағы ақпараттық құрылымды және олардың арасындағы өзарабайланыстарды бейнелейді) және функциональдық (ақпаратты өңдеу технологиясы мен әдістерін бейнелейді).

Ақпараттық үлгілер негізі мән-байланыс типті бинарлық үлгілерге ұқсас Чен үлгілерінің арнайы түрі болып табылады. Бұл үлгілерде екі болмыстар арасында анықталған өзара байланыстар болуы мүмкін.

Функциональдық сипаттаудың зерттеу облысы функциялар иерархиясы диаграммалары және деректер ағымы үлгілері көмегімен жүзеге асады. Үлгілердің бірінші түрі ортақ функцияларды ішкі функцияларға декомпозициялауды жобалайды, олардың әрқайсысы, өз кезегінде, неғұрлым кіші функцияларға жіктеледі және т.с.с. Қажеттілік кезінде нақты функцияларды орындауға шақыратын оқиғаларды сипаттауға болады. Деректер ағымы диаграммалары ұйымдастырушылық құрылымдардың жұмысы кезінде деректердің қозғалысын сипаттайды.

Designer/2000 тұжырымдамалық модельдеу жүйелерінде графикалық редакторлар жиынтығы келесілерді сүйемелдейді: ER-диаграммалар, функция иерархиясы және деректер ағымы диаграммасы. Көрініс үлгілерінен басқа, диаграмма элементтері туралы қосымша ақпаратты енгізу, диаграммаларды толықтылық мен шынайылыққа семантикалық тексеруді орындау, тұжырымдамалық модельдеу бойынша есеп беруді және құжаты алу редакторы бар.

Жобалау кезеңінде алынған тұжырымдамалық үлгілерден ДҚ құрылымын және құрамын, сонымен қатар модульдердің бағдарламалық жинағын сипаттайтын қолданбалы жүйелердің техникалық мамандануы қалыптасады. Құрылған мамандануларақпараттық және функциональдық болып бөлінген.

ДҚ құрылымы және құрамын сипаттау келесілерді іске қосады: ДҚ кестелері тізімі, әрбір кестелер бағаналары (өрісі) құрамы, кілттік өрістер құрамы, индекстер құрамы, бағаналар мәніне қойылатын шектеулер, тұтастыққа қойылатын шектеулер және т.б.

Құрылатын қосымшаны функциональдық сипаттау пайдаланушы интерфейсі мәзірі құрылымын, экрандық формаларды, есеп берулерді, процедуралық модульдерді және басқаларды жобалайды.

Мамандандудың алғашқы нұсқасын арнайы қосымшаларды қолданып алуға болады.

Жобалау кезеңі бағдарламалар сызбасы, өзарабайланысқан модульдер диаграммасы және модульдер сызбасы редакторлары көмегімен жүзеге асады. Диаграммаларды тұрғызудан басқа аталған редакторлар диаграмманың жекелеген элементтері туралы қосымша ақпаратты қолданады.

Бағдарламалар құру кезеңінде бағдарламалық код генераторы қолданылады, олар бұл кезеңді автоматтандырады, өңдеу уақытын айтарлықтай азайтады, алынатын өнімнің сапасын және сенімділігін арттырады. Жүйелерде қолданылатын генераторлар екі топтарға бөлінеді: серверлік бөлімдер генераторы және клиенттік бөлімдер генераторы.

Серверлік бөлімдер генераторы ДҚ мамандануы бойынша автоматты түрде SQL тілінде бағдарламалар мәтінін (ДҚ сызбасын анықтау операторы, триггерлер, сақталатын процедуралар және т.б.) құрады. *Клиенттік бөлімдер генераторы* мамандануы бойынша автоматты түрде бағдарламалық модульдер мәтінін (экрандық формалар, есеп берулер, процедуралар және басқалар) синтездейді. Модульдердің әрбір типі үшін өз генераторы бар.

Генераторлардың жұмысын қосымшаның сыртқы бейнеленуін немесе мәтіндерді безендіруді, жұмыс жасау режимін және т.б. өзгертуге мүмкіндік беретін төрт жүзден астам параметрлері бар, сонымен қатар, алынған бағдарламалар мәтінін өзгертуге мүмкіндік беретін төменгі деңгейдегі өңдеу құралдары бар тапсырмалар арқылы басқаруға болады.

7.7 Объектілі-бағытталған жүйелер

Объектілі-бағытталған CASE-жүйелер объектілі-бағытталған тәсілдеменің құрылымдық тәсілдемеге қарағанда көптеген артықшылықтары бар болғандықтан пайда болды. Ол үш маңызды қасиеттерге негізделген: инкапсуляция, мұрагерлік және полиморфизм.

Инкапсуляция деректерді және алгоритмдерді (функция және әдіс) оларды өңдеу, сонымен қатар объектілер ішінде деректерді жасыру үшін біртұтас етіп біріктіруді білдіреді. ол құрылатын бағдарламалық қамтамасыз етудің сенімділігін арттыруға мүмкіндік береді. *Мұрагерлік* және *полиморфизм* қасиеттері бағдарламаларды өңдеу процесін жылдамдатады, сонымен қатар жүйелердің қолданылған жаңа шарттарға бейімделуін жеңілдетеді.

Объектілі-бағытталған аспаптық жүйелердің қолдану облысы операциялық жүйелер, қосымшаны өңдеу құралдары және нақты уақыт жүйелерін құру сияқты күрделі жобалар болады.

Заманауи объектілі-бағытталған CASE-жүйелерді екі негізгі топтарға бөлуге болады: бірнеше объектілі-бағытталған үлгілерді сүйемелдейтін CASE-құралдар және үлгілердің тек қана бір түріне бағытталған құралдар. Бірінші түрдегі жүйелерде әдетте бір үлгілерден басқа үлгіге көшу мүмкіндігі бар. Кей жағдайда осы жүйелерде өзіндік нотация құру мүмкіндігі ұсынылады.

Объектілі-бағытталған Rational Rose жүйелері

Rational Rose жүйелері Rational Software Corporation фирмасының объектілі-бағытталған CASE-жүйелерінің бірі болып саналады. Ол БҚЕ талдау және жобалауды автоматтандыру, әртүрлі тілдердегі кодтарды іздестіру және құжаттандыруды дайындау үшін керек. Сонымен қатар, оның құрамында жаңа жобаларда бағдарламалық компоненттерді қайталап қолдануды жүзеге асыратын бағдарламалар реинжинирингі құралдары бар. Осы жүйелерде объектілі-бағытталған талдау және Г. Буча, Д. Рамбо және И.Джекобсон жобалаудың синтез-әдістемесі, UML модельдеудің сәйкестендірілген тілі (7.4-бөлім) қолданылады.

Жүйелердің нақты нұсқасы бағдарламалар коды генерациясы орындалатын тілмен (C++, Smalltalk, PowerBuilder, Ada, SqlWindows және ObjectPro) анықталады. Жүйелердің негізгі нұсқасы Rational Rose/C++ болып табылады, ол бағдарламалық кодты C++ тіліне сәйкестендіруге мүмкіндік береді.

Rational Rose көмегімен жұмыс кезінде диаграммалар тұрғызу және үлгілердің логикалық және физикалық құрылымын, оның статистикалық және динамикалық қасиеттерін анықтайтын мамандануларды тұрғызу орындалады. Олардың құрамына келесі диаграммалар кіреді: класстар, жағдайлар, сценарийлер, модульдер және процестер диаграммалары.

Жүйелердің негізгі компоненттері:

- объектілі-бағытталған ДҚ ұсынатын репозиторий;
- графикалық тұрғыдан пайдаланушының интерфейсі;
- жоба элементтері бойынша, сонымен қатар кластардың және ішкі жүйелердің иерархиясы бойынша, диаграммалар түрлері арасында орын ауыстыруды жүзеге асыратын жобаны көру құралдары;
- қателіктерді тауып, жоятын жобаны бақылау құралдары;
- статистикалық мәліметтер жинау құралдары;
- репозиторий ақпараттары негізінде шығыс құжаттары мәтінін қалыптастыруға мүмкіндік беретін құжаттар генераторы.

Сонымен қатар, әрбір бағдарламалау тілі үшін өзіндік код генераторы қосылады. Бағдарламалар кодын C++ негізінде автоматты түрде іздестіру құралдары бастапқы файлдар мен кластардың және объектілердің сипаттамалық файлдарын қалыптастырады. Осылайша алынған бағдарламалар қаңқасын C++ тілінде тікелей бағдарламалау арқылы толықтыруға болады.

Талдаулар бастапқы мәтіндердің дұрыстығын бақылауды жүзеге асырады және қателіктерді болжайды. Нәтижесінде алынған үлгі бірнеше жобаға қолданылуы мүмкін.

Өңдеу нәтижесінде Rational Rose көмегімен келесі диаграммалар қалыптасады: кластар, жағдайлар, сценарийлер, модульдер және процестер. Сонымен қатар, келесі компоненттер құрылады:

- кластардың, объектілердің, атрибуттар және операциялардың мамандануы;
- бағдарламалардың дайын мәтіндері;

- бағдарламалық жүйелер үлгісі.

Бағдарламалық жүйелер үлгісі жоба туралы барлық ақпаратты қамтитын мәтіндік файл болып саналады. Бағдарламалардың дайын мәтіндері бастапқы файлдар түрінде қалыптасады. Жүйелер бағдарламалық файлдарда комментарийлерді іске қосады.

7.8. CASE-жүйелерді пайдалануға ұсыныстар

Көптеген заманауи CASE-жүйелердің сипаттамалары және мүмкіндіктерін талдау келесі тұжырымдарға келуге мүмкіндік береді.

1. CASE-жүйелер өңдеуді жылдамдатады және жеңілдетеді, құрылатын бағдарламалар және ақпараттық жүйелердің сапасын арттырады. Көптеген CASE-жүйелерде жобамен топтық жұмыс жасауды басқару құралдары бар.

2. CASE-жүйелер өңдеудің алғашқы кезеңдерінде ерекше пайдалы болады. Олар бағдарлама құрушының құрал-жабдықтарында міндетті емес түрде болады және әзірше ДҚБЖ жобалау және өңдеу құралдарын алмастыра алмайды. Бұның бір себебі қосымшаны өңдеу құралдарының, бағдарламалық-аппараттық платформалар және жобалау әдістемелерінің сантүрлілігі болып табылады.

3. Көптеген ұсынылатын CASE-жүйелерде ДҚ тұжырымдамалық үлгілерінен физикалық үлгілеріне және кері қарай өту мүмкіндігі бар.

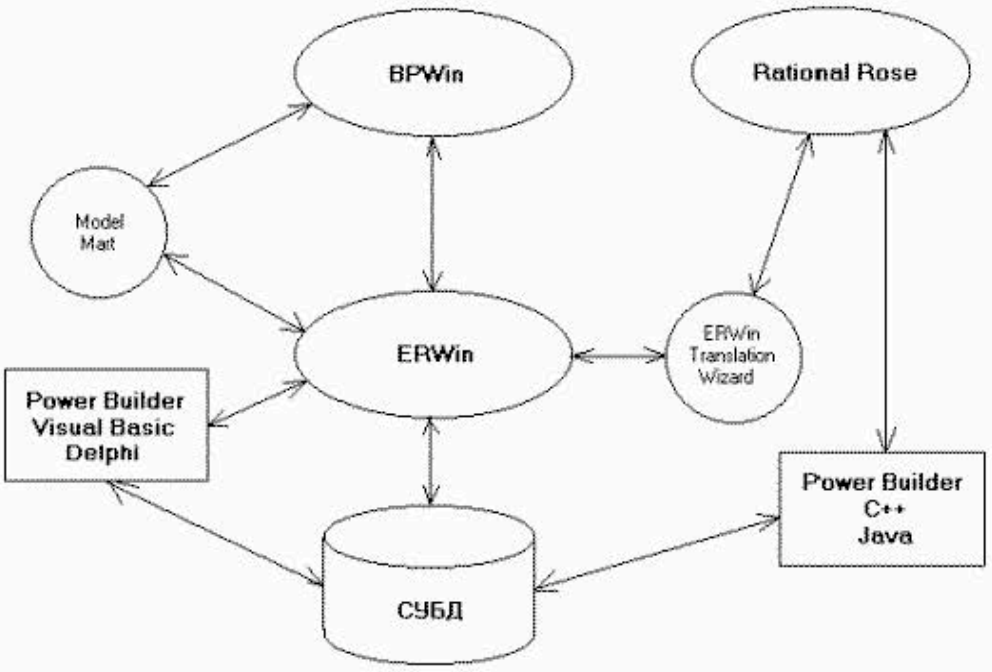
4. Көптеген заманауи CASE-жүйелер құрылымдық болады, бірақ объектілі-бағытталған жүйелердің қайсыбір артықшылықтары арқасында олар танымал.

5. Заманауи CASE-жүйелер мамандарған пайдаланушыға негізделген, себебі оларды пайдалану деректер қоры үшінн жобалауды білу талап етілген. Ендеше, мысалы, ДҚ құрылымын S-Designor жүйелер көмегімен өңдеу үшін жобаланған ақпараттық жүйелер туралы ақпаратты ER-үлгілер түрінде беру керек.

Кей жағдайда бірнеше CASE-жүйелерді қолданған жөн.

Бірнеше CASE-жүйелерді қолдану қолданылатын жүйелердің құндылықтарын біріктіруге мүмкіндік береді және тапсырманы орындау уақытын айтарлықтай азайтады. Мысал ретінде ERWin, BPWin және Rational Rose CASE-жүйелерді бірлесе пайдалану сызбасын келтірейік.

7.9-суретте көрсетілгендей әртүрлі CASE-жүйелер бір-бірімен тікелей (ERWin және BPWin) өзара байланысуы немесе қосымша модульдер (Model Mart – топтық өңдеу құралдары, ERWin Translation Wizard – ERWin үлгілердегі импорттау модулі) көмегімен байланысуы мүмкін.



7.9-сурет. ERWin, BPWin және Rational Rose өзара әрекеті сызбасы

Бақылау сұрақтары және тапсырмалар

1. CASE-құралдар және CASE-технологияларға анықтама беріңіз.
2. Өміршеңдік кезең түсінігіне анықтама беріңіз және үлгілердің негізгі нұсқаларын атаңыз.
3. Топтық CASE-жүйелерге қойылатын талаптарды атап шығыңыз.
4. БҚЕ өміршеңдік кезеңінің спираль үлгісін сипаттаңыз.
5. Құрылымдық талдау және жобалауда қолданылатын графикалық көрініс диаграммаларын атап шығыңыз.
6. Деректер ағымы диаграммаларына мысал келтіріңіз.
7. Функциональдық модельдеу әдістемесін сипаттаңыз, диаграммалар декомпозициясына мысал келтіріңіз.
8. UML модельдеуге сәйкестендірілген тіл?
9. Диаграммалар түрін атаңыз.
10. Ілесулер диаграммаларына мысал келтіріңіз.
11. CASE-құралдарды жіктеу белгілерін атаңыз.
12. CASE-жүйелер функциональдық бейімделуі бойынша қандай топтарға бөлінеді?
13. Тәуелсіз CASE-құрылымдық типті жүйелерге мысал келтіріңіз және сипаттама беріңіз.
14. Designer/2000 CASE-жүйелерін сипаттаңыз.
15. Rational Rose CASE-жүйелердің қандай түріне жатады?
16. Rational Rose жүйелерінің негізгі компоненттерін атаңыз.

8. ДЕРЕКТЕР ҚОРЫН ҚОЛДАНУ

Бөлімде деректер қорын қолданудың маңызды мәселелері: баптау және басқару, ақпаратты қорғау қарастырылады. Сонымен қатар, кеңінен қолданылатын мультимедиа құралдарының мүмкіндіктері сипатталады.

8.1. Баптау және басқару

ДҚ қолданатын ақпараттық жүйелермен ойдағыдай жұмыс жасау үшін ДҚБЖ және ДҚ серверін таңдау жеткіліксіз. Ақпараттық жүйелерді жүктеудің алғашқы кезеңінде және оны жүзеге асыру процесінде әртүрлі функцияларды басқару керек. Басқарудың маңызды мәселесі ақпаратты қорғау және пайдаланушылардың қатынас құруына шектеу қою (келесі бөлімді қараңыз) болады. Баптау және басқарудың маңызды мәселелеріне мыналар жатады:

- дискіге файлдардың орналасуы әдісін таңдау;
- дискілік жадыдан қажетті көлемді анықтау;
- ақпараттарды дискіге орналастыру;
- резервтік көшірме алу.

Осы мәселелердің шешімін қысқаша қарастырайық.

Дискіге файлдардың орналасуы әдісін таңдау

Көптеген ДҚБЖ жүйелер администраторы дискіге файлдардың орналасуының екі әдісінің бірін таңдайды: *"таза"* дискіге немесе ОЖ *файлдық жүйелерінде.* Бірінші жағдайда, жекелеген жинақтауыштарда сақталатын деректерді басқару төменгі деңгейлі құралдармен ДҚБЖ өзінде жүзеге асады. Қайсібір ДҚБЖ, мысалы, Ingres және Interbase, міндетті түрде файлдық жүйелерді пайдалануды талап етеді (UNIX ОЖ ортасы). Дискіге файлдардың орналасуының екі нұсқасының артықшылықтары және кемшіліктеріи қарастырайық.

Ақпараттарды ***"таза"*** дискілерге сақтаудың *артықшылығы - сыртқы жады* неғұрлым тиімді қолданылады және қағидаға сәйкес, дискілер арасында ақпарат алмасу *өнімділігі* артады. Сыртқы жадыны 10-20% үнемдеу файлдық жүйелерді ұйымдастыру қажеттілігін жою негізінде жүзеге асады. Ендеше, UNIX ОЖ-де қызметші ақпарат файлдық жүйелердің 10% жуығын құрайды. Жадыдан шамамен осынша орын файлдардың келесі мүмкін кеңейтпелері үшін сақталады. Өнімділігі (әдетте 10%) дискілерге қатынас құру кезінде бағдарламалық қамтамасыз етудің қосымша тобын өшіру негізінде артады.

Көптеген ДҚБЖ дискілермен ***файлдық жүйелер*** арқылы жұмыс жасауды қажет етеді, ол келесі артықшылықтарды қамтамасыз етеді. Біріншіден, файлдық жүйелерді қолдану жоғары *икемделушілікке* ие, өйткені администраторға файлдарға қызмет көрсетудің стандартты құралдары беріледі: резервтік көшірме қосымшалары, архивтеу, қалпына келтіру, сонымен қатар басқа бағдарламалар файлдарымен (редакторлар, антивирустық бағдарламалар, жинақтауыш сапасын бақылау қосымшасы және т.б.) жұмыс жасау мүмкіндігі. Екіншіден, қайсібір жағдайда енгізу/шығару операцияларын орындау арқылы файлдық жүйелер *тиімділеуді* қамтамасыз етеді, оны ДҚБЖ жүзеге асыра алмауы мүмкін. Жеке жағдайда, UNIX жүйелерінде неғұрлым ірі физикалық бірліктерде деректерді кластерлерге бөлу жүзеге асады, ол көбісінде ДҚБЖ жұмыс өнімділігін арттырады.

Дискілік жадыдан қажетті көлемді анықтау

Дискілік жадыдан қажетті көлемді анықтау кезінде деректерді өңдеу үшін ДҚБЖ дискілерде орналасқан қызметтік ақпараттардың үлкен көлемін қолданатынын ескеру

керек. Қызметтік ақпараттарға келесілер жатады: деректер қоры сызбасын сипаттау, индекстер кестелері, уақытша кестелер, хеш-кесте және индекстер үшін алдын-ала бөлінген кеңістік, сұрыптау үшін кеңістік, журнал файлдары, архивтер және басқалар.

Егер ДҚ үшін қызметтік ақпараттардың көлемі туралы нақты ақпарат болмаса, онда деректердің көлемі оның орналасқан дискілік жады көлемінен аспауын ескереміз. Құрылғының істен шығуы кезінде деректерді қалпына келтіру сенімділігін арттыру үшін транзакциялар журналы файлын және архивті деректер қоры сақталған дискіден басқа физикалық дискілерге сақтаған жөн.

Ақпараттарды дискілерге бөліп орналастыру

Ақпараттарды дискілерге бөліп орналастырудың тиімді әдісі негізгі деректерді өңдеуді бір немесе бірнеше дискілерде орындау болып табылады. Мысал ретінде төрт дискіні қолдануды атауға болады: біріншісі – операциялық жүйелер және жүктеу облысы (swap) үшін, екіншісі – деректер үшін, үшіншісі – транзакциялар журналы үшін және төртіншісі – индекстер үшін.

Техникалық тұрғыдан бірнеше функциональдық дискілер облысын бірдей (физикалық немесе логикалық) дискілерге орналастыруға болады. Себебі, практикалық тұрғыдан, ДҚБЖ әрқашан жұмыс жасайтын ДЭЕМ-де құрылады. Бірақ, әртекті ақпараттарды бірдей дискілерге бірлесе сақтау ішкі жүйелердің енгізу/шығаруын қайта жүктеуге апарады және, сәйкесінше, өнімділігі азаяды.

Түсініктемелер берейік. Деректер қорын жаңарту қосымшалармен орындалсын. Себебі: жазба журналда негізгі деректерді жаңарту синхронды орындалуы керек.

Резервтік көшірме

Резервтік көшірменің негізгі тағайындалуы деректер қорының жойылуын болдырмау болып табылады. Резервтік көшірме деректер қорының резервті көшірмелерін құрудан тұрады. Ескере кетейік, тек қана өзіндік деректердің ғана емес, сонымен қатар, ДҚБЖ қызметтік ақпараттарының (деректер сөздігі, транзакциялар журналы және т.б.) да резервті көшірмелерін алу керек.

ДҚ резервті көшірмелерін құрған кезде әдетте магниттік ленталар (МЛ) немесе дискілер қолданылады. Резервті көшірме бастапқы ДҚ нақты көшірмесі немесе архивтік көшірмесі болуы мүмкін. Тығыздау аппараттық немесе бағдарламалық жүзеге асуы мүмкін. Аппараттық тығыздау осы операцияға кететін уақытша шығындар тұрғысынан қарағанда тиімді, бірақ аппараттық бөлімдер құнын арттырады.

Резервтік көшірме ДҚ-мен жұмыс жасау (online режимінде) кезінде немесе басқа уақытта жүзеге асуы мүмкін. Көшірме оператор қалауы бойынша немесе автоматты түрде белгілі бір уақытта сәйкес қосымшаларды жүктеу жолымен құрылуы мүмкін.

Резервтік көшірме құру кезінде администратор алдында келесі мәселелер туындайды:

- резервтік көшірме құру үшін қандай құрылғыны таңдау керек;
- резервтік көшірме құру қашан және қаншалықты жиі орындалады.

1. Резервті көшірмелерді сақтау үшін құрылғы (магниттік лента немесе диск) түрін анықтау кезінде ең алдымен, көшірме алу процедурасына бөлінген уақыт есепке алынады. Үлкен ДҚ үшін және баяу жұмыс жасайтын құрылғылар үшін көшірме құруға кететін уақыт өте көп болуы мүмкін. Ендеше, мысалы, көлемі 20 Гбайт, 500 Кбайт/с жылдамдықпен жұмыс жасайтын деректер қоры үшін 12 сағатқа жуық уақыт жұмсалады.

Қазіргі уақытта шығарылатын жинақтауыштар деректерді шамамен сағатына 1,25 Гбайт жылдамдығымен көшіре алады. Қайсібір магниттік лентадағы жинақтауыштар 8 мм лентада 3 Гбайт/сағат дейінгі жылдамдықпен қамтамасыз ете алады. Кейбір құрылғылар көлемі 25 Гбайт дейін болса да, бірақ неғұрлым типтік құрылғылардың сыйымдылығы 8-10 Гбайт болып табылады.

Fujitsu Model 2480 құрылғысы сағатына 10 Гбайт жылдамдықты қамтамасыз ете алады, бірақ жинақтауыш көлемі шектеулі (200-400 Мбайт). Неғұрлым заманауи құрылғылар, мысалы Metrum фирмасы өнімдері, үлкен жылдамдыққа (15 Мбайт/с) және көлемге (14 Гбайт) ие.

Резервтік көшірме және қалпына келтіру неғұрлым жылдам жүзеге асады. Бұл функцияларды аздаған шығынмен орындайтын бағдарламалық қамтамасыз ету мысалы Online:DiskSuite болып табылады.

2. Егер ДҚ-мен жұмыс үздіксіз жүзеге асса немесе жүйелердің жұмыс жасау ұзақтығы көшірме құру үшін жеткілікті болатын уақытша терезелері бар болса, онда резервті көшірме online режимінде құрылады. Басқа жағдайларда көшірме құру жұмыс күні немесе аптаның соңында орындалады.

Online режимінде көшірмесін алу кезінде ДҚ кестесімен келісімді қамтамасыз ету мәселесі пайда болады, себебі көшірмелерді алу кезінде деректермен жұмыс параллель орындалады. Бұл мәселені шешу үшін ДҚБЖ деректер қоры туралы барлық жаңартуға дейінгі ақпарат толық көшірмесін алғанға дейін кестеде жаңартуларсыз "жинақталады" немесе деректер қоры пайдаланушыларға тек қана оқу үшін қатынас құруға рұқсат етеді. Егер ДҚБЖ бұл міндетті атқармаса, онда оны ДҚ администраторы орындайды.

Орындалып жатқан резервтік көшірме құру процесін тексеріп отыру керек. Сонымен қатар, резервті көшірме алуды құжаттандыру туралы ұмытпаған жөн.

8.2. Ақпаратты қорғау

Деректер қорындағы ақпараттарды қорғау мәселесін есептеуіш жүйелерді (ЕЖ) қорғау мәселесімен бірге қарастырған жөн. Шындығында, ДҚБЖ жұмыс жасау ортасы – деректерді басқарудың негізгі құралы есептеуіш жүйелер ортасы болып табылады.

Ақпараттарды қорғау жүйелерін тұрғызу принциптерін білу және есептеуіш жүйелердің әртүрлі компоненттері (операциялық жүйелер, бағдарламалармен қызмет көрсету, ДҚБЖ маманданған дестелермен және дербес құрылғылармен қорғау) ұсынатын мүмкіндіктер құпия ақпараттарды қорғауды сауатты түрде ұйымдастыруға мүмкіндік береді.

Негізгі ұғымдар

Компьютердің басқа бөлімдерімен байланысқан немесе жергілікті не ауқымды желіге қосылған дербес компьютерде және машинада ақпаратты қорғау талап етіледі. Себебі желілік режим компьютерлерді пайдаланудың неғұрлым жалпы жағдайы болып табылады. Ақпараттарды қорғау мәселелерін қарастырған кезде, жалпы жағдайда, компьютерлердің арасында желілік байланыс бар деп есептейік.

Есептеу жүйесіндегі ***ақпараттар*** (ақпараттық қамтамасыз ету) дегеніміз жинақталатын, сақталатын және өңделетін деректердің кез-келген түрі. Жеке жағдайда. ақпараттар деректерді өңдеуді жүзеге асыратын бағдарламалар (жүйелік және қолданбалы) жиынтығы болып табылады.

Есептеу жүйесінде ақпараттарды қорғауды қарастыру кезінде ақпараттық-бағдарламалық қамтамасыз етуді қорғау туралы айтамыз.

Ақпараттық-бағдарламалық қамтамасыз етуді ***қорғау мақсаты*** сақталатын және өңделетін ақпараттардың, сонымен қатар қолданылатын бағдарламалық құралдардың қауіпсіздігін қамтамасыз ету болып табылады.

Жүйелерді қорғау ЕЖ-ні бастапқы өңдеу, оны ақпараттық-бағдарламалық қамтамасыз ету және орындаудың өміршеңдік кезеңінің барлық кезеңдерінде орындалуы керек. Іс жүзінде, ол ЕЖ және оның ақпараттық-бағдарламалық қамтамасыз етуі құрылған кезде басталады. Жүйелерді қорғау көпдеңгейлі болуы, олардың жұмыс

жасаудың жаңа шарттарына бейімделуі, құралдар, әдістер және іс-шаралардың ұйымдасқан жиынтығын қамтуы қажет.

ЕЖ қауіптердің *негізгі түрлері*:

1. ЕЖ ресурстарын рұқсатсыз пайдалану:

- Деректерді қолдану (көшірмесін алу, түрлендіру, өшіру, баспаға шығару және т.б.);
- көшірмесін алу және бағдарламаларды түрлендіру;
- жүйелерге рұқсатсыз ену үшін бағдарламаларды зерттеу.

2. ЕЖ ресурстарын дұрыс емес пайдалану:

- қолданбалы бағдарламалардың негізгі жадының басқа бөліктеріне кездейсоқ қатынас құруы;
- дискілік жадының жүйелік облысына кездейсоқ қатынас құру;
- деректер қорындағы жаңылыс өзгерістер (дұрыс емес деректерді енгізу, деректер тұтастығын бұзу);
- пайдаланушылар мен қызметкерлердің қате әрекеттері.

3. Бағдарламалық және аппараттық құралдарда қателіктердің болуы.
4. Байланыстар және жөнелту жүйелеріндегі желілерде деректердің жоғалуы.
5. Электромагниттік сәуле шығару кезіндегі рұқсатсыз тіркелулер.
6. Ақпараттарды және құжаттарды жинақтауыштардың ұрлануы.
7. ЕЖ компоненттері құрамына рұқсат етілмеген өзгерістер енгізу, ақпараттарды жөнелту және т.б.

Қорғауды бұзудың салдарлары:

- Құпия мәліметтерді алу;
- Өнімділіктің азаюы немесе жүйелердің істен шығуы;
- операциялық жүйелерді қатқыл дисктен жүктеу мүмкіндігінің болмау;
- материальық шығын.

Қорғау мақсаты ЕЖ-гі ақпараттардың қауіпсіздігін қамтамасыз ету болып табылады, оның бұзылуы мүмкін (кездейсоқ немесе әдейі).

Жоғарыда айтылғандардың негізінде қорғаудың негізгі үш мәселесін бөліп көрсетуге болады:

- ақпаратты ұрлаудан қорғау;
- ақпаратты жоғалудан қорғау;
- ЕЖ тоқтап қалу мен істен шығудан қорғау.

Ақпаратты ұрлаудан қорғау дегеніміз ақпараттарды сақтау құрылғылары және жинақтауыштардың физикалық ұрлануын, рұқсат етілмеген ақпараттарды алуды (көшірме алу, қарап шығу, қиып алу және т.б.) және рұқсат етілмеген бағдарламаларды таратуды болдырмау.

Ақпаратты жоғалудан қорғау дегеніміз ақпараттардың тұтастығын және дұрыстығын сүйемелдеу, ол ақпараттардың тұтастығын физикалық, логикалық және семантикалық қамтамасыз етуді білдіреді. ЕЖ аппараттық-бағдарламалық қамтамасыз етілуін *тоқтап қалу мен істен шығудан қорғау* жүйелермен қалыпты жұмыс жасаудың қажетті шарттарының бірі болып табылады. Егер есептеу жүйесі сенімді емес болса, онда ақпарат жиі бұрмаланады және кей жағдайда жоғалады. Тоқтап қалу мен істен шығудан қорғау жүйелерінде негізгі күш жүйелік аппараттық-бағдарламалық компоненттерге түседі: процессор, негізгі жады, сыртқы есте сақтау құрылғысы, енгізу-шығару құрылғысы және басқа құрылғылар, сонымен қатар операциялық жүйелер бағдарламалары. Жеткіліксіз сенімді жүйелік құралдарда тоқтап қалу мен істен шығудан қорғау шаралары қолданбалы бағдарламаларда қарастырылуы керек.

БҚЕ сенімділік деңгейі өңдеу, ұйымдастыру мен оны сүйемелдеу процесін автоматтандырудың сапасы және деңгейімен анықталады. Өйткені, бағдарламалардың 100% сенімділігіне жету іс жүзінде мүмкін емес, бағдарламалар және деректердің жұмыс жасау қабілетін жылдам қалпына келтіру құралдарын қарастыру керек.

ЕЖ-де ақпараттарды кешенді қорғауды ұйымдастыру үшін жалпы жағдайда ***4 қорғау деңгейі*** қарастырылуы мүмкін.

1. ЕЖ барлық орналасу аумағын қамтитын сыртқы деңгей.
2. ЕЖ құрылғылары және олардың арасындағы байланыс желілері орналасқан жекелеген ғимараттар немесе бөлмелер деңгейі.
3. ЕЖ компоненттері және ақпараттарды сыртқы жинақтауыштар деңгейі.
4. ақпараттарды сақтау, өңдеу және жөнелту технологиялық процестерінің деңгейі.

Алдыңғы үш деңгейде көбінесе қатынас құруды бөгеу жолымен физикалық бөгет, сигнализация жүйелері, рұқсатнамалық тәртіпті ұйымдастыру және т.б. қамтамасыз етіледі. Соңғы деңгей физикалық қатынас құру орындалмаған кезде, ақпараттарды логикалық қорғау үшін қолданылады.

Қорғау әдістері мен құралдары

Қолданыстағы қорғау әдістерін төрт топқа бөлуге болады:

- физикалық;
- аппараттық;
- бағдарламалық;
- ұйымдастырушылық.

Физикалық қорғау көбінесе жоғары деңгейде қорғауда қолданылады және басқа тұлғалардың ЕЖ ғимаратына енуін физикалық бөгеу жолымен жүзеге асады. Физикалық қорғау үшін келесі құралдар қолданылады:

- ультрадыбыстық және инфрақызыл жүйелер: қозғалыстағы объектілер табылғанда олардың өлшемін, жылдамдығын және бағытын анықтау, орын ауыстыру;
- жарық сәулелерінің қиылысуын сезінетін лазерлік және оптикалық жүйелер;
- қорғаныс объектілерін бақылаудың телевизиялық жүйелері;
- кішігірім объектілерді сыммен қоршап қоюға негізделген кабельдік жүйелер;
- рұқсат етілмеген енулерден, сонымен қатар бақылау мен тыңдаудан есіктер мен терезелерді қорғау жүйелері;
- есіктер мен қақпаларға механикалық және электрондық құлыптар салу;
- сәуле шығаруды бейтараптандыру жүйелері.

Аппараттық қорғау ЭЕМ құрамындағы мамандандырылған құрылғылар көмегімен жүзеге асады. Негізгі аппараттық құралдармен қорғау құралдары орналасады: процессорда және негізгі жадыда, енгізу-шығару құрылғысында, байланыстар каналы бойынша деректерді жөнелту жүйелерінде, қуат көзімен қамтамасыз ету жүйелерінде, сыртқы жады құрылғыларында және т.б.

Бағдарламалық қорғау әртүрлі бағдарламалар көмегімен жүзеге асады: операциялық жүйелер, қызмет көрсету бағдарламалары, антивирустық дестелер, аспаптық жүйелер (ДҚБЖ, электрондық кесте, мәтіндік процессорлар, бағдарламалау жүйелері және т.б.), мамандандырылған қорғау бағдарламалары және дайын қолданбалы бағдарламалар.

Ұйымдастырушылық қорғау ақпараттарды қорғауға бағытталған ұйымдасқан-техникалық іс-шаралар жиынтығы арқылы, ақпараттарды қорғау мәселелері бойынша заңдық актілер құру және қабылдау арқылы жүзеге асады. Неғұрлым жан-жақты және жиі қолданылатын бағдарламалық-аппараттық қорғау әдістеріне тоқталайық.

Бағдарламалық-аппараттық қорғау әдістері

Бағдарламалық-аппараттық құралдар көмегімен ЕЖ-де ақпараттық-бағдарламалық қамтамасыз етуді және бағдарламаларды қателіктерден (ұрланудан, жоғалудан,

құрылғының тоқтап қалуы немесе істен шығуынан) қорғаудың негізгі міндеттерін шешуге болады. Жүйелерде осы міндеттерді шешу келесі әдістермен жүзеге асады:

1) пайдаланушылар және бағдарламалар тарапынан ресурстарға рұқсат етілмеген қатынас құрулардан қорғау;
2) қатынас құру кезінде ресурстарды рұқсатсыз пайдаланудан қорғау;
3) ресурстарды дұрыс емес пайдаланудан қорғау;
4) құрылымдық, функциональдық және ақпараттық артықтық енгізу;
5) бағдарламаларды өңдеудің аппараттық құралдарының жоғары сапасы.

Аталған әдістерді және олардың жүзеге асуын неғұрлым жан-жақты қарастырайық.

1. Пайдаланушылар және бағдарламалар тарапынан ресурстарға рұқсат етілмеген қатынас құрулардан қорғау үшін, ең алдымен, сенімді *тіркеу жүйелері*, сонымен қатар шапшаң *сигнализация* керек. Сенімді тіркеу жүйелері мен сигнализацияның болмауы, сонымен қатар, айналып өту жолдары мен «тесіктердің» бар болуы жүйелерге заңсыз енудің себебі болып табылады. Тіркеп отыру үшін ЕЖ әдетте арнайы журнал немесе деректер қорын жүргізеді.

Заманауи жүйелерде пайдаланушылар тарапынан рұқсат етілмеген қатынас құрулардан қорғау көбінесе негізгі екі әдіспен жүзеге асады: құпия сөзбен қорғау, сонымен қатар сәйкестендіру және аутентификация жолымен қорғау.

Құпия сөзбен қорғау, егер құпия сөз шифрленбесе, онда қарапайым қорғау болып табылады. Оның негізгі кемшілігі: барлық пайдаланушылардың бірдей құпия сөз қолдануы. Тағы бір кемшілігі - құпия сөзді есте сақтау керек. Егер ол қарапайым және қысқа болса, онда оны жылдам тауып алуға болады, егер күрделі болса – оны әдетте бір жерге жазып қояды. Сондықтан, оның басқалар үшін қолжетімді болуы әбден мүмкін.

Пайдаланушыларды сәйкестендіру құпия сөз көмегімен орындалуы мүмкін. Пайдаланушыларды *аутентификациялау* немесе тексеру үшін келесі әдістер жиі қолданылады:

- құпия сөзді сұрау;
- қандай да бір аса маңызды жеке ақпараттарды сұрау;
- әдеттегі кілттің (электронды кілт) аналогы ретінде қолданылатын физикалық объектінің болуын тексеру;
- микропроцессорлық карточкаларды қолдану;
- танып айырудың белсенді құралдары;
- биометрикалық құралдар.

Рұқсат етілмеген қатынас құрушы бағдарламалардың бірі компьютерлік вирустар болады. Компьютерлік вирустар саны өте көп. Тіпті жаңа инженерлік пән – компьютерлік вирусология пайда болды. Компьютерлік вирустар әсері әртүрлі: олар ЭЕМ жұмысын баяулатудан бастап, есептеуіш жүйелер немесе желі жұмысын күйретуге дейін апарады. Осыдан *компьютерлік вирустардан қорғау* қажеттілігі пайда болады.

2. Қатынас құру кезінде ресурстарды рұқсатсыз пайдаланудан қорғау сұраныстарды орындау кезінде *тіркеу құралдарының* болуын талап етеді және заңсыз пайдалану кезінде *сигнализация* іске қосылады. Ескере кетейік, мұнда ресурстарды маңыздылық тұрғысынан қарағандағы қорғау туралы сөз болып отыр. Егер барлық сұраныстардың барлық оқиғаларын тіркеп отырсақ, онда басқа жұмысқа процессордың уақыты жетпейді.

Ақпараттық-бағдарламалық ресурстарды рұқсатсыз пайдаланудан қорғау үшін келесі қорғау нұсқалары қолданылады: *көшірме алудан, деректерді көруден, түрлендірулерден және өшіруден қорғау*. Оларды қолдануға мысалдар келтірейік.

Бағдарламаларды рұқсат етілмеген *көшірме алудан* қорғау үшін орындалатын кодта оларды құрылғыға бекітеді. Сонда көшірме бағдарламалары басқа компьютерде жұмыс жасамайды.

Деректерді *түрлендіруларден* қорғау үшін файлдың қайсібір сипаттамаларын эталонмен салыстыруға болады. Онда, егер біреу файл құрамын өзгертсе, оның бақылау қосындысы өзгереді де, ол бірден көзге түседі. Бақылау қосындысысын тексеру құралдарын бағдарламалауға немесе файлдардың (бағдарламалар және деректердің) түрлендірулерін бақылаудың бағдарламалық жүйелеріне қосуға болады.

Бағдарламалар немесе деректерді *өшіруден қорғауды* есептеуіш жүйелерде файлдарды өшіру операцияларын рұқсатсыз орындауға тыйым салу арқылы жүзеге асыруға балады. Өкінішке орай, кеңінен таралған MS DOS және MS Windows операциялық жүйелерінде осы міндетті атқаратын стандартты тиімді құралдар жоқ.

Деректерді *көруден қорғаудың* жеткілікті қуатты құралдары оларды *шифрлеу* болып табылады. Ақпараттарды бастапқы қалпына келтіру шифрлеу кілтін білуді талап етеді.

Шифрлеу ақпараттарды файлдарға немесе деректер қорына сақтау кезінде, байланыстар желісі бойынша ақпараттарды жөнелту кезінде ақпараттарды қорғау үшін өте қажет.

Деректерді шифрлеу ақпараттарды жөнелту режимінде (On-Line) және дербес жұмыс жасау режимінде (Off-Line) жүзеге асады. Алғашқы әдіс көбінесе ақпараттарды қабылдау-жөнелту жүйелерінде, ал екіншісі – сақталатын ақпараттарды құпиялау үшін қолданылады.

Заманауи жүйелерде көбінесе қорғаудың екі алгоритмі қолданылады: DES және RSA. Оларды қысқаша қарастырайық.

IBM фирмасымен 70-жылдардың басында құрылған деректерді шифрлеу стандарты – Data Encryption Standard (DES) Америка банкирлері ассоциациясымен ұсынылған және сандық шифрлеудің үкіметтік стандарты болып табылады.

DES алгоритмінде ұзындығы 56 биттік кілт және 8 бит жұпқа тексеру қолданылады. Ол шифрлеуге аз шығынмен жоғары деңгейлі қорғауды қамтамасыз етеді.

DES алгоритмі *симметриялық* болып табылады, яғни шифрлеу және бастапқы қалпына келтіру үшін бір ғана кілтті қолданады.

RSA алгоритмін Ривестт, Шамир және Альдеман 1976 жылы ұсынған. Алгоритм неғұрлым жетілдірілген болып табылады және Ұлттық стандарттар бюросының стандарты ретінде қабылданған.

RSA алгоритмінде шифрлеу және дешифрлеу үшін әртүрлі кілттер қолданылады, яғни ол *асимметриялық* болып табылады. Себебі: шифрлеу кілті оны бастапқы қалпына келтіру үшін жарамсыз. Оны желі бойынша батыл жөнелтуге болады, сондықтан шифрлеу кілтін ашық кілт деп те атайды.

3. Ресурстарды дұрыс емес пайдаланудан қорғау әдетте ОЖ бағдарламаларымен орындалады. Дұрыс емес пайдаланудан қорғау функциялары келесі әрекеттерді қарастырады: *әртүрлі бағдарламалар үшін ерекшеленгендерді жедел жадыда бір бірінен оқшаулау, сыртқы жадының жүйелік облысын қорғау және командаларды бақылау.*

Қолданбалы бағдарламалық қамтамасыз ету деректердің *тұтастығын және қайшылықсыздығын* қамтамасыз ету керек.

4. *Құрылымдық артықтық* аппараттық компоненттерді әртүрлі деңгейде резервке алуды білдіреді: ЭЕМ; жекелеген құрылғылар және сызба құрылғылары. Резервке алу кезінде, ең алдымен, тұрақты және үздіксіз қуат көзімен қамтамасыз ету керек.

Функциональдық резервке алу басқару, сақтау және ақпаратты өңдеу функциялары жүйелердің бірнеше элементтерімен жүзеге асқан кезде есептеу процесін

ұйымдастыруды білдіреді. Функциональдық элемент істен шыққан кезде оны басқа элемент алмастырады.

Ақпараттық резервке алу ақпараттардың жоғалуын толықтай болдырмау үшін қолданылады және неғұрлым құнды ақпараттардың бір реттік немесе периодты түрде көшірмесін алу және архивтеуді жүзеге асырады. Оған ең алдымен, пайдаланушының қолданбалы бағдарламаларын, сонымен қатар әртүрлі деректерді жатқызуға болады: құжаттар, ДҚ, файлдар, ОЖ негізгі бағдарламалары, ақпараттық-бағдарламалық қамтамасыз ету (ДҚБЖ, мәтіндік, кестелік және графикалық процессорлар және т.с.с.).

ДҚ қорғау құралдары

Әртүрлі ДҚБЖ-да ДҚ қорғау құралдары бір-бірінен өзгеше. Borland және Microsoft фирмаларының заманауи ДҚБЖ талдау негізінде ДҚ қорғау құралдарын шартты түрде екі топқа бөлуге болады: негізгі және қосымша.

Негізгі ***ақпараттарды қорғау құралдарына*** келесі құралдар жатады:

- құпия сөзбен қорғау;
- деректер және бағдарламаларды шифрлеу;
- ДҚ объектілеріне қатынас құруды тағайындау;
- ДҚ кестесінің өрісі және жазбаларын қорғау.

Құпия сөзбен қорғау ДҚ рұқсат етілмеген қатынас құрулардан қорғаудың қарапайым және тиімді әдісі. Құпия сөз пайдаланушылармен немесе ДҚ администраторымен орнатылады. Әдетте құпия сөз ДҚБЖ қайсыбір жүйелік файлында жасырын түрде сақталады. Сондықтан, құпия сөзді табу және анықтау мүмкін емес. Құпия сөзді енгізгеннен кейін, ДҚБЖ пайдаланушыға қорғалған ДҚ-мен жұмыс жасау кезінде барлық мүмкіндіктер беріледі.

Деректерді шифрлеу (деректер қоры немесе жекелеген кесте) «осы ДҚБЖ-ның ДҚ форматын білетін» басқа бағдарламалардың деректерді оқымауы үшін қолданады. Мұндай шифрлеу (Microsoft Access бағдарламасында пайдаланатын) аса сенімді емес, себебі кез-келген "өзіндік" ДҚБЖ көмегімен ашуға болады. Егер шифрлеу және шифрден алу құпия сөз талап етсе, онда қорғаныс сенімдірек болар еді.

Бағдарламалардың бастапқы мәтіндерін шифрлеу рұқсат етілмеген пайдаланушыдан сәйкес алгоритмдердің сипаттамасын жасыруға мүмкіндік береді.

Кестелерге қатынас құру бойынша, жалпы жағдайда, мынадай қатынас құру құқықтары бар:

- Деректерді көру (оқу);
- Деректерге өзгерістер енгізу (редактірлеу);
- Жаңа жазбалар кірістіру немесе қосу;
- Деректерді кірістіру немесе қосу және өшіру;
- Барлық операциялар, сонымен қатар, кестелердегі өзгерістер құрылымы.

Реляциялық ДҚБЖ-да ішінара жазба арнайы қорғалмайды, бірақ, іс жүзінде бұған мысал келтіруге болады. Кесте өрісіндегі деректерге қатынас құру деңгейлері:

- қатынас құруға толықтай тыйым салу;
- тек қана оқу үшін;
- барлық операцияларға рұқсат беру (көру, жаңа мәндерді енгізу өшіру және өзгерістер енгізу).

Формаларға қатынасы бойынша екі негізгі операциялар қарастырылады: жұмыс жасау үшін шақыру және құру (конструкторды шақыру). Экрандық формаларда элементтер ішінара қорғалған болуы мүмкін.

Есеп беру көбінесе экрандық формаларға ұқсас. Біріншіден, олар кестелер деректеріне өзгерістер енгізбейді, ал екіншіден, олардың негізгі тағайындалуы – ақпараттарды баспаға шығару.

ДҚ қорғаудың қосымша құралдарын қорғау құралдарына жатқызуға болмайды, бірақ олар деректердің қауіпсіздігіне тікелей әсер етеді. Олар:

- сәйкес типті деректердің мәндерін кіріктірілген бақылау құралдары;
- енгізілген деректердің дұрыстығын арттыру;
- кестедегі байланыстың тұтастығын қамтамасыз ету;
- объектіні бірлесе пайдалануды ұйымдастыру.

8.3. Мультимедиа-деректермен жұмыс жасау

Мультимедиалық ақпараттарды өңдеу бойынша ДЭЕМ мүмкіншіліктерінің (графикалық, аудио- және видео-) дамуы осы ақпараттарды басқаруды маманданған бағдарламалық қамтамасыз етуді құжет етеді. Әртүрлі типті ақпаратты өңдеу және сақтау қажеттілігін сәйкес ДҚБЖ талап етті. Ендеше, мысалы, геоақпараттық жүйелер пайда болды. Бағдарлама құрушы реляциялық жүйелерде, жаңа қажеттіліктерді ескере отырып, ДҚБЖ-да жаңа деректер типін енгізе бастады.

Осылайша, бір жағынан, мамандандырылған өңдеу және мультимедиалық ақпараттарды сақтау құралдары пайда болды, ал басқа жағынан, – реляциялық жүйелер ақпараттық-бағдарламалық қамтамасыз етумен толықтырылды. Бағдарламалық құралдарды таңдау кезінде келесі факторлар ескеріледі: шығарылатын есептің мамандануы, бағдарлама құрушының құрылатын қосымшаның архитектурасына көзқарасы және қайсібір басқа жағдайлар.

Мультимедиа сипаттамасы

Мультимедиа (multimedia) сөзі 90-жылдардан бастап кеңінен қолданыла бастады. Оның ағылшын тілінен нақты аудармасы жоқ, бірақ ол «көпорталық» немесе «орталар жиыны» деген мағынаны береді. Орта ұғымы әртүрлі типті деректер түрінде түсіндіріледі: дыбыстық, видео-, графикалық, мәтіндік және т.б.

Кең мағынада, мультимедиа өндірістік технологиялардың жиынтығын білдіреді және онда қолданылған әртүрлі аппараттық және бағдарламалық құралдар компьютердің осы типті ақпараттарымен жұмысын сүйемелдеуге мүмкіндік береді.

Мультимедианы пайдалануға мысалдар келтірейік:

- электрондық оқулықтар, энциклопедиялар және каталогтар құру;
- бағдарламалық қамтамасыз ету және техникалық құжаттандырудың үлкен көлемін компакт-дискілерге орналастыру;
- жергілікті желіде телеконференциялар өткізу.

Жергілікті желіде телеконференциялар өткізу процесінде әрбір қатысушы видеокамера және байланыстардың шапшаң желілері көмегімен нақты уақыт аралығында басқа адамдармен еркін сөйлеседі.

Мультимедианы аппараттық қамтамасыз ету

Мультимедиа-ақпараттармен жұмыс жасау үшін компьютерге қойылатын талаптарды қарастырайық. Мұны бірмәнді анықтау қиын, себебі компьютерлердің сипаттамалары және оларға қойылатын талаптар жиі өзгереді.

Аталған мәселені шешу тәсілдерінің бірі стандарттау болып табылады. Microsoft фирмасы басқа фирмалармен бірлесе отырып, мультимедианы дербес ЭЕМ-ге енгізу негізі ретінде Red Book стандартын құрды. Мультимедианың бірнеше стандарттары бар, себебі: ДЭЕМ (микропроцессорлар: 8086-дан 80486-ға дейін, Pentium, Pentium II, P6 және т.б.; жылдамдығы неғұрлым жоғары CD-ROM құрылғылары; қатқал магниттік дискідегі жинақтауыштар; 640 Кбайттан до 64 Мбайтқа дейінгі жедел жады және т.б.) мүмкіндіктері кеңеюде және ақпараттарды өңдеу және сақтау қызметін қолданушылардың талабы артуда.

Мультимедиа стандарты талаптарын қанағаттандыратын және мультимедиа үшін маркетинг кеңесімен мақұлданған компьютер MPC логотипін алады. Алғашқы MPC1 стандарты 1990 жылы, екінші стандарт MPC2 стандарты 1993 жылы, үшінші MPC3 стандарты 1994 жылы қабылданды.

Мультимедианың негізгі компоненттерінен (CD-ROM, дыбыстық адаптер, микрофон) басқа MPC-компьютердің құрамына адаптерлер немесе аналогты-цифрлық түрлендіргіштер енуі мүмкін. Олар компьютерді видеокамераларға, видеомагнитофондарға, телевизорларға, синтезаторларға және басқа құрылғыларға қосу үшін керек. Видео-бейнені басқару технологияларына жататындар: MPEG (Motion Picture Experts Group), Indeo, Cinepak және TrueMotion-S.

Мультимедиа-ақпараттармен жұмыс жасау үшін MPC компьютердің ақпараттық-бағдарламалық қамтамасыз етілуі міндетті емес. Қағидаға сәйкес, кез-келген IBM-үйлесімді ДК мультимедиа- немесе «мультимедиаға жуық» компьютерге айналуы оңай. Бұл жағдайда міндетті элементтер CD-ROM, дыбыстық карта және микрофон болады. Аккустикалық жүйелер ретінде кәдімгі тұрмыстық магнитофондар немесе аудиоорталықтарды қолдануға болады.

Мультимедианы бағдарламалық қамтамасыз ету

MS Windows операциялық жүйелерінде мультимедиа-ақпараттармен жұмыс жасау үшін бағдарламалық қамтамасыз етуді шартты түрде үш топтарға жіктеуге болады:

- базалық;
- қолданбалы;
- қосымшаны өңдеу құралдары.

Базалық бағдарламалық қамтамасыз ету мультимедианың аппараттық бөлімін ең аз басқарушылық баптау болып табылады. Ол мультимедиа-құрылғылар драйверлері және басқарудың резидентті бағдарламаларын іске қосады.

CD-ROM құрылғылары және дыбыстық карталармен бірге өндіруші фирманың драйверлері беріледі. Неғұрлым кең таралған құрылғы әдетте Windows құрамына кіретін драйверлерді сүйемелдейді.

Қосымшаны және дыбыстық картаны баптау үшін электрондық пультпен басқарылатын Windows-қосымшасы қолданылады.

Қолданбалы бағдарламалық қамтамасыз етуді дыбысты және бейнені басқарудың негізгі функцияларын жүзеге асыратын Windows-қосымшасы құрайды.

Бұл топтың *қарапайым бағдарламалары* дыбыспен және анимациямен күрделі емес операцияларды орындайды.

Қарапайым қолданбалы бағдарламалар мысалдары Windows операциялық жүйесінің стандартты қосымшалары: Басқару панеліндегі Дыбыс ішкі бағдарламасы, Sound Recorder қосымшалары және Media Player қосымшалары.

Анимацияларды өңдеудің күрделі емес бағдарламаларының мысалдары Animator және Animator Pro бағдарламалары болады.

Анимацияны толықтай қолмен құру оңай емес, себебі бейнелердің қатар кадрлері ұқсас болуы және оларды экраннан көрсету бүтін бір әрекет ретінде жүзеге асуы қажет. Барлық бейнелер тізбегін изображений сәйкестендіру үшін Animator Pro бағдарламаларында арнайы құралдар бар.

Неғұрлым *қуатты қолданбалы мультимедиа-бағдарламалары* дыбыстық ақпараттармен, сонымен қатар дыбыстық бейне-ақпараттармен неғұрлым күрделі операцияларды орындайды. Бағдарламалардың бұл тобына Voyetra Technologies фирмасының Sound Galaxy адаптерімен бірге ұсынылатын WinDAT қосымшаларын жатқызуға болады.

Видеофильмдерді жазуға, сақтауға, редактрлеуге және іске қосуға мүмкіндік беретін қосымша мысалы - Microsoft Video for Windows жүйелері. Тағы бір мысалы IPI

фирмасының Voice Toolkit қосымшалары, бұл бағдарлама энергетикалық спектрлерді үшөлшемді түрде бейнелеуге мүмкіндік береді. Сонымен қатар, бейнелеудің кез-келген бөлігін үлкейтуге және кеңістікте бұруға болады.

Заманауи Windows-қосымшаларда әдетте сыртқы мультимедиа-ақпараттарды өңдеудің өзіндік құралдары бар. Мұндай қасиетке Microsoft Write бағдарламасынан басқа, Microsoft Office дестесі құрамына кіретін: Excel, Word, Access және басқа бағдарламалар ие. ***Қосымшаны өңдеу құралдары*** ретінде арнайы қосымша құру құрал-жабдықтары немесе мамандарған кеңінен таралған қосымшалар қолданылады. Қосымшаны өңдеу құралдары жоғарыда сипатталған қолданбалы бағдарламалық қамтамасыз етумен бірге қолданылады. Мультимедиа технологияларын қолданатын Windows операциялық жүйесінің заманауи қосымшаны өңдеу құралдарын үш деңгейге бөлуге болады.

Бірінші деңгей (ең жоғары) қарапайым пайдаланушыларға арналған, онда қосымша бағдарламалаусыз құрылады. Бұл деңгейге Windows қосымшасы жатады. Ондай құралдар жылдам жарнама құру, күрделі емес оқыту жүйелерін құру немесе ДҚ-да дыбыстық және видеоақпараттарды қолдану үшін қажет.

Екінші деңгей MCI (Media Control Interface – басқару ортасы) интерфейсін қолдану арқылы бағдарламалауды жобалайды. Бұл интерфейс көмегімен Windows қосымшалары дыбыстық және видеоақпараттарды енгізуа-шығару драйверлерін басқаруы мүмкін. *Үшінші деңгей* қосымшаға жазылатын және дыбысталатын ақпараттар буферлеріне қатынас құруға мүмкіндік береді, олар басқа қосымша мүмкіндіктерді қолдану арқылы файлдардың ішкі құрылымымен жұмыс жасайды. Бұл деңгей алдыңғы екі деңгей қауқарсыз болған жағдайда пайдаланылады, көбінесе төменгі деңгей қолданылады.

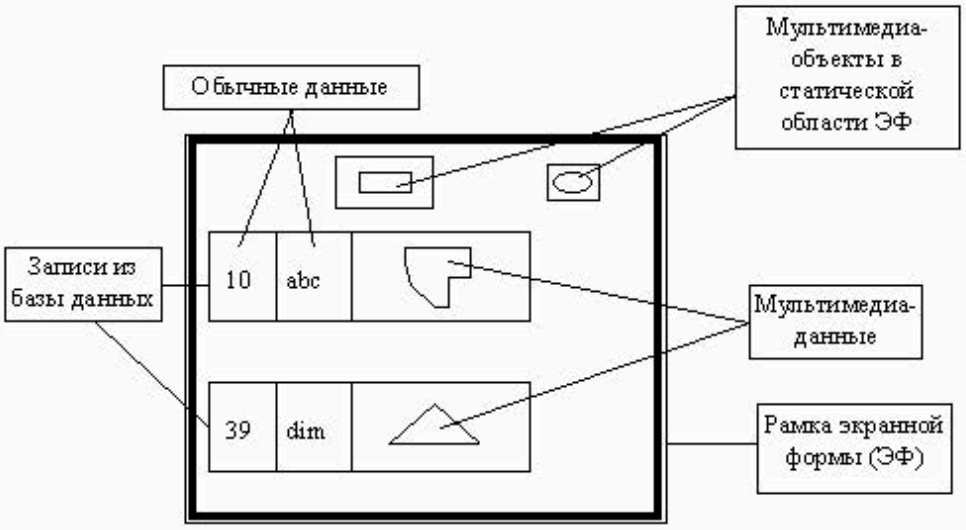
Екінші және *үшінші* деңгей функциялары әдетте бағдарламалау жүйелеріне қосылған арнайы бағдарламалар кітапханасы (динамикалық жүктелген) көмегімен жүзеге асады. Қажетті функциялар бағдарламалардан пайдаланушының кәдімгі ішкі бағдарламалары ретінде шақырылады. Мысалы: Windows 9x жүйесінде мультимедиа көмегімен Си бағдарламалау тілінде mmsystem.dll кітапханасымен жұмыс жасау (әдетте ол C:\WINDOWS\SYSTEM бумасында орналасады).

Реляциялық жүйелердің мультимедиа-мүмкіндіктері

Реляциялық жүйелердің мультимедиа-мүмкіндіктері туралы сөз қозғаған кезде олардың негізгі ролі – мультимедиа-деректерді сақтау екендігін ескере кетейік.

Реляциялық жүйелерде деректерді, сонымен қатар мультимедиа-деректерді сақтаудың негізгі орны кесте болып табылады. Мультимедиа-деректерді сақтауды қамтамасыз ету кестелері үшін құрылымында сәйкес өрістер болуы керек. Сонымен қатар, мультимедиа-деректер экрандық формалар мен есеп берулерде сақталуы мүмкін.

Аталған әдістердің негізгі айырмашылықтары: бірінші жағдайда мультимедиа-деректер деректер қорының әрбір жазбасымен байланысады, ал екінші жағдайда мультимедиа-деректер экрандық форманы немесе есеп беруді бір рет қамтиды. Ендеше, мультимедиа-деректер деректер қорының әрбір жазбасымен байланысты болған кезде өзгерістер енгізу ағымдағы жазбаны өзгертеді, ал мультимедиа-деректер экрандық формалар болған кезде – өзгеріссіз қалады. Мультимедиа-деректер экрандық формаларда және есеп берулерде олардың экранда бейнеленуі кезінде көрнекілігін арттыру үшін қолданылады (8.1-сурет).



8.1-сурет. Экрандық формадағы мультимедиа-деректер

Әртүрлі ДҚБЖ мультимедиа-деректерді сүйемелдеудің әртүрлі механизмдері қолданылады. Көбінесе оларды сақтау үшін BLOB-өрістер (Binary Large OBject – үлкен екілік объектілер) қолданылады. Себебі: мультимедиа-деректер әртүрлі типті (аудио-, видео-, графикалық және т.б. ақпарат) болуы мүмкін, ал әрбір типке әртүрлі форматтар сәйкес келеді (мысалы, графикалық ақпараттарды сақтау үшін келесі кеңейтпелері бар файлдар қолданылады: bmp, pcx, tif, gif, eps және т.б.). Байланыстарда неғұрлым кең таралған ақпараттық-бағдарламалық қамтамасыз етудің BLOB-өрісі OLE-өрістер болады.

Мысалы, MS Access бағдарламасы OLE-өрісін сүйемелдейді, ал Paradox жүйелерінде graphic және binary типті өрістер құруға болады.

Бақылау сұрақтары және тапсырмалар

1. Деректер қорын баптау және басқарудың негізгі мәселелерін атаңыз.
2. Файлдардың дискіде орналасу әдісін таңдау тәсілдемесін сипаттаңыз.
3. Диск жадысынан қажетті көлем қалай анықталады?
4. Ақпараттардың дискіге орналасуын сипаттаңыз.
5. Резервтік көшірме тағайындалуы және технологиясын сипаттаңыз.
6. ЕЖ-гі негізгі қауіп түрлерін атаңыз.
7. Жүйелер қорғанысының бұзылуынан туындайтын жағдайларды атап шығыңыз.
8. Негізгі қорғау мәселесін сипаттаңыз.
9. Ақпараттарды кешенді қорғау деңгейлерін түсіндіріңіз.
10. ЕЖ қорғау әдістерін сипаттаңыз.
11. Бағдарламалық-аппараттық құралдармен қорғауға сипаттама беріңіз.
12. Пайдаланушыларды сәйкестендіру және аутентификация түсінігінің мағынасы неде?
13. Аутентификацияның негізгі әдістерін сипаттаңыз.
14. DES және RSA шифрлеу алгоритмдеріне сипаттама беріңіз.
15. ДҚБЖ-да ақпараттарды қорғаудың негізгі құралдарын сипаттаңыз.
16. ДҚ-да ақпараттарды қорғаудың қосымша құралдарын атаңыз.
17. Мультимедианы бағдарламалық қамтамасыз етуді не құрайды?
18. Реляциялық жүйелердің мультимедиа-мүмкіндіктерін сипаттаңыз.

9. ДЕРЕКТЕР ҚОРЫН ҚОЛДАНУДЫҢ ҚОСЫМША СҰРАҚТАРЫ

Бөлімде аппараттық-бағдарламалық платформалар сипаттамасы, ДҚБЖ таңдау және аппараттық құралдар құрылымы, деректерді өңдеудің көппроцессорлық жүйелері, ДҚБЖ даму келешегі және стандарты, деректер қоры және ақпараттық жүйелер құру қарастырылады.

9.1. Бағдарламалық-аппараттық платформалар

Деректер қорын құру кезінде қолданылатын және қолданылған бағдарламалық-аппараттық платформалар олардың жұмыс жасау тиімділігіне елеулі ықпал етеді. Деректер қоры үшін аппараттық-бағдарламалық платформалар таңдау мәселелерін екіге бөлуге болады:

- ДҚБЖ таңдау;
- деректер қорының аппараттық өңдеу құралдарын таңдау.

Деректерді өңдеу тиімділігін арттырудың перспективті бағыты деректер қорын өңдеудің көппроцессорлық жүйелерін қолдану болып табылады.

ДҚБЖ таңдау

ДҚ администраторы, кәсіпорын жетекшісі және қарапайым пайдаланушылар алдында ДҚБЖ таңдау мәселесі көбінесе оны сатып алу алдында және жаңа аппараттық-бағдарламалық құралдарға көшу кезінде пайда болады.

ДҚБЖ таңдау әдістері. ДҚБЖ таңдаудың негізгі принципі - қойылған талаптарға сай бағдарламалық өнімді анықтау. Практикалық тұрғыдан бұл мәселені шешу оңай емес. Біріншіден, ДҚБЖ-ға көптеген талаптар қойылады және ол талаптар уақыт өте келе өзгеріп отырады – жүйелердегі өзгерістер жаңа мүмкіндіктерді талап етеді. Екіншіден, ДҚБЖ-ғы параметрлер саны өте көп, оларды салыстыру қиын. Сонымен қатар, ДҚБЖ туралы ақпарат әдетте жарнамалық сипатқа ие, ол дұрыс таңдау жасауға кедергі келтіреді.

ДҚБЖ сипаттамасын бағалау технологиясын және олардың қойылатын талаптарға сай болу деңгейін анықтауды қарастырайық. Бағдарламалық өнімдер әдетте келесі ақпараттарды қамтиды:

- бағдарлама құрушылар және сатушының жарнамалық мәліметтеріндегі ақпараттар;
- бағдарламаны пайдаланушылар, бағдарлама құрушылар және өніммен жұмыс жасау тәжірибесі бар администраторлар ақпараттары;
- талдау жасаушылар және сарапшылар ақпараттары.

Өнімді таңдау кезінде негізгі параметрлерге баса назар аудару керек, ал қалғандары – «шектен шықпаса» болғаны.

ДҚБЖ таңдау процедурасын үш кезеңмен жүзеге асыру тиімді. Алдымен, сапалық деңгейде, ұсынылған бағдарламалық өнімдерді жарамдылыққа бағалау. Содан соң, таңдап алынған жүйелердің техникалық сипаттамаларын неғұрлым бөлшектеп қарастыру. Ең соңында, соңғы шешімді қабылдау үшін қалғандарының өнімділігін бағалау.

Бағдарламалық өнімдердің негізгі *жарамдылық көрсеткіштеріне* жататындар:

1. Бағдарламалық өнім түрі.
2. Пайдаланушылар санаты.
3. Пайдаланудың қолайлылығы және қарапайымдылығы .
4. Деректерді ұсыну үлгісі.
5. Өңдеу құралдарының сапасы (қосымшаны, сұраныстарды, экрандық формаларды, есеп берулерді және т.б.).

6. Деректер қорын қорғау және дұрыстығын бақылау құралдарының сапасы.
7. Коммуникациялық құралдар сапасы.
8. Бағдарлама құрушы фирма.
9. Құны.

Қажетті өнімді іздеуді қажеттіліктер мен мүмкіндіктерді меңгеруден бастаған жөн. Дестенің не үшін қажет екендігін де анықтау керек: кәсіби бағдарлама құрушының қолданбалы жүйелерді өңдеуі үшін немесе қарапайым пайдаланушылардың белсенді режимде жұмыс жасауы үшін және т.б. Таңдаудың бірінші кезеңде анықтаушы параметрлер бағдарламалық өнім түрі және пайдаланушылар деңгейі болады.

Негізгі көрсеткіштерге көбінесе алдыңғы екі жағдай жатады. Қалған көрсеткіштер шығарылатын есептердің ерекшеліктерінен тәуелді. Аталған көрсеткіштерді қарастырайық.

Жарамдылық көрсеткіштері. ДҚБЖ түрі және оларды жіктеу 1.3-бөлімде келтірілген, ДҚБЖ жарамдылық көрсеткіштерінің қалғандарын қарастырайық.

Пайдаланушылар санаты. Бағдарламалық өнімді пайдаланушылар санаттары:

- кәсіби бағдарлама құрушылар – ДҚБЖ, ДҚ серверлері және басқа бағдарламаларды құрушылар;
- ДҚ администраторы;
- қосымша құратын маманданған пайдаланушылар;
- қарапайым (мамандандбаған) пайдаланушылар;
- аталған санаттардың әртүрлі комбинациялары.

Бағдарламалық өнімдерді таңдау кезінде неғұрлым кеңінен қолданылатын бағдарламаларға назар аудару керек. Көптеген кең таралған, толық қызмет атқаратын ДҚБЖ-да пайдаланушылар және администраторлар, бағдарлама құрушылар үшін арнайы құралдар бар. Ендеше, Microsoft Access ДҚБЖ Visual Basic бағдарламалау қосымшасын қолдануға мүмкіндік береді.

Пайдаланудың қолайлылығы және қарапайымдылығы. Қолайлылық және қарапайымдылық түсінігі уақыт өте келе өзгереді, сонымен қатар, қойылатын талаптар тұрғысынан қарағанда қиындатылады. Бағдарламаларды пайдаланудың қолайлылығы және қарапайымдылығы келесілерді сипаттайды:

- бағдарламалық өнімдерді орнатудың түсінікті процедуралары;
- қарапайым пайдаланушыға қолайлы және сәйкестендірілген интерфейс;
- қарапайым операциялар орындалуының қарапайымдылығы: ДҚ құру, навигациялар, деректерді түрлендірулер, сұраныстар, есеп берулер және басқаларды дайындау және орындалу;
- жұмыс жасау және оқу кезінде анықтама немесе көмек интеллектуальды ішкі жүйелерінің бар болуы.

Деректерді ұсыну үлгісі. Қазіргі уақытта неғұрлым кеңінен таралған, теориялық және практикалық тұрғыдан тәжірибеден өткен деректердің реляциялық үлгісі болып табылады (2-бөлім). Объектілі бейімделу үлгілері перспективті болады, себебі олар зерттеу облысы семантикасын бейнелеудің үлкен мүмкіндіктеріне ие. Сондықтан, көп жағдайларда реляциялық және объектілі-бағытталған деректер үлгісін таңдайды. Кейде деректерді ұсынудың басқа үлгілерін пайдалану қажеттілігі туындауы мүмкін.

Өңдеу құралдарының сапасы. Өңдеу құралдарының сапасын бағалау кезінде келесілер ескеріледі: пайдаланушы интерфейсін құру мүмкіндіктері; бағдарламалар құру (кода автоматты түрде іздестіру, орындау, тұтастықты қамтамасыз ету деректердің деңгейінде процессора ДҚ, ал не көмегімен команд тілі) тілдері; автоматтандыру өңдеу әртүрлі объектов: экрандық формалар, есеп беруов, сұраныстар. Предпочтение от-

беріледі жүйелерм, имеющим толық қызмет атқаратын генераторы (Мастера, Построители және т.с.с.) және обеспечивающим қолайлылығы жұмыс пайдаланушының.

Деректер қорын қорғау және дұрыстығын бақылау құралдарының сапасы. Заманауи ақпараттық жүйелерде ақпараттарды қорғау талаптары ДҚБЖ-да бірнеше шараларды қабылдауды талап етеді. Қорғау функцияларына қатынас құру бағдарламаларды өңдеу құралдары деңгейінде және пайдаланушының (қарапайым пайдаланушы, ДҚ администраторы) (8.2-бөлімін қараңыз) деңгейінде қарастырылуы керек.

ДҚ дұрыстығын бақылаудың маңызды функцияларына мыналар жатады:

- алғашқы кілт бойынша ДҚ жазбаларының бірегейлігін қамтамасыз ету (не әрбір толық функционалдық ДҚБЖ үшін);
- кестелер арасында байланыс тұтастығын (сілтемелік тұтастық) автоматты түрде бақылау кезінде жаңарту, кірістіру және жазбаларды өшіру операцияларының орындалуы (3.4-бөлім);
- ДҚ-да мәндердің дұрыстығыи тексеру (деректердің түрін бақылау, үлгімен сәйкестігін тексеру, мүмкін мәндер аумағын анықтау, анықтамалық кесте және басқалар бойынша мәнді бақылау).

Коммуникациялық құралдар сапасы. Коммуникациялық құралдар сапасын бағалау кезінде бағдарламалық өнімдердің келесі қасиеттеріне назар аударады:

- өнімнің әртүрлі желілердегі жұмысын жүзеге асыратын желілік хаттамаларды сүйемелдеу;
- ДҚ-да стандартты интерфейстерді: SQL, ODBC, IDAPI, SAA және басқалар (9.3-бөлім) сүйемелдеу;
- ДҚ ақпараттарымен топтық жұмыс жасау құралдарының бар болуы (тілдің өңдеу құралдары; пайдаланушының интерфейсі функциялары; топтарды ұйымдастыру бойынша ДҚ администраторының функциялары, рұқсат етілмеген қатынас құрулардан қорғау үшін шектеу қою және т.б.);
- ДҚ басқа форматтарға импорттаусыз немесе өзгертулерсіз қолдану және түрлендіру әдістемесі.

Бағдарлама құрушы фирма. Бағдарламалық өнімдерді таңдау кезінде өнімді шығарушыға үлкен мән беріледі. Танымал бағдарлама құрушы фирмалар дестесі, қағидаға сәйкес, келесі артықшылықтарды береді:

- өнімнің жоғары сапасы;
- құжаттандыру және әдістемелік материалдардың бар болуы;
- пайда болған мәселелер бойынша кеңес алу үшін «шұғыл байланыстың» бар болуы;
- неғұрлым жетілдірілген нұсқасы шығатынына сенімді болу.

Ескере кетейік, ДҚБЖ кезекті нұсқасы жеткілікті түрде жылдам болады. Өнімді таңдау кезінде оның құрылған уақытына назар аудару керек.

Қаржылық тұрғыдан орныққан және аппараттық-бағдарламалық құралдары үнемі дамып отырған, келешегі бар фирманың өнімін таңдаған жөн.

Құны. Бағдарламалық өнімнің құнына көбінесе бағдарламалық өнім түрі және бағдарлама құрушы фирма әсер етеді. Толық қызмет атқаратын ДҚБЖ құны әдетте $500-$1000 шамасында. ДҚ серверлері қымбатырақ. Жалпы құны қолданбалы құралжабдықтар, жүйелерді баптау құралдары, ДҚ басқару және сүйемелдеу құнын қамтиды. Кей жағдайда реляциялық ДҚ-да құрылған ірі жүйелердің жалпы құны миллион долларға жетеді. Жүйелердің жалпы құнын анықтайтын негізгі фактор, көбінесе сүйемелдейтін пайдаланушылар саны болып табылады.

WWW технологиясы пайда болғаннан кейін бағдарламалық өнімдер, сонымен қатар ДҚБЖ тегін алу мүмкіндігі туды. Мұндай өнімнің мысалы: еркін таратылатын және

«клиент-сервер» үлгілеріне негізделген постреляциялық POSTGRES95 ДҚБЖ болып табылады.

Техникалық сипаттамалары. Сапалық деңгейге қарағанда ДҚБЖ техникалық деңгейінде сипаттамалар өте көп. Солардың маңыздыларына тоқталайық. Сипаттамаларды көру ыңғайлылығы үшін оларды кесте түрінде қарастырайық.

9.1-кесте. ДҚБЖ-ның негізгі техникалық сипаттамалары

Сипаттамалар түрі	Сипаттамасы
Жалпы параметры	Операциялық орта (ОЖ сүйемелдейтін типтер және коммуникациялық хаттамалар) Жедел жадыдан алатын орны ДҚ көлемінің ең үлкен мәніне шектеу Бір уақытта қосылулар (пайдаланушылар, қосымшалар) санына шектеу
Деректермен орындалатын операцияларға шектеулер	Бағананың (өрістердің) ең үлкен өлшемі Жолдардың ең үлкен өлшемі, кбайт Ең үлкен сан: - кесте өрісіндегі; - индексті өрістегі; - кесте жолдарындағы; - бір уақытта ашық кестеде
Деректердің типтері	Ұзындығы тұрақты мәтіндік Ұзындығы айнымалы мәтіндік Сандық Бүтін санды Өзгермелі үтірлі ондық Мерзімдік Уақыт Мерзім-уақыт (мерзімі және/немесе уақыты туралы деректер) Логикалық Комментарийлер Виртуалдық (есептелетін) Екілік – графикалық, аудио-, видео- және басқа ақпараттарды (OLE, BLOB және т.б.) сақтау Гиперссылка (hyperlink) – деректер қорынан тысқары жергілікті компьютерде немесе желіде орналасқан файлдар немесе құжаттарға сілтеме
Сұраныстарды қалыптастыру және орындау құралдарының мүмкіндіктері	Сұраныстар тілі: - SQL; - QBE; - өзіндік Сұраныстар интерфейсі: - командалық жол; - үлгі (стандартты форма) Прекомпилятор және оптимизатор Сұраныстарды сақтау Кесте санына және байланыс түріне шектеу Іздеу үшін өрістің ең үлкен саны

	Сұрыптау: - кез-келген бір өріс бойынша; - бірнеше өрістер бойынша Сұраныстарда есептеулердің бар болуы Топтық операциялар және жиындармен орындалатын операциялар
Көппайдала-нушылық ортада жұмыс жасау	Типтерге шектеу қою: - ерекше; - жалпы Деңгейге шектеу қою: - ДҚ-на шектеу қою; - объектілерді өңдеуге шектеу қою (бағдарламалар мәтіндері, есеп берулер, экрандық формалар және басқалар); - кестелерге шектеу қою (файлдар); - жазбаға шектеу қою; - өрістерге шектеу қою Шектеулерді орнататын бекеттерді сәйкестендіру Түрлендірулерден кейін ақпараттарды жаңарту Уақытты бақылау және қайталап қатынас құру Транзакцияларды өңдеу: - транзакцияларды анықтау; - өзгерістерді белгілеу және кері қайту; - толық шектеу қою
Қосымшаны өңдеудің аспаптық құралдары	Пайдаланушы интерфейсі генераторы (сонымен қатар экрандық формалар, пернелер, мәзірлер, терезелер және т.б.) Есеп берулер генераторы Қосымшалар генераторы Орындалу уақыты нұсқасы (бағдарламаларды жүктеу ядросы) Тәуелсіз exe-модульдер тудыру
Импорт және экспорт	ASCII – файлы DBF – форматы WK – форматы XLS – форматы Басқа форматтар

Жүйелер сипаттамаларының көбісі түсінікті және түсініктеме берудің қажеті жоқ. Ескере кетейік, барлық ДҚБЖ-да келтірілген сипаттамалар бар.

Өнімділігін бағалау. Реляциялық ДҚБЖ талдау және сынақтан өткізуді PC Magazine журналының лабораториясы жүзеге асырады. Ол ұсынатын ДҚБЖ өнімділігін талдау әдістемесін ДҚБЖ бағдарламалары бар АЖ үшін пайдалануға болады, оны қысқаша қарастырайық.

Реляциялық ДҚБЖ тестілеу AS^3AP (ANSI SQL Standard Scalable and Portable) жинағының эталондық тестілері көмегімен орындалады. Оларда жиі кездесетін ДҚ-мен орындалатын операциялар бақыланады және жалғызпайдаланушылық және көппайдаланушылық орта қалыптасады. Сыналатын ДҚ жүз мыңдаған жазбалары бар төрт кестеден тұрады.

Қолданылатын тесттердің негізгі түрлерінің аттары:

- «Таңдау»;
- «Толық сканерлеу»;
- «Жүктеу және индексация»;
- «Жаңарту»;

- «Ерікті таңдаумен оқу»;
- «Ерікті қатынас құру жазбасы»;
- «Есеп берулерді орындау».

Құрылғыны тестілеу үшін әрбір бағдарламалық өнімді жеткізуші фирма сәйкес қолданбалы бағдарламаларды қолданады.

«Таңдау» тестісінде әрбір бағдарламалық өнім сұранысты неғұрлым жылдам орындауы мүмкін, жауабы кестелер жолының белгілі бір пайыздық мөлшерін қамтиды. Сұраныстар сандық және мәтіндік деректерді қамтиды. Мұнда индекстелген өрістер қолданылады, ал мәні берілген аралықта жатуы керек.

«Толық сканерлеу» тесті индекстелген өріс немесе кестедегі мәнді іздеу үшін қажетті уақытты өлшейді. Осылайша, ДҚ-на ең қолайлы сұраныс жасалады.

«Жүктеу және индексация» тестісінде индекстердің белгілі бір санын импорттау және құру қаншалықты жылдам орындалатыны бағаланады.

«Жаңарту» тестісі жазбаларды түрлендіру (Modify), кірістіру немесе қосу (Append) және өшіру (Delete) операцияларының орындалуы кезінде индекстерді жаңарту уақытының өлшеміне арналған.

«Ерікті таңдаумен оқу» тестісі деректерге параллель қатынас құрудың ең үлкен санын анықтауға мүмкіндік береді. Бұл сипаттамаларды алу үшін жұмыс бекеттерінде бөлек компьютерде орналасқан дестелерді орындауы тиіс бір кестенің жазбаларының кездейсоқ номерлерімен сұраныстар қалыптасты. Қабылданған жауаптар сұранысы жұмыс бекеттерінен өшіріледі.

«Ерікті қатынас құру жазбасы» тестісі алдыңғы тестіге ұқсас, тек қана бекеттерінің әрқайсысы жаңартылатын жазбаны кездейсоқ таңдайды, онда бүтін санды өріс өзгереді.

«Есеп берулерді орындау» тестісі бір жұмыс бекетінің есеп беруді дайындау және басып шығаруын жобалайды, ал басқа машиналар топтық жұмыс кезінде негізгі бекетке қойылатын талаптар ағымын имитациялайды. Имитация «Ерікті таңдаумен оқу» тестісі көмегімен жүзеге асады.

Өнімділігі параметрлері бойынша 10 неғұрлым жоғары сапалы ДҚБЖ бағаланды: Clarion Database Developer 3.0, DataEase for DOS 4.53, DataFlex 3.05 (DOS), dBase IV 2.0, Microsoft Access 2.0, Microsoft FoxPro 2.6 for DOS (Professional Edition), Microsoft FoxPro 2.6 for Windows (Professional Edition), Paradox for DOS 4.5, Paradox for Windows 4.5 (Professional Edition) және R:BASE 4.5 Plus. Жалпылай алғанда, Microsoft FoxPro дестесінің екі өнімі алдыңғы орындарды иеленді. Бұл өнім, өкінішке орай, практикалық тұрғыдан ДҚ тұтастығын бақылаудың кіріктірілген құралдарын қамтымайды.

Деректердің тұтастығын және дұрыстығын бақылау құралдары дамыған, пайдаланушы интерфейсінің үлкен мүмкіндіктері бар, өнімділік сипаттамасы жаман емес өнімдерді таңдаған дұрыс. Оларға Paradox және Access жатады.

Ескерту.

Транзакцияларды өңдеу өнімділігі бойынша (TPC – Transaction Processing Performance Council) Кеңеспен құрылған тестілердің басқа да топтары бар. TPC тестілер жинағына тұрақты түрде жетілдірілетін және толықтырылатын келесі тестілер кіреді: TPC Benchmark A (кротко – TPC-A), TPC-B, TPC-C, TPC-D және TPC-E. Бұл тестілер өнімділікті анықтау үшін құрылған.

Аппараттық құралдар құрылымын таңдау

Деректер қорында деректерді өңдеуге арналған ақпараттық-есептеуіш жүйелерді таңдау талап етілсе, онда бағдарламалық және аппараттық құралдар құрылымына негізделген мәселелерді шешу Деректерді өңдеуді бағдарламалық-аппараттық қамтамасыз етудің рационалды құрылымын анықтау үшін, жалпы жағдайда, келесі сұрақтарға жауап беру керек:

- ДҚБЖ қандай құрылымға ие (бір ағымды немесе көп ағымды);

- транзакциялар мониторы қолданылады ма;
- клиент-сервер архитектурасын қолдануға болады ма;
- жүйелер бір уақытта қанша белсенді пайдаланушыларды сүйемелдейді;
- сұраныстардың барлық жиынынан негізгі үлгіні ерекшелеуге болады ма;
- индекстеудің қандай ерекшеліктері бар;
- қандай сұраныстар индекстеуді сүйемелдейді, ал қайсысы деректер қорын сканерлеу көмегімен жүзеге асады;
- деректер қорының нақты өлшемі қандай;
- жүктемені өңдеу үшін дискілік жинақтауыштар және интерфейстік адаптерлердің жоспарланған қамтамасыз етілуі жеткілікті ма;
- деректерді, индекстерді, уақытша кестені сақтау үшін және деректердің көлемін үлкейту үшін дискілік жады көлемі жеткілікті ма;
- көптеген пайдаланушылармен жұмыс жасау үшін процессорлар жеткілікті ма;
- клиент-сервер жүйелерінде клиенттік жүйелер және серверлер арасындағы байланыстар желісін ұйымдастыру талап етіледі ма.

Аталған сұрақтарға жауап беру үшін ең алдымен, келесі факторларды ескеру керек:

1) қолданбалы есептер, әдістер және оны шешу құралдары ерекшеліктері;
2) таңдалған ДҚБЖ сипаттамалары;
3) операциялық жүйелермен жұмыс жасау мүмкіндіктері және тиімділігі;
4) аппараттық бөлімдер және желілік құралдар сипаттамалары.

Қосымшамен және/немесе ДҚБЖ-мен жұмыс жасаудың бағдарламалық ортасы операциялық жүйе (*үшінші құрамдас бөлік)* болып табылады. Олардан ақпараттық жүйелер жұмысының интегралдық бағасы тәуелді. Кей жағдайда ОЖ таңдау принциптік сипаттамаға ие. Мысалы, Windows ортасының ДҚБЖ тарапының талаптары UNIX және MS DOS пайдалануға мүмкіндік бермейді.

Есептеуіш жүйелердің маңызды бөлігі аппаратура (*төртінші құрамдас бөлік)* болып табылады. Себебі: тиімді ақпараттық жүйелер тұрғызудың қажетті шарты жүйелердің аппараттық бөлімдерінің құрылымын анықтау болып табылады.

Негізгі жады. Негізгі жады (НЖ) сипаттамалары компьютердің басқа сипаттамалары арасындағы маңыздысы - оның көлемі болады. Компьютерді таңдау кезінде оның жадысының көлемін болашақта үлкейту мүмкіндігін қарастырған жөн. Көптеген заманауи компьютерлер бастапқы жады көлемін оңай үлкейтеді.

Негізгі жадыда бағдарламалар және деректер сақталады. ДҚ-мен жұмыс жасау кезінде БЖ, қағидаға сәйкес, НЖ үлкен бөлімін сыртқы жадымен алмасу буфері (кэш) ретінде қолданады. НЖ үлкен көлемін кэш облысы алады. Оның өлшемін анықтауды қарастырайық. Буфер облысының тиімді өлшемін нақты анықтау оңай емес. Бұның негізгі себебі – алдын-ала белгісіз қажеттіліктер (шығарылатын есептер, клиенттік машиналар сипаттамалары, пайдаланушылар санаты және т.б.).

Орталық процессор. Процессордың ресурстарын серверлік жүйелерде пайдалану қолданатын қосымшадан, ДҚБЖ-нен, дербес пайдаланушылардан және уақыттан өте тәуелді. Сондықтан, орталық процессор құрамы мен қуаты бойынша бірмәнді ұсыныстар бері жалпы жағдайда мүмкін емес. Қайсібір бағалауларды тестілеу нәтижесінде беруге болады.

Енгізу/шығарудың дискілік жүйелері. Кез-келген деректер қорын жаңарту индекстерді жаңарту және журналға жазба орындаумен байланысты екені белгілі.

Дискілік енгізу/шығару ішкі жүйелерінің өнімділігіне қолданылатын жүйелік шинаның саны және түрі үлкен ықпал етеді. Әртүрлі шиналар дискілердің әртүрлі санының жұмысын сүйемелдейді. Ендеше, бір Fast SCSI-2 (10 Мб/с) шинасында

бірнеше дискілер санын (3-5) құруға болады, ал Fast-and-Wide SCSI (20 Мб/с) шинасында – 8-10 дискіге дейін.

Енгізу/шығару ішкі жүйелерінің өнімділігіне осы компоненттерді қолдану деңгейі де үлкен ықпал етеді.

Деректер қоры өңдеудің көппроцессорлық жүйелері

Деректер қорында ақпаратты өңдеу жүйелерінің өнімділігін және өміршеңдігін арттыру жолдарының бірі жақсартылған архитектуралы есептеуіш жүйелерді қолдану болып табылады. Осы үшін келесі бағыттар бойынша өңдеудің жаңа әдістері мен алгоритмдерін қолданады:

- компьютердің жедел жадысында ақпараттарды сақтау және өңдеу;
- көппроцессорлық және көпмашиналық есептеу жүйелерінде әртүрлі құрылымдермен сұраныстар орындау;
- ДҚ-да ақпараттық-бағдарламалық қамтамасыз ету операцияларын (сұрыптау, іздеу, ұйымдастыру, дестелік жүктеу/шығару және басқалар) тиімді жүзеге асыру;
- есептеуіш жүйелер түйіндерінің істен шығуы жағдайында деректерді қалпына келтіру және т.б.

ДҚ пайдаланатын жүйелердің негізгі параметрлері көлемдік (деректердің ең үлкен көлемі және олардың орналасуын сақтау мүмкіндігі) және уақытша (бөлек операциялар, транзакциялар, немесе барлық жұмыстардың уақытша орындалуы) болады. Осы параметрлер мәні ДҚБЖ-мен жұмыс жасайтын есептеуіш ортаны ұйымдастыру мен қуатынан айтарлықтай тәуелді. ДҚБЖ құратын фирмалар аталған сипаттамаларды жақсарту үшін келесі аппараттық құралдарды қолданды.

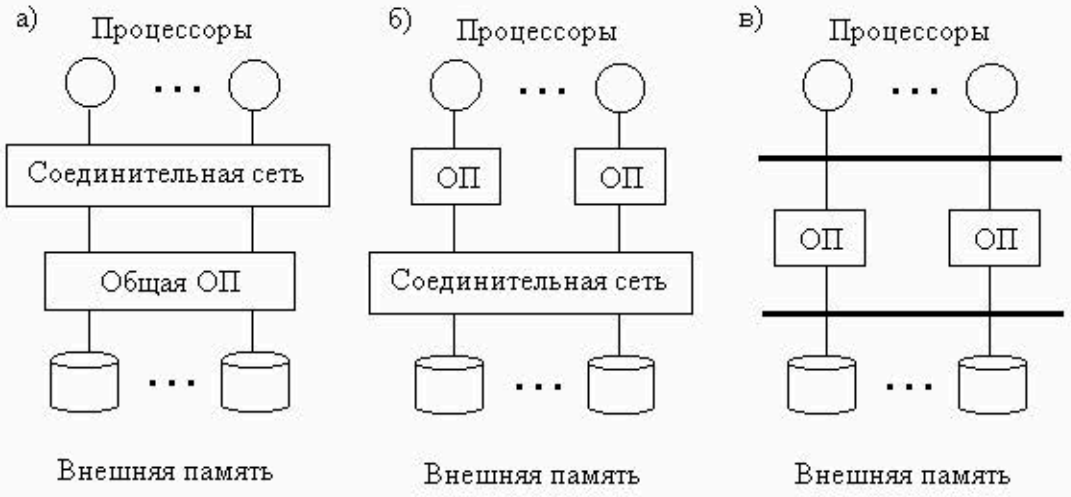
1. Өнімділігі жоғары дәстүрлі бірпроцессорлық ЭЕМ.
2. Мамандандырылған деректер қоры процессорлары – деректер қоры машиналары.
3. Көппроцессорлық құрылым базасындағы есептеуіш жүйелер.

Бірінші типті жүйелердің кеңінен таралғандығына қарамастан, қазіргі уақытта үшінші типті жүйелер жақсы нәтижелер көрсетуде. Параллель жүйелер мысалдары Teradata, Tandem, Gamma, Bubba және Arbre деректер қоры болады.

Көппроцессорлық жүйелердің екі негізгі архитектуралық бағыты бар: *күшті байланысты* және *әлсіз байланысты* есептеуіш жүйелер.

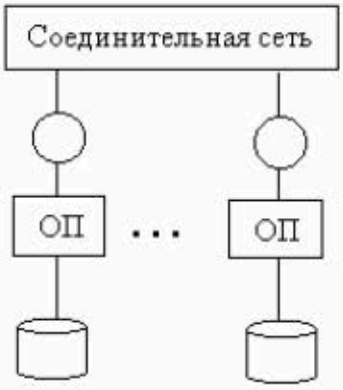
Күшті байланысты есептеуіш жүйелерге мыналар жатады:

- Жадыны бірлесе қолданатын жүйелер (9.1а-сурет), онда процессорлар жалпы жедел жадыға және барлық дискілерге қатынас құра алады (IBM/370, Digital VAX, Sequent Symmetry);
- Дискілерді бірлесе қолданатын жүйелер (9.1б-сурет), онда әрбір процессордың өзінің негізгі жадысы бар және барлық дискілерге тікелей қатынас құруды қамтамасыз етеді (IBM Sysplex және Digital VAXcluster алғашқы нұсқасы);
- Жаппай параллель жүйелер – жүздеген және мыңдаған процессорлы жүйелер, бір-бірімен ерікті түрде біріккен (9.1в-сурет).



9.1-сурет. Күшті байланысты есептеуіш жүйелер

Әлсіз байланысты көппроцессорлық есептеуіш жүйелер ақпараттарды жөнелтуді жылдам орындау ортасын біртұтас жүйелерде біріктірген компьютерлердің жиынтығы (9.2-сурет). Процессорлар хабарламалар жөнелту жолымен бір-бірімен байланысты сүйемелдейді. Әлсіз байланысқан көппроцессорлық жүйелер мысалдары: Teradata, Gamma жүйелері.



9.2-сурет. Әлсіз байланысты есептеу жүйесі

Әлсіз байланысқан көппроцессорлық жүйелердің негізгі *артықшылығы* процессорлар санын, олардың жұмысына кедергі келтірмей, жүздеп немесе тіпті мыңдап арттыру. Тағы бір құндылығы жоғары сенімділігі және ақпаратты өңдеуді басқару процесінің қарапайымдылығы болып табылады.

Әлсіз байланысқан көппроцессорлық жүйелерде деректерді өңдеудің жоғары көрсеткіштеріне жету реляциялық үлгілерді пайдалану арқасында мүмкін.

Үлестірімді ақпараттық жүйелерді ұйымдастыру кезінде ДҚ жиынтығының тұтастығын қамтамасыз етудің неғұрлым жетілдірілген сызбалары қолданады.

9.2. ДҚБЖ даму келешегі

Заманауи ДҚБЖ талдау олардың дамуының келесі бағыттарын береді:

1) неғұрлым жетілдірілген үлгілерді іздеу және деректер қорында деректерді ақпараттық-бағдарламалық қамтамасыз ету;

2) ДҚБЖ жаңа архитектурасын құру;

3) ДҚ қолдану облысын кеңейту;

4) Қарапайым пайдаланушылар, администраторлар және бағдарлама құрушыларға қызмет көрсетуді жақсарту.

Бірінші бағыт мәселелері көптеген заманауи ДҚБЖ әртүрлі ақпараттық-бағдарламалық қамтамасыз етулері екілік деректерді және *гиперсілтемелерді* сүйемелдеуі арқылы шешіледі.

Заманауи ақпараттық жүйелер кейбір жағдайларда ДҚБЖ-дан Петабайт (Petabyte – 10^{15} байтов) ретті деректер көлемін сақтау және өңдеу мүмкіндіктерін талап етеді. Бұл «үшінші жады» жинақтауыштарында орындалады. Мұндай құрылғы мысалы StorageTek корпорациясының VSM (Virtual Storage Manager – виртуалды жады менеджері) буферлік жүйелері болып табылады. VSM жүйелері деректерді жинақтайды және оларды қатқыл дискідегі деректердің буферіне сақтайды, мұнда олар магниттік лентадағы виртуалдық томдар (әрбір дискілік буферде 100 000 виртуалдық томға дейін) түрінде жинақталады. Пайдаланушының деректерді жөнелтуінің ең үлкен жылдамдығы – 45 Мбайт/с дейін.

ДҚБЖ қолданудың жаңа облыстарына келесі екі тапсырманы жатқызуға болады: өте үлкен көлемді ақпараттарды өңдеу және ЭЕМ желілерінде үлестірімді ақпаратты өңдеу.

Заманауи тұрғыдан ақпараттық қызмет көрсетудің *бейімделгіштік* қажеттілігін қамтамасыз ету мәселесі туындайды.

9.3. Деректер қорын стандарттау

Деректерді өңдеу облысында стандарттау қажеттілігі алғашқы «жаппай» ДҚБЖ енгізу кезінде туындады. Қазіргі уақытта 250-ден астам стандарттау ұйымдары комитеттері жұмыс жасайды, олар мыңдаған стандарттарды бекітті. Стандарттау сипаттамаларын және деректер қоры үшін неғұрлым кең таралған стандартты қарастырайық.

Стандарттау процесінде алдымен бағдарлама құрушы ұжым тәжірибелері және теоретиктер ұсыныстары жинақталады. Содан соң жалпылау, болашақтағы дамуын болжау және ортақ стандарттар бойынша ұсыныстар жасау орындалады. Бұдан соң қажетті мекеме өкілдерімен құжаттар келісіледі. Ең соңында, бағдарлама құруда қабылданған стандарттар жүзеге асады.

Стандарттар қабылдауға көптеген факторлар ықпал етеді. Біріншіден, стандарттарды құруды бірнеше мекемелер орындайды және олардың әрқайсысы басқалардың стандарттарынан өзгеше өз стандартын қабылдайды. Екіншіден, стандарттарды өңдеу процесін ұйымдастыру құру процесімен қатар жүзеге асады, сондықтан дайындалған құжаттар тұрақты модификацияланады. Үшіншіден, шешім қабылдауды тежейтін ғылыми зерттеулер талап етіледі. Төртіншіден, қабылданған шешімдерге компьютерлік индустрияны қуатты ұйымдастыру (саясат элементі) және т.б. ықпал етеді. Бұлардың барлығы, әрине, стандартты қабылдау уақытын ұзартады. Стандартты енгізу процесі ірі өнеркәсіптер үшін неғұрлым жеңіл, себебі олар әдетте стандартты қабылдауға қатысады. Неғұрлым кішігірім және жас ұжымдарда стандартты енгізуде қиындықтар кездеседі.

Халықаралық стандарттар және ұсынымдар барлық мүмкін қайшылықтарды жоймайды. Стандарттардың пайдасы әрине бар, себебі, оларсыз жұмыс жасау қиын. Мысалы, байланыс стандартын қабылдамай, Internet желісін ұйымдастыру мүмкін емес еді.

Неғұрлым маңызды стандарттармен танысып, пайдаланушы кез-келген стандартты оңай жүзеге асырады. Жеке жағдайда, бұл ДҚ-на қатынас құрудың стандартты тілі –

SQL тіліне қатысты. Оның жүзеге асуы құралдары әртүрлі ДҚБЖ-да бір бірінен аздаған өзгеше. Стандарттаудың негізгі тілі - SQL тілін қарастырайық.

SQL тілі

SQL тілі 70-ші жылдардың ортасында IBM фирмасының ғылыми-зерттеу лабораториясында құрылған. Тілдің бастапқы атауы – SEQUEL (Structured English QUEry Language). SQL тілі кортеж айнымалыларымен реляциялық есептеулер жүргізуге негізделген (3.7- бөлім). Сұраныстарды қалыптастырудың негізгі операторлары және ДҚ-мен іс-әрекеттер жасаудан басқа, ол сипаттау құралдары және ДҚ сызбасына өзгерістер енгізу, тұтастыққа шектеу қоюды анықтау, қатынастарға және олардың өрістеріне қатынас құруды қорғау, транзакцияларды сақтау, кері қайтуларды орындау және басқаларды қамтиды.

SQL тілін біртіндеп әртүрлі бағдарлама құрушы фирмалар қолдана бастады және SQL стандарты келесі ұжымдарда пайда болды: ANSI (American National Standards Institute – Америка Ұлттық Стандарттар Институты), SAG (SQL Access Group), X/Open (UNIX үшін стандарттар тобы), ISO (International Standard Organization – стандарттау бойынша Халықаралық ұйымдастыру), IBM фирмасы.

Неғұрлым кеңінен қолданылатын стандарттар: 1989 жылы қабылданған ANSI SQL-89 стандарты және 1992 жылы қабылданған ANSI SQL-92 және ISO SQL-92 стандарттары.

ANSI SQL-89 стандарты SQL пайдаланудың үш нұсқасын (үш интерфейс) сипаттайды: “модульдік тіл”, “кіріктірілген SQL” және “ шақыру тілі”.

Модульдік тіл дәстүрлі бағдарламалау тілдеріндегі (Си, Кобол, Фортран және басқа) бағдарламалардан шақырылатын процедуралар құру мүмкіндігін қарастырады.

Кіріктірілген SQL бағдарламаларға қарапайым бағдарламалау тіліндегі SQL-операторларды қосуды қарастырады. SQL-операторлар жиынтығы HOST-тілі деп аталады. SQL-операторларын қолдану арқылы бағдарламаларды өңдеу екі кезеңде жүзеге асады: алдымен бағдарламалау тілін ескере отырып, SQL-операторын қайсібір командаға айналдыратын ДҚБЖ қолданатын препроцессор жұмыс жасайды, содан соң – негізгі бағдарламалау тілімен кәдімгі компилятор жұмыс жасайды. SQL-операторларды қосу кезінде статикалық әдіс қолданылады.

SQL операторларының негізгі жинағы ДҚ сызбасын анықтау операторын, деректермен іс-әрекеттер жасау және таңдауды, транзакциялармен басқаруды, деректерге қатынас құруды автоматтандыруды және басқа көмекші құралдарды іске қосады.

Мүмкіндіктерінің кеңдігіне қарамастан, бұл стандарт функциональды толық емес. SQL-мен жұмыс жасау негізінде 1992 жылы жаңа стандарттар шықты. Олар кеңінен қолданылатын және мағынасы бойынша эквивалентті ISO SQL-92 және ANSI SQL-92 стандарттары.

SQL-92 стандарты, практикалық тұрғыдан, барлық заманауи бағдарламалардың деректер қорымен жұмысын сүйемелдейді. Ол алдыңғы стандарттардың функцияларын кеңейтеді, негізгі динамикалық SQL қарастырады. Бағдарламалардың орындалу жылдамдығының кемуіне қарамастан, SQL-операторлардың орындалуының динамикалық әдісі деректер қорымен жұмыс икемділігін арттыруға және қосымшаның деректер қорын ақпараттық-бағдарламалық қамтамасыз етуінен тәуелсіздігін арттыруға мүмкіндік береді.

Осы стандартты қабылдағаннан кейін жаңа SQL3 стандарты құрыла бастады.

SQL3 стандартын ANSI мен ISO бірлесе құрды. Ағымдағы SQL3 жобасы, практикалық тұрғыдан, SQL2 мамандануын толық іске қосады, сонымен қатар жаңа мүмкіндіктерді қамтиды.

SQL3 жаңа мүмкіндігі, ең алдымен, ДҚ кестесіне объектілі негізделген қамтамасыз ету. Кәдімгі реляциялық кестелермен салыстырғанда, енді кесте жолдары абстрактылық деректер типін (АДТ) және басқа жолдарға сілтемелер қамти алады, сонымен қатар идентификаторы бар.

Іште сақталатын процедуралар операторларына мыналар жатады: SQL операторлары, арифметикалық операторлар, жөнелтуді басқару операторы, функцияларды сипаттау және т.б.

ODMG-93 стандарты

ODMG-93 стандартының негізгі мақсаты қолданбалы жүйелердің деректер қоры объектілерін басқару жүйелерінен (ДҚОБЖ) тәуелсіздігін (бейімделгіштік) қамтамасыз ету болып табылады. Стандартты құру кезінде объектілі-бағытталған бағдарламалау тілдері – OMG (Object Management Group) облысындағы стандарттау мәселелерін шешу міндеті қойылды.

ODMG-93, ДҚОБЖ-ға сәйкес, бір жағынан, деректері объектілі үлгіге ие ДҚБЖ ретінде, ал басқа жағынан көрініс құралдары ретінде қарастырылады. ДҚОБЖ расширяют бағдарламалау тілдерін кеңейтеді.

ODMG-93 стандартында деректер қорының келесі компоненттеріне анықтама беріледі:

- ДҚОБЖ сүйемелдейтін объектілі үлгілер;
- ODL (Object Definition Language) объектілерін анықтау тілі;
- OQL (Object Query Language) объектілі сұраныстар тілі;
- объектілі-бағытталған бағдарламалау тілінде қолданылатын деректермен іс-әрекеттер жасау тілдері.

ODMG-93 стандарты СУОДҚ құру кезінде қолданылады.

Бақылау сұрақтары және тапсырмалар

1. ДҚБЖ таңдау негізін сипаттаңыз.
2. Бағдарламалық өнімдердің негізгі жарамдылық көрсеткіштерін атаңыз.
3. ДҚБЖ жарамдылық көрсеткіштерін атап шығыңыз.
4. ДҚБЖ негізгі техникалық сипаттамасына мысал келтіріңіз.
5. Өнімділігін бағалау үшін қолданылатын тестілердің негізгі түрін атаңыз.
6. Бағдарламалық-аппараттық қамтамасыз ету құрылымын анықтауға байланысты сұрақтар құрамын атаңыз.
7. Ақпараттық жүйелер серверін таңдау кезінде ескеру қажет факторларды түсіндіріңіз.
8. Ақпараттық жүйелердің серверлік бөлімдері үшін компьютердің негізгі жадысы өлшемі қалай анықталады?
9. Кэш өлшемін анықтаудың эмпирикалық ережелерін түсіндіріңіз.
10. Күшті байланысқан есептеуіш жүйелердің әртүрлілігін түсіндіріңіз.
11. Әлсіз байланысты есептеуіш жүйелерді сипаттаңыз.
12. Деректерді өңдеудің параллель әдістерін атаңыз.
13. ДҚБЖ дамуының негізгі бағыттарын атап шығыңыз.
14. Стандарттау процесіне жалпы сипаттама беріңіз.
15. SQL тілі стандартын атап шығыңыз және қысқаша сипаттама беріңіз.
16. ODMG-93 стандартының тағайындалуы қандай?

10. BORLAND C++ BUILDER

C++ Builder қолданбалы бағдарламалары немесе қосымшалары біріктірілген *өңдеу ортасында* (IDE — Integrated Development Environment) құрылады. Бұл ортаның пайдаланушы интерфейсі бағдарлама құрушымен өзара әрекетін ұйымдастыру үшін керек және әртүрлі басқару элементтерін қамтитын терезелерді іске қосады. Интегралдық орта құралдары көмегімен бағдарлама құрушы қосымшаның интерфейстік бөлімін тиімді жобалайды, сонымен қатар бағдарламалық код жазып, оны басқару элементтерімен байланыстырады.

10.1. Пайдаланушы интерфейсі

Өңдеудің интегралдық ортасы C++ Builder көп терезелік жүйе болып саналады. Бағдарламаны әзірлеудің біріктірілген ортасының (пайдаланушы интерфейсі) түрі баптаулардан тәуелді өзгеше болуы мүмкін. C++ Builder интерфейсін жүктегеннен кейін 10.1-суретте көрсетілгендей алты терезе іске қосылады:

- Негізгі терезе (**C++ Builder - Project1**);
- Объектілердің ағаш тәріздес құрылымын бақылаушы терезесі (**Object TreeView**);
- Объектілер инспекторы терезесі (**Object Inspector**);
- Формалар терезесі, немесе формалар Конструкторы терезесі (**Form1**);
- Код редакторы терезесі (**Unit1.cpp**).
- Класс сілтемесі терезесі (**ClassExplorer**).

Соңғы екі терезе Формалар терезесі артына орналасқан, сонымен қатар, Класс сілтемесі терезесі Код Редакторы терезесінің сол жағында орналасқан, сондықтан осы терезе де ортақ **Unit1.cpp** тақырыбына ие.

Экранда көрсетілген терезелерден басқа, сәйкес құралдарды шақыру кезінде шығатын, терезелер болуы мүмкін, мысалы, Бейнелер редакторы (**Image Editor**) терезесі. C++ Builder терезелерінің орынын ауыстыруға, өлшемдерін өзгертуге және экраннан алып тастауға (негізгі терезеден басқасын) болады.

Көптеген терезелердің бар болуына қарамастан, C++ Builder бір құжатты орта болып табылады және бір уақытта тек қана бір қосымшамен (қосымша жобасы) жұмыс жасауға мүмкіндік береді. Қосымша жобасы атауы экранның жоғарғы жағында негізгі терезенің тақырып жолында жазылады.

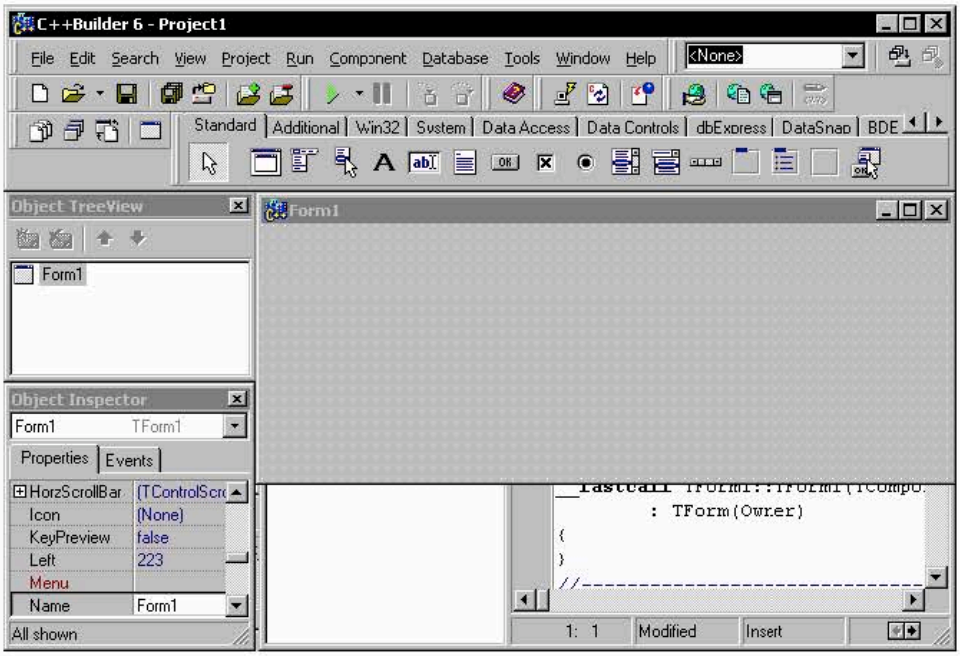
Негізгі терезені бүктеген кезде C++ Builder интерфейсі толықтай және, сәйкесінше, барлық ашық терезелер бүктеледі; негізгі терезені жапқан кезде C++ Builder жұмыс тоқтайды. C++ Builder негізгі терезесі іске қосады:

- негізгі мәзір;
- аспаптар панелі;
- компоненттер палитрасы.

Негізгі мәзір C++ Builder функцияларына қатынас құру үшін командалардың кеңейтілген жинағын қамтиды, олардың негізгілері осы командалармен байланысқан операцияларды меңгеру кезінде қарастырылады.

Аспаптар панелі негізгі мәзірдің астында, негізгі терезенің сол жақ бөлімде орналасқан және негізгі мәзірдің неғұрлым жиі қолданылатын командаларын шақыру үшін он бес батырмадан тұрады, мысалы, **File\Open** (Файл\Ашу) немесе **Run\Run** (Орындау\Орындайды).

Негізгі мәзірдің көптеген командаларын пернелер комбинациясы көмегімен де орындауға болады. Олар сәйкес команда аттарының оң жағында көрсетілген. Мысалы, **Run\Run** командасын <F9> пернесі көмегімен, ал **View\Units** (Көру\Модульдер) командасын <Ctrl>+<F12> пернелер комбинациясы көмегімен орындауға болады.



10.1-сурет. Бағдарламаны әзірлеудің біріктірілген ортасы

Барлығы 7 аспаптар панелі бар:

- ❒ **Standard** (Стандартты);
- ❒ **View** (Көру);
- ❒ **Debug** (Орындау);
- ❒ **Custom** (Пайдаланушы);
- ❒ **Desktop** (Жұмыс үстелі);
- ❒ **CORBA**;
- ❒ **Internet** (Интернет).

Аспаптар панелі бейнеленуін және оларға сәйкес батырмаларды баптауды басқаруға болады. Бұл әрекет тышқанның оң жақ батырмасын басу арқылы шақырылатын, аспаптар панелінің контексті мәзірі көмегімен орындалады.

Контексті мәзір көмегімен Компоненттер палитрасын (**Component Palette**) да басқаруға болады.

Компоненттер палитрасы негізгі мәзірдің астында, негізгі терезенің оң жақ бөлімінде орналасады және құрылатын формаларда орналасатын компоненттер жиынын қамтиды. *Компоненттер* құрастырушы блоктар болады, олардан қосымша формалары құрастырылады. Барлық компоненттер топтарға бөлінген, олардың әрқайсысы компоненттер палитрасында жекелеген беттерде орналасады, ал компоненттердің өздері белгілермен көрсетілген. Компоненттер палитрасының қажетті беті оның белгісін тышқанмен шерту арқылы таңдалады. Компоненттер палитрасының құрамында келесі беттер болады:

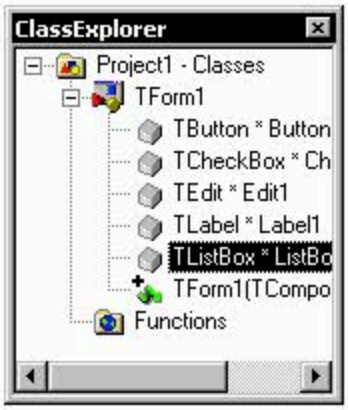
- ❒ **Standard** — стандарт;
- ❒ **Additional** — қосымша;
- ❒ **Win32** — Windows 32-разрядты интерфейсі;
- ❒ **System** — жүйелік функцияларға қатынас құру;
- ❒ **Data Access** — деректер қорының ақпараттарымен жұмыс;
- ❒ **Data Controls** — деректерді басқару элементтерін құру;
- ❒ **dbExpress** — SQL-серверлерге қатынас құру;
- ❒ **DataSnap** — деректер қорының көпдеңгейлі қосымшасын құру;
- ❒ **BDE** — BDE көмегімен деректерге қатынас құру;
- ❒ **Internet** — Internet қосымшасын құру;
- ❒ **QReport** — қосымшалардағы есеп берулер генерациясы;
- ❒ **Dialogs** — сұқбаттасулар стандарты.

Формалар терезесі (немесе формалар конструкторы) алғашқыда экранның ортасында орналасады және **Form1** тақырыбы бар. Онда формаларды жобалау орындалады, жобалау процесінде Компоненттер палитрасынан қажетті компоненттер формаға орналастырылады. Сонымен қатар, жобалау формаларды визуалды құрастырады, ал бағдарлама құрушы әрекеті қарапайым графикалық редактор ортасындағы жұмысқа ұқсас. Себебі: жобалау кезінде бағдарлама құрушы тікелей формамен жұмыс жасайды, сондықтан формалар конструкторы терезесін жиі Формалар терезесі немесе жай ғана "форма" деп атайды.

Код редакторы терезесі (**Unit1.cpp**) бағдарламалау жүйелері жүктелгеннен кейін Формалар терезесі астында орналасады және онымен толықтай жабылады. Код редакторы қарапайым мәтіндік редактор болып саналады, оның көмегімен мәтінді және басқа мәтіндік файлдарды редактрлеуге болады, мысалы, жоба файлы. Әрбір редактірленетін файл жекелеген бетте Код редакторы терезесінде орналасады, оған қатынас құру сәйкес белгішені шерту арқылы жүзеге асады. Алғашқыда **Code** бетіндегі Код Редакторы терезесі бір **Unit1** қосымшасы құрылатын қосымшаның бастапқы Form1 формалар модулі кодын қамтиды.

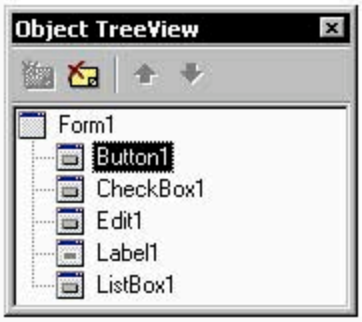
Формалар және Код редакторы терезелерінің арасында <F12> пернесінің көмегімен терезелерді тиімді ауыстырып қосуға болады.

Класс сілтемесі терезесі (**ClassExplorer**) Код редакторы терезесінің сол жағында орналасқан. Онда ағаш тәріздес құрылым түрінде формалар модулінің барлық объектілері бейнеленеді, мысалы айнымалылар және процедуралар (10.2-сурет). Класс сілтемесі терезесі **ClassExplorer\View** (Көру\Класс сілтемесі) командасымен ашылады.



10.2-сурет. Класс сілтемесі терезесі

Объектілердің ағаш тәріздес құрылымын бақылаушы терезесі (11.3-сурет) жүктелгеннен кейін жүйелер негізгі терезеде орналасады және ағымдағы формалар (алғашқыда Form1) объектілерінің ағаш тәріздес құрылымын бейнелейді. Оны **View\Object TreeView** (Көру/Объектілердің ағаш тәріздес құрылымын көру) командасымен ашуға болады.



10.3-сурет. Объектілердің ағаш тәріздес құрылымын бақылаушы терезесі

Объектілер инспекторы терезесі (10.1-сурет) Объектілердің ағаш тәріздес құрылымын бақылаушы терезесі астына, экранның сол жақ бөлімінде орналасады және ағымдағы Form1 формасы үшін объектілер қасиеттері мен оқиғаларын бейнелейді. Оны **View\Object Inspector** (Көру\Объектілер инспекторы) командасының көмегімен немесе <F11> пернесін басу арқылы ашуға болады.

Объектілер инспекторы терезесі екі беттен тұрады: **Properties** (Қасиеттер) және **Events** (Оқиғалар).

Properties беті Формалар терезесіндегі ағымдағы (таңдалған) компонент туралы ақпаратты бейнелейді және формаларды жобалау кезінде компоненттердің көптеген қасиеттерін тиімді өзгертуге мүмкіндік береді.

Events беті әртүрлі жағдайлар үшін таңдалған компонентті өңдеу процедураларын анықтайды. Егер мұндай процедуралар қандай да бір оқиғалар үшін берілсе, онда қосымшаның орындалу процесінде бұл оқиғалар пайда болған кезде процедура автоматты түрде шақырылады. Ондай процедуралар сәйкес жағдайларды өңдеу үшін қолданылады, сондықтан оларды *жағдайларды өңдеуші процедуралар* немесе қарапайым *өңдеушілер* деп атайды. Ескере кетейік, оқиғалар өзінің өңдеушілерінің қасиеттері де болады.

Нақты бір уақыт мезгілінде Объектілер инспекторы ағымдағы (таңдалған) компоненттің қасиеттері мен оқиғаларын бейнелейді., Формада орналасқан компонентті тышқанның көмегімен немесе Объектілер инспекторы тізімінен таңдауға болады. Әрбір компоненттің оның ерекшеліктерін анықтайтын қасиеттер және жағдайлар жинағы бар.

Объектілер инспекторы қасиеттер және оқиғаларды деңгейі бойынша немесе алфавит бойынша топтастыруға мүмкіндік береді. Қасиеттер (және олардың мәні) әртүрлі түрде бейнеленеді. Объектілер инспекторында тек қана оқу үшін арналған қасиеттері болады.

Объектілер инспекторы үнсіз келісім бойынша қасиеттер және жағдайлар аттарын алфавит бойынша бейнелейді (10.1-суретті қараңыз). Олардың деңгейі бойынша бейнеленуі Объектілер инспекторы контексті мәзірінің **Arrange\by Category** (Орналастыру\Деңгейі бойынша) командасымен орындалады.

Үнсіз келісім бойынша Объектілер инспекторы барлық объектілердің қасиеттері мен оқиғаларын бейнелейді. Қайсібір санатты бейнелеуді контексті мәзірдің **View** командасының ішкі мәзірінен сәйкес пунктегі (мысалы, **Action**) белгіні алу/қою арқылы ажыратуға/қосуға болады.

C++ Builder *Dock-терезелер* технологиясын сүйемелдейді, олар бір-бірімен тышқан көмегімен байланысқан. Ондай терезелер бағдарламаны әзірлеудің біріктірілген ортасының құрал-саймандық (сұқбаттасу терезесі емес) терезесі болады, сонымен қатар Объектілер инспекторы және Код сілтемесі терезесі.

Екі терезелерді жалғастыру үшін тышқанның көмегімен олардың бірін екіншісіне орналастыру керек, содан соң бұл терезе автоматты түрде екінші тереземен жапарласып орналасады. Терезелерді бөлу ортақ тақырып атауы астындағы қос сызықтың орнын ауыстыру арқылы орындалады. Жалғастырылған терезелер бірнеше бөлімдерге бөлінген ортақ бір терезеге айналады. Жалғастыру/ажырату кезінде терезе өз атауын өзгертеді. Объектілер инспекторы және Объектілердің ағаш тәріздес құрылымын бақылаушы терезелері жалғасқаннан кейін өз аттарын біріктіреді (барлық терезелер аттары үтір арқылы ажыратылып көрсетіледі).

Объектілер инспекторы және Объектілердің ағаш тәріздес құрылымын бақылаушы терезелері үшін оларды басқа терезелердің үстіне орналастыру арқылы **Stay on Top** (жоғарыдан орналастыру) режимін орнатуға болады. Бұл контекстті мәзірде бір атаулы белгілеуді қосу арқылы орындалады.

10.2. Жоба сипаттамасы

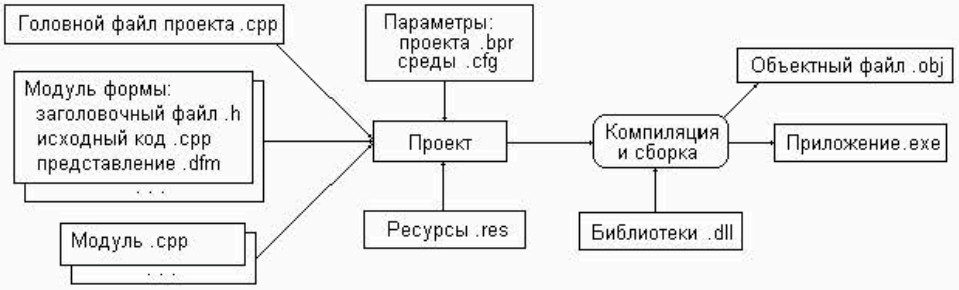
Бұл бөлімде қарастырылады: жоба құрамы, жоба файлы, формалар файлы, модульдер файлы, ресурстар файлы және жоба параметрлері.

Жоба құрамы

C++ Builder ортасында құрылатын қосымшалар жобада біріктірілген бірнеше элементтерден тұрады. Жоба құрамына келесі элементтер кіреді (жақша ішінде файл атауларының кеңейтпелері көрсетілген):

- Жобаның бастапқы коды (cpp);
- Формалар модулі мен модульдердің бастапқы коды (cpp);
- формалар мен модульдер файлының тақырыптық модульдері;
- формалар көрінісі (dfm —Windows үшін, xfm — кроссплатформалық нұсқа);
- жоба параметрлері (bpr);
- орта параметрлері (cfg);
- ресурстарды сипаттау (res).

Жоба бөлімдері арасындағы өзарабайланыстар 10.4-суретте келтірілген.



10.4-сурет. Жоба файлдары арасындағы байланыстар

Келтірілген файлдардан басқа, автоматты түрде басқа файлдар құрылуы мүмкін, мысалы, резервтік көшірмелер файлдары: ~dp — кеңейтпесі .dpr файлдары үшін; ~cp — кеңейтпесі .cp файлдары үшін; ~h — кеңейтпесі .h файлдары үшін. Жоба топтарын құрған кезде кеңейтпесі .bpg мәтіндік файл; дестелермен жұмыс жасау кезінде кеңейтпесі .bpl және .bpi екілік файлдар құрылады.

C++ Builder жүктеу кезінде автоматты түрде Project1 атауымен жаңа жоба құрылады, ол C++ Builder негізгі терезесінің тақырыбында көрсетіледі. Бұл жобада бір Form1 формасы бар, оның атауын Формалар терезесінен көруге болады. Бағдарлама құрушы үнсіз келісім бойынша қабылданған жоба атын өзгертуі мүмкін, сонымен қатар

осылайша орта параметрлерін таңдай алады. C++ Builder жүктелгеннен кейін автоматты түрде қосымшалар жүктеледі.

Әдетте жоба файлы бір каталогте орналасады. Себебі, тіпті салыстырмалы түрде қарапайым жоба жеткілікті көп файлдарды іске қосады, ал жобаға жаңа формалар қосу кезінде осы файлдар саны артады. Әрбір жаңа жоба үшін барлық жоба файлдары сақталатын жеке каталог құрған жөн.

Жобаның бас файлы

Жобаның бас файлы кеңейтпесі .cpp болатын жобаның орталық файлы болып табылады. Құрамында бір форма бар қосымша үшін жобаның бас файлы келесі түрде болады:

```
//---------------------------------------------------------------------
// Препроцессор директивалары
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------
// Макростар
USEFILE("readme.txt");
USEFORM("main.cpp", FormMain);
USERES("scrollba.res");
//---------------------------------------------------------------------------
// WinMain негізгі функциясы
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
    Application->Initialize();
    Application->CreateForm(__classid(TFormMain), &FormMain);
    Application->Run();
  }
  catch (Exception &exception)
  {
    Application->ShowException(&exception);
  }
  return 0;
}
//---------------------------------------------------------------------------
```

Жоба аты (бағдарламалары) жобаның бас файлы атымен сәйкес келеді және бұл файлды дискіге сақтау кезінде алғашқыда бұл Project1 атын көрсетеді. Онда ресурстар файлы аттары бар, жоба файлы атын өзгерткен кезде деректер файлы атауы автоматты түрде өзгереді.

Барлық жоба жиынтығы жоба файлын компиляциялау кезінде орындалады. Сонымен қатар, құрылатын қосымша (exe-файл) аты немесе динамикалық жүктелетін кітапханалар (dll-файл) жоба файлы атауымен сәйкес келеді. Бұдан былай біз динамикалық жүктелетін кітапханалар емес, қосымшалар құрылады деп есептейміз.

Жобаның бас файлы бастапқыда әртүрлі препроцессор директивалары болады. Жеке жағдайда #include <vcl.h> директивасы көмегімен тақырыптық файлды іске қосу орындалады.

Бұдан әрі ресурстар және формалар файлдарын қосуды орындайтын макростар жүктеледі. Жеке жағдайда, Main.cpp файлында орналасқан FormMain атауы бар формалар модулін қосуды орындайтын USEFORM("Main.cpp", FormMain) макросы. Өз

кезегінде, USERES("scrollba.res") макросы scrollba.res ресурстар файлын қосуды орындайды.

Бұдан әрі негізгі функция WinMain орналасқан. Бұл функциялардың бірінші және екінші параметрлері қосымшаның берілген үлгісі мен алдыңғы үлгісінің (егер бір уақытта бірнеше осындай қосымшалар орындалса) дескрипторлары болады. Үшінші параметр параметрлері бар жолға сілтеме болады, олар қосымша арқылы командалық жолға беріледі.

Жоба бағдарламасы үш операторды қамтиды; қосымшаны инициализациялау (Application->Initialize();), формалар құру (Application->CreateForm(__classid(TFormMain), &FormMain);) Form1 және қосымшаны жүктеу (Application->Run();). Аталған операторлар try блогында орналасқан, онан соң catch блогы орналасқан.

C++ Builder жобасымен бағдарлама құрушының қандай да бір операцияларды орындауы кезінде жоба файлы коды автоматты түрде қалыптасады. Мысалы, жобаға жаңа формалар қосу кезінде файлдар осы формаға қатысты екі жолға қосылады, формаларды жобадан шығару кезінде бұл жолдар автоматты түрде жойылады. Қажеттілік кезінде бағдарлама құрушы жобаға өзгерістер енгізе алады, бірақ оның бұл әрекеті жоба тұтастығын бұзуы мүмкін және сондықтан әдетте өзгертулерді тек қана тәжірибелі бағдарлама құрушылар енгізеді. Ескере кетейік, қайсібір операциялар C++ Builder жүйесінде автоматты түрде орындалмайды және жоба файлынан өзіндік кодтауды талап етеді.

Жобаның бас файлы кодының Код редакторы терезесінде бейнеленуі **Project\View Source** (Жоба\Түпнұсқаны көру) командасымен беріледі.

Жоба файлында көптеген қосымшалардың кодтары ұқсас.

Формалар файлы

Әрбір формалар үшін жоба құрамында көрініс формалары файлы (кеңейтпесі .dfm) және формалар модулі файлы: тақырыптық файл (кеңейтпесі .h) және бастапқы формалар коды файлы (кеңейтпесі .cpp) автоматты түрде құрылады.

Көрініс формалары файлы C++ Builder ресурсы болып табылады және формалар және оның компоненттері сипаттамаларын қамтиды. Бағдарлама құрушы әдетте осы файл арқылы Формалар терезесі және Объектілер инспекторын басқарады. Файлда формалар құрастыру кезінде сәйкес өзгерістер енгізуге сипаттамалар автоматты түрде жасалады. Көрініс формалары файлының құрамы оның түрін анықтайды. Қажеттілік кезінде бұл файлды экранда мәтіндік түрде көруге болады, ол **View as Text** (Мәтін ретінде көру) командасымен орындалады. Сонымен қатар, Формалар терезесі экраннан жоғалады, ал Көрініс формалары файлы құрамы Код Редакторы терезесінде ашылады, оны көруге және өңдеуге болады. Мысалы, object Button1 пернесін және object Edit1 біржолдық редакторын қамтитын Көрініс формалары файлы келесі мәтіннен тұрады:

```
object Form1: TForm1
  Left = 192
  Top = 133
  Width = 544
  Height = 375
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
```

```
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
    Left = 144
    Top = 40
    Width = 57
    Height = 49
    Caption = 'Button1'
    TabOrder = 0
  end
  object Edit1: TEdit
    Left = 224
    Top = 136
    Width = 57
    Height = 21
    TabOrder = 1
    Text = 'Edit1'
  end
end
```

Файлды экранда мәтіндік түрде көруден кері қарай формалардың сыртқы түрін көруге қайта оралу үшін **View as Form** (Форма ретінде көру) командасын орындайды.

Көрініс файлы формалардың барлық объектілерінің тізімін, форманың өзін, сонымен қатар осы объектілер қасиеттерін қамтиды. Әрбір объект үшін оның типі көрсетіледі; формалар үшін оның типі (класс) TForm1 осы формалар модулінде сипатталады. Егер формалар тақырыбын анықтайтын Caption = 'Form1' жолында Form1 орнына мән енгізілсе, мысалы, Бірінші форма мәтіні, онда формалар тақырыбы жаңарады. Бірақ іс жүзінде мұндай әрекеттер әдетте Объектілер инспекторы терезесінде орындалады.

Формалар терезесін қайталап ашу **View\Forms** (Көру\Формалар) командасымен немесе <Shift>+<F12> пернелер комбинациясымен орындалады, содан соң **View Form** (формаларды көру) сұқбаттасу терезесі ашылады, тізімнен қажетті форма таңдалады.

Бір уақытта экранда бірнеше форманы қатар көрсетуге болады. Формалар терезесін жабу үшін **File\Close** (Файл\Жабу) командасын орындайды немесе сәйкес терезедегі жабу батырмасын тышқанмен шертеді. *Формалар модулінің тақырыптық файлы* формалар класын сипаттауды қамтиды. Мысалы, (Button1), жазу (Label1) және біржолдық редактор (Edit1) орналасқан формалар үшін формалар модулінің тақырыптық файлы келесі кодты қамтиды:

```
//---------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm1 : public TForm
{
__published:   // IDE-managed Components
      TButton *Button1;
```

```
        TLabel *Label1;
        TEdit *Edit1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public: // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------
#endif
```

Бастапқыда тақырыптық файла автоматты түрде қосылатын препроцессор директивалары болады. Қажеттілік кезінде сәйкес директиваны бағдарлама құрушы қолмен қосуы мүмкін. Бұдан әрі формалар классы (біздің мысалда TForm1) сипатталады. Бөлімде компоненттер формаларында (Button1, Label1 және Edit1) орналасқан автоматты түрде қосылатын __published хабарламалар болады. private (өзіндік) және public (жалпы қолжетімді) бөлімдерінде бағдарлама құрушы өзінің ақпараттық-бағдарламалық қамтамасыз ету, айнымалылар және функциялар хабарламаларын орналастыра алады. Сонымен қатар, public бөлімінде автоматты түрде қосылатын формалар конструкторы түпнұсқасы (__fastcall TForm1(TComponent* Owner);) орналасады. private бөлімде орналасқан хабарламалар тек қана берілген модуль шегінде қолжетімді; public бөлімде орналасқан хабарламалар басқа кластар және модульдер үшін де қолжетімді. Бұдан әрі автоматты түрде қосылған PACKAGE қосымшасы орналасқан.

Жүзеге асу файлы деп те аталатын формалардың *бастапқы коды* файлы препроцессор директивалары, ақпараттық-бағдарламалық қамтамасыз ету және айнымалылар хабарламалары, сонымен қатар жағдайлар құрушының бастапқы кодын қамтиды. Файл жүзеге асуы мысалы келесі түрде болуы мүмкін:

```
//---------------------------------------------------------------
// Препроцессор директивалары
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
// формаларды объект ретінде хабарлау
TForm1 *Form1;
//---------------------------------------------------------------------
// формалар конструкторын шақыру
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
// ақпараттық-бағдарламалық қамтамасыз ету және айнымалылар,
// функция сипаттамалары хабарламалары орналасатын орын
//---------------------------------------------------------------------
// Button1 батырмасын басу кезіндегі оқиғалар өңдеушісі
void __fastcall TForm1::Button1Click(TObject *Sender)
{
```

```
Edit1->Text = "0";
}
//---------------------------------------------------------------------------
```

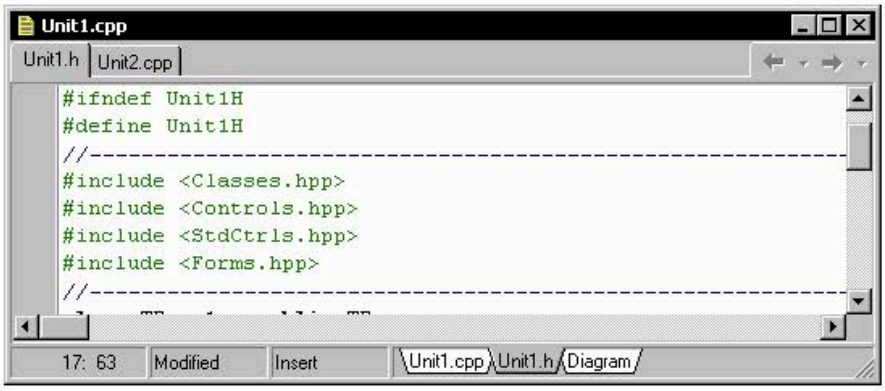
Бастапқыда файл жүзеге асуының формалар модулі автоматты түрде қосылатын препроцессор директивалары болады. Қажеттілік кезінде бағдарлама құрушы өзінің препроцессор директиваларын қосуы мүмкін, мысалы, басқа модульдерді қосу үшін.

C++ Builder жаңа формалар қосу кезінде автоматты түрде формалар модулі файлын құрады. Үнсіз келісім бойынша жобаға компоненттері жоқ TForm типті жаңа форма қосылады.

Формаға компоненттерді орналастыру кезінде, сонымен қатар жағдайлар құрған кезде формалар модуліне сәйкес өзгерістер енгізіледі. Сонымен қатар, осы өзгерістерді C++ Builder автоматты түрде орындайды. Әдетте бағдарлама құрушының барлық әрекеті бағдарламалаумен байланысты.

Формалар модулі файлдарының мәтіні Код редакторы терезесінің көмегімен бейнеленеді және редактрленеді. Формалар модулі файлын ашу үшін файлды ашудың стандартты терезесін (**File\Open** (Файл\Ашу) командасы) таңдаймыз немесе **View Unit** сұқбаттасу терезесінде **View\Units** (Көру\Модульдер) командасымен ашамыз, немесе <Ctrl>+<F12> пернелер комбинациясын қолданамыз. Формалар модулі файлын ашу терезесінде жоба файлын да таңдауға болады. Қажетті модульді (немесе жобаны) таңдап, **OK** батырмасын басқаннан кейін оның мәтіні Код Редакторы терезесінің жеке бетінде пайда болады.

Егер модуль формалар болса, онда Код Редакторы терезесінде алғашқыда формалардың жүзеге асуы файлының құрамы бейнеленеді (мысалы, Unit1.cpp). Осы формалар модулінің тақырыптық файлының құрамын көру үшін Код Редакторы терезесінің төменгі бөлігіндегі Unit1.h атауы бар қосымшаны таңдау жеткілікті (10.5-суретті қараңыз).



10.5-сурет. Код Редакторы терезесіндегі тақырыптық формалар модулі файлы түрі

Ескере кетейік, әрбір формалар (көрініс және модуль) файлдары жоба файлы атауынан өзгеше бірдей атауларға ие.

Модульдер файлы

Бағдарламалау кезінде формалар құрамындағы модульдерден басқа, қандай да бір формамен байланыссыз *ішінара модульдерді* қолдануға болады. Олар C++ тілінің ережелері бойынша жазылады және жекелеген файлдарда сақталады. Модульді жүктеу үшін оның атын препроцессор директивінде #include көрсетеді.

Жеке модульге процедуралар, функциялар, тұрақтылар және айнымалыларды жазуға болады.

Ресурстар файлы

Жоба алғашқы рет сақталған кезде автоматты түрде ресурстар файлы (кеңейтпесі .res) құрылады. Оның атауы жоба файлы атымен сәйкес келеді. Ресурстар файлы келесі ресурстардан тұруы мүмкін:

- ❒ белгілер;
- ❒ растрлік бейнелеулер;
- ❒ курсорлар.

Аталған компоненттер Windows ресурстары болады, себебі олар осы операциялық жүйенің стандартына сәйкес құрылған және түсіндіріледі.

Алғашқыда ресурстар файлы жоба белгісін қамтиды, олар үнсіз келісім бойынша факельмен бейнеленеді. Оны өзгертуге немесе алмастыруға болады.

C++ Builder құрамында ресурстар файлдарымен жұмыс жасау үшін **Tools\Image Editor** (Құралдар\Бейнелеулер редакторы) командасымен шақырылатын Image Editor графикалық редакторы қосылған

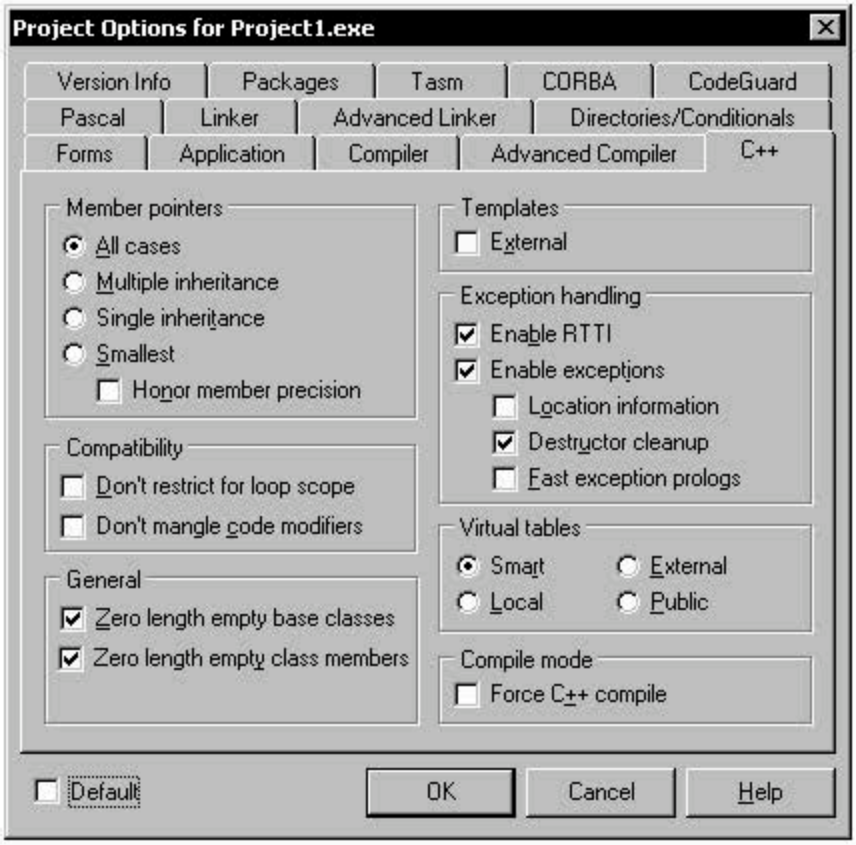
Ресурстар файлы құрылымы ағаш тәріздес, онда ресурстар топтарға бөлінген. Ресурс аты оны құрған кезде беріледі және осы ресурстарға қосымшалардаың қатынас құруы үшін қолданылады. Жоба белгісі Icon топтарында орналасады және үнсіз келісім бойынша аты MAINICON.

Кеңейтпесі res файлдардан басқа, бірнеше ресурстарды біріктіретін Image Editor редакторы келесі ресурстарды қамтитын файлдармен жұмыс жасауға мүмкіндік береді (жақша ішінде кеңейтпелері көрсетілген):

- ❒ компоненттер белгілері (dcr);
- ❒ растрлік бейнелеулер (bmp);
- ❒ қосымшаның белгілері (ico);
- ❒ курсорлар (cur).

Жоба параметрлері

Жобада параметрлерді орнату үшін жоба параметрі терезесі (**Project Options**) қолданылады, ол **Project\Options** (Жоба\Параметры) командасымен немесе <Shift>+<Ctrl>+<F11> пернелер комбинациясымен ашылады. Параметрлер топтарға бөлінген, олардың әрқайсысы өзінің бетінде жоба параметрлері терезесінде орналасады (10.6-сурет).



10.6-сурет. Жоба параметрлері терезесі

Жекелеген параметрлерді орнатқаннан кейін C++ Builder автоматты түрде сәйкес жоба файлына қажетті өзгерістерді енгізеді. Ендеше, **Forms** және **Application** беттерінің параметрлері файлдар және ресурстар жобасына, ал **Compiler** және **Linker** беттерінің параметрлері жоба параметрлері файлына енгізіледі.

Мысал ретінде жоба параметрлері файлының үзіндісін келтірейік.

```
<IDEOPTIONS>
[Version Info]
IncludeVerInfo=0
. . .
DLL=0
Locale=1049
CodePage=1251
. . .
[Debugging]
DebugSourceDirs=$(BCB)\source\vcl

[Parameters]
RunParams=
Launcher=
UseLauncher=0
. . .
[Compiler]
ShowInfoMsgs=0
LinkDebugVcl=0
LinkCGLIB=0
```

Жоғарыдағы мысалдан жоба параметрлері файлы мәтіндік файл болатынын көреміз, онда параметрлер және олардың мәні жолдарға жазылған.

10.3. Компиляция және жобаны жүзеге асыру

Компиляциялау процесінде пайдалануға дайын файл құрылады, онда *қосымшалар* (кеңейтпесі exe) немесе *динамикалық жүктелетін кітапхана* (кеңейтпесі dll) болуы мүмкін. Жоғарыда айтылғандай, біз тек қана қосымшаны қарастырамыз. Компиляция нәтижесінде алынатын қосымша аты жоба файлы атымен сәйкес келеді, ал қосымшалардың өзі дербес болып табылады және өзінің жұмыс жасауы үшін C++ Builder қосымша файлдарын талап етпейді.

Ескертулер.

Егер қосымшаның динамикалық орындалу процесінде басқа файлдар, мысалы, суреттер немесе анықтама файлы қолданылса, онда бұл файл бар болуы керек.

Деректер қорымен жұмыс жасайтын қосымшаны құрған кезде деректер қорын, сонымен қатар деректер қоры процессорын құрайтын файлдар керек.

Компиляция **Project\Compile <Project1>** (Жоба\ <Жоба1> компиляциялау) командасын шақыру арқылы немесе <Ctrl>+<F9> пернелер комбинациясымен орындалады. Командада дәл осы кезде құрылатын жоба аты жазылады (алғашқыда Project1). Жобаны басқа атаумен сақтау кезінде сәйкесінше мәзір командасындағы жоба аты да өзгеруі керек.

Қосымша үшін жобаны компиляциялау жобаны өңдеудің кез-келген кезеңінде орындалуы мүмкін. Бұл формалардың жекелеген компоненттерінің тиімді жұмыс жасау түрі мен дұрыстығын тексеру үшін, сонымен қатар құрылатын код фрагменттерін тестілеу үшін керек. Жобаны компиляциялау кезінде төменде келтірілген әрекеттер орындалады.

❒ Соңғы компиляциялау кезінен кейін құрамы өзгерген барлық модульдер файлдары компиляцияланады. Егер модульдің бастапқы мәтіні қандай да бір себептермен компиляторға қолжетімді емес болса, онда ол қайтадан компиляцияланбайды.

❒ Егер модулге өзгерістер енгізізілген болса, онда тек қана осы модуль ғана емес, оның көмегімен #include препроцессор директивалары қолданатын модульдер қайтадан компиляцияланады.

❒ Қайтадан компиляциялау модулі объектілік файлдарға (кеңейту obj) немесе қосылатын файлдарға өзгерістер енгізу кезінде жүзеге асады.

❒ Жобаның барлық модульдері компиляцияланғаннан кейін жоба файлы компиляцияланады және жоба файлы атымен орындалатын қосымша файлы құрылады.

Компиляциядан басқа, жобаны *құрастыру* орындалуы мүмкін. Құрастыру кезінде жобаға кіретін, өзгерістердің енгізілгендігінен немесе енгізілмегендігінен тәуелсіз барлық файлдар компиляцияланады. Жобаны құрастыру үшін **Project\Build <Project1>** (Жоба\ <Жоба1> құрастыру) командасы қолданылады.

Жобаны орындауға жүктеу C++ Builder ортасында және Windows ортасында жүзеге асады.

C++ Builder ортасында жобаны орындау **Run\Run** командасымен немесе <F9> пернесін басу арқылы жүзеге асады. Сонымен қатар, құрылған қосымшалар өз жұмысын бастайды. Егер жоба файлына өзгерістер енгізілсе, онда жобаны компиляциялау алдын-ала орындалады.

Run\Make командасының көмегімен компиляция және қосымшаның орындалуынсыз жобаны құрастыру (алынған файл кеңейтпесі .exe) беріледі. **Make** және **Run** командалары әрекеттерінің айырмашылығы - қосымшаның жүктелмеуі.

Жүктелген қосымшалар C++ Builder ортасынан тысқары жүктелген қосымшалар сияқты жұмыс жасайды, бірақ қайсібір ерекшеліктері бар:

❒ қосымшаның екінші көшірмесін жүктеу мүмкін емес;

❒ жобаны өңдеуді жалғастыру тек қана қосымшалар жұмысын аяқтағаннан кейін ғана мүмкін;

❒ қосымшаның тұрып қалуы (тоқтап қалуы) кезінде оны орындауды C++ Builder құралдарымен **Run\Program Reset** (Орындау\қайтадан жүктеу бағдарламалары) командасының көмегімен немесе <Ctrl>+<F2> пернелер комбинациясымен аяқтау қажет.

C++ Builder ортасында қосымшаны жүзеге асыру үшін бағдарлама құрушы құралдарын қолдануға болады. *Windows ортасында* құрылған қосымшаларды, кез-келген басқа қосымшалар сияқты, мысалы, Сілтеуіш көмегімен жүктеуге болады.

10.4. Қосымшаны құру

C++ Builder *RAD* (Rapid Application Development, қосымшаны жылдам құру) *жүйелері* деп аталатын *визуальды бағдарламалау жүйелеріне* жатады. C++ Builder ортасында қосымшаны құру екі өзарабайланысқан кезеңді іске қосады:

❒ қосымшаның пайдаланушы интерфейсін құру;

❒ қосымшаның қызметін анықтау.

Қосымшаның *пайдаланушы интерфейсі* пайдаланушы мен қосымшаның өзара әрекеті әдісін анықтайды. Интерфейс формада компоненттердің орналасу жолымен құрастырылады, оны *интерфейстік компоненттер* немесе *басқару элементтері* деп атайды. Қосымшаның пайдаланушы интерфейсі көмегімен Формалар терезесі құрылады.

Қосымшаның қызметі қандай да бір жағдайлар пайда болған кезде орындалатын процедуралармен анықталады, мысалы, пайдаланушының элементтері басқару формаларымен әрекеттер жасауы кезінде.

Осылайша, қосымшаны өңдеу процесінде формаға компоненттер орналастырылады, олар үшін қажетті қасиеттер орнатылады және жағдайларды өңдеушілер құрылады.

Қарапайым қосымшаның мысалы

Қарапайым қосымшаларға мысал келтірейік. Бұл жағдайда "құру" сөзі арттықтау болып табылады, себебі ешқандай құру және сонымен қатар бағдарламалау қажет емес: C++ Builder бастапқыдан бір формадан тұратын дайын қосымшалар ұсынады.

Жаңа қосымшаны құрғаннан кейін C++ Builder бірден бағдарлама құрушыға "бос" форма ұсынады. Бұл форма тура мағынада бос емес — ол Windows терезесінің негізгі элементтерін қамтиды: **Form1** тақырыбы, терезені бүктеу, ашу және жабу батырмалары, терезе өлшемдеріне өзгерістер енгізу және терезенің жүйелік мәзірінен батырманы шақыру.

Кез-келген қосымшалар Windows операциялық жүйесінде сәйкес терезеде орындалады. Тіпті егер ол қызметтік тұрғыдан ештеңе жасамаса да, яғни бос болса да, оның өз терезесі бар. C++ Builder — бұл Windows үшін қосымшаны өңдеу ортасы, сондықтан кез-келген құрылатын қосымша үшін автоматты түрде терезе (форма) ұсынады, оларда екі файл бар — сипаттамамен және модульмен.

Ендеше, қарапайым қосымшалар әрбір жаңа жобамен жұмыс басында автоматты түрде құрылады.

Қарапайым қосымшалар бағдарлама құрушыны барлық қажеттіліктермен қамтамасыз ететін тірек немесе дайын нұсқа. Ендеше, терезелермен жұмыс жасау үшін процедуралар дестесін құру қажеттілігі жоқ. Бұның барлығы C++ Builder

құрушылармен жүзеге асқан, және қосымшаның дайын нұсқасы толық аяқталған болып саналады.

Ескере кетейік, терезе (онымен бірге қосымшалар) шындығында пайдаланушының тұрғысынан қарағанда ештеңе жасамайды — әрбір қосымша үшін маманданған қызмет ұсынбайды. Сонымен қатар, бұл бос терезе бағдарлама құрушы тұрғысынан қарағанда жеткілікті түрде үлкен қызмет атқарады. Мысалы, ол тышқанмен және пернетақтамен байланысты пайдаланушының әрекетін күтеді, және өлшем өзгерісіне, орын ауыстыру, жабу және қайсібір басқа командаларға орай әрекет етеді. Терезенің бұл мүмкіндіктерін толық дәрәжеде тек қана бағдарлама құрушы бағалай алады.

Жобаны компиляциялау кезінде арнайы динамикалық жүктелетін кітапханалар (DLL) дестесін қолдануға болады, ол қосымшаның өлшемін айтарлықтай деңгейде азайтуға мүмкіндік береді. Бірақ, сонымен қатар қосымшалар енді дербес болмайды және өзінің жұмыс жасау процесінде дестелерге қатынас құрады.

Қосымшаны құрастыру кезінде бағдарлама құрушы қарапайым қосымшаға жаңа формалар, басқарушы элементтер, сонымен қатар жаңа жағдайларды өңдеушілер қосады.

Пайдаланушы интерфейсін құру

Қосымшаның пайдаланушы интерфейсі бағдарлама құрушы компоненттер палитрасынан таңдаған және формада орналастырған компоненттерін құрайды. Интерфейс құрастыру кезінде қосымша WYSIWYG (What You See Is What You Get — "не көрсең соны аласың") принципімен жұмыс жасайды, және бағдарлама құрушы қосымша құрған кезде көрген форманың түрін оны орындау кезінде дәл сол күйінде көреді.

Компоненттер құрылымдық бірліктер болады және визуалды (көрінетін) және визуалды емес (жүйелік) болып бөлінеді. Сонымен қатар, "көрінетін" және "көрінбейтін" түсінігі тек қана орындалу кезеңіне жатады, жобалау кезеңінде қосымшаның барлық компоненттері көрінеді.

Визуалды компоненттерге, мысалы, батырмалар, тізімдер немесе ауыстырып қосқыштар, сонымен қатар форманың өзі жатады. Өйткені визуалды компоненттер көмегімен пайдаланушы қосымшаларды басқарады, оларды *басқарушы компоненттер* немесе *басқару элементтері* деп те атайды. Визуалды компоненттерді қосымшаның пайдаланушы интерфейсі құрастырады.

Визуалды емес компоненттерге көмекші, бірақ маңызды әрекеттерді орындайтын компоненттер жатады, мысалы, таймер Timer (уақыт аралығын анықтауға мүмкіндік береді).

Қосымшаның интерфейсін құрған кезде әрбір компонент үшін келесі операциялар орындалады:

- ❒ компоненттер палитрасында компонентті таңдау және оны формада орналастыру;
- ❒ компонент қасиеттеріне өзгерістер енгізу.

Бағдарлама құрушы бұл операцияларды Компоненттер палитрасы және Объектілер инспекторын пайдаланып Формалар терезесінде орындайды. Сонымен қатар, бағдарлама құрушы әрекеті графикалық редактор ортасындағы жұмысқа ұқсас, ал қосымшаның интерфейсін құру процесі дәстүрлі бағдарламалауға қарағанда үлкен көлемді құрастыруды көрсетеді. Байланыстарда интерфейс құру бойынша жұмысты құрастыру деп атайды.

Компоненттер палитрасында компонентті таңдау қажетті компонентте тышқанды шерту арқылы орындалады, мысалы Button батырмасында, нәтижесінде оның белгісі батырылған (басылған) түрге келеді. Егер бұдан соң формалардың бос орнында тышқанды шертсек, онда осы формада таңдалған компонент пайда болады, ал оның компоненттер палитрасындағы белгісі қарапайым (басылмаған) түрге келеді.

Компоненттер белгілері компоненттердің тағайындалуын көрсетеді, және қажетті компонентті таңдау жеткілікті түрде жылдам жүзеге асады. Сонымен қатар, әрбір компонентке тышқанның сілтемесін апару кезінде оның тағайындалуы мен ақпараттар көрінеді.

C++ Builder ортасында кластардың (объектілерді ақпараттық-бағдарламалық қамтамасыз ету) белгілеулері, сонымен қатар компоненттер, T әрпінен басталады. Кей жағдайда компоненттерді белгілеу үшін атаулар орнына (Button, Label) типтер (TButton, TLabel) қолданылады.

Компоненттер формада орналасқаннан кейін C++ Builder жүйелері модуль файлына және көріністер формаларына автоматты түрде өзгерістер енгізеді. Формалар класын (тақырыптық формалар модулі файлы) сипаттауда әрбір жаңа компонент үшін жол қосылады:

<Компонент типі> *<Компонент аты>;

Жаңа компонент аты оның қасиеттер Name мәні болып табылады, ал типі ақпараттық-бағдарламалық қамтамасыз етуде таңдалған типпен сәйкес келеді. Мысалы, Button1 батырмасы үшін бұл жол алғашқыда келесі түрде болады:

```
TButton *Button1;
```

Көрініс файлында Button батырмасы үшін келесі код жазылуы мүмкін:

```
object Button1: TButton
  Left = 98
  Top = 110
  Width = 75
  Height = 25
  Caption = 'Button1'
  TabOrder = 0
end
```

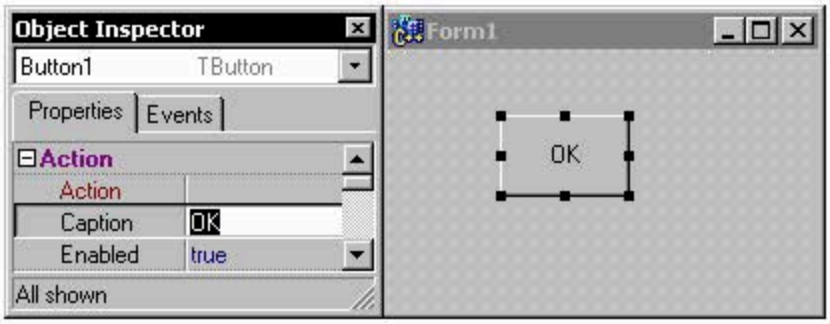
Бірнеше бірдей компоненттердің формада орналасуы үшін компоненттер палитрасынан компонентті таңдау алдында <Shift> пернесін басып ұстап тұру тиімді. Бұл жағдайда тышқанды формалар облысында шерткеннен кейін және таңдалған компоненттің онда орналасқанынан кейін оның палитрадағы белгісі ерекшеленген түрде қалады, және формада тышқанды әрбір келесі шерту формада тағы бір сондай компоненттің пайда болуына әкеледі.

Формада компонента орналасқаннан кейін тышқанның көмегімен оның орнын және өлшемін өзгертуге болады. Бірнеше компоненттер жағдайында бір немесе басқа компонентті алдыңғы немесе соңғы орынға орналастыруға болады. Сонымен қатар, бағдарлама құрушы әрекетінің әдеттегі графикалық редактор әрекеттерінен айырмашылығы жоқ. Бір уақытта формада бірнеше компоненттерді ерекшелеу <Shift> пернесінің көмегімен жүзеге асады.

Үнсіз келісім бойынша компоненттер формада торкөз бойынша тураланады, оны **Snap to grid** (торкөз бойынша туралау) арқылы орындайды. Кейбір жағдайларда бұл команданың белгісін өшіріп қойған жөн, мысалы, компоненттер формада тығыз орналасқан жағдайда. Үнсіз келісім бойынша торкөз қадамы сегіз пиксельге тең, ал торкөз жобалау кезінде формалар кеңістігінде бейнеленеді. Торкөз бойынша туралау қажеттілігі, торкөз көрінісі (**Display grid** белгішесі) және торкөз қадамы өлшемі **Tools** (Құралдар) мәзірінің аттас командасымен шақырылатын **Environment Options** (Орта параметрлері) сұқбаттасу терезесінің **Designer** (Конструктор) қосымшасында орнатылады.

Компонент формада ерекшеленген болса және оның айналасында бөлу маркері пайда болса, онда компоненттің сыртқы түрі Объектілер инспекторы терезесінде көруге болатын оның қасиеттерімен анықталады (11.7-сурет). Формалар қасиеттеріне қатынас

құру осыған ұқсас жүзеге асады, бірақ форма таңдалған жағдайда маркермен ерекшеленбейді. Формаларды бөлу (таңдау) үшін оның кез-келген жерінде тышқанды шерту жеткілікті.



10.7-сурет. Компонент қасиеттеріне қатынас құру

Объектілер инспекторы терезесінің жоғарғы бөлігінде орналасқан ашылған тізімде компонент аты және оның типі бейнеленеді. Компонентті таңдау сәйкесінше, оның қасиеттеріне қатынас құру Объектілер инспекторының осы тізімінде орындалады. Таңдаудың мұндай тәсілі компонент басқа объектілермен толық жабық болғанда тиімді.

Объектілер инспекторы терезесінің төменгі бөлігінде сол жақта компоненттің барлық қасиеттерінің атаулары беріледі, олар қосымшаны өңдеу кезеңінде қолжетімді болады. Әрбір қасиеттердің оң жағында оның мәні берілген. Ескере кетейік, компоненттердің осы қасиеттерінен басқа да қасиеттері болуы мүмкін және олар тек қана *қосымшаны орындау кезінде* қолжетімді.

Қасиеттер бейнелеу әдісін және қосымшаның орындалу кезінде жұмыс жасайтын компоненттерін анықтайтын атрибуттарды білдіреді. Алғашқыда қасиеттер мәні үнсіз келісім бойынша беріледі. Компонент формаға орналасқаннан кейін оның қасиеттерін жобалау процесінде немесе қосымшаның орындалу барысында өзгертуге болады.

Жобалау процесінде қасиеттерді басқару компоненттер қасиеттері мәндерін тікелей формалар терезесінде немесе Объектілер инспекторы көмегімен өзгертуді білдіреді.

Бағдарлама құрушы компонент қасиеттері мәнін, қажетті мәнді енгізіп немесе таңдап, өзгертуі мүмкін. Сонымен қатар, бір уақытта сәйкес компонент өзгереді, яғни жобалау кезінде жасалған өзгерістер нәтижелері көрінеді. Мысалы, батырма (Caption) тақырыбын өзгерту кезінде ол бірден оның жоғарғы жағында бейнеленеді.

Қасиеттердің жаңа мәнін бекіту үшін <Enter> пернесін басу немесе компоненттің басқа қасиеттеріне өту жеткілікті. Өзгерістерді болдырмау <Esc> пернесімен жүзеге асады. Егер қасиеттер мәні дұрыс берілмесе, онда ескертетін хабарлама шығады, ал өзгерістер мәні қабылданбайды. Қасиеттерге өзгерістер енгізу автоматты түрде формаларды сипаттау файлында ескеріледі.

Компоненттер қасиеттерінің көпшілігінің (мысалы, Color (түсі), Caption (тақырыбы) және Visible (көрінуі) қасиеттері үшін) үнсіз келісім бойынша мәндері бар.

Қосымшалардағы компоненттерге қатынас құруға Name қасиеті арналған, оның алғашқы мәні автоматты түрде келесі жолмен қалыптасады: компонент атына оның номері формада орналасу реті бойынша қосылады. Мысалы, бірінші батырма Button аты Button1, екінші — Button2 және т.б. Алғашқыда қасиеттер мәнінен Name өз мәнін және Caption қасиеттерін алады.

Әдетте бағдарлама құрушы компоненттерге, үнсіз келісім бойынша атауларға қарағанда, неғұрлым ақпарат беруші аттарды бергісі келеді. Сонымен қатар, компонент аты компонент типі және оның қосымшалардағы тағайындалуы туралы деректерді қамтығаны жөн. Ендеше, терезені жабуға арналған TButton типті батырмаға btnClose

немесе ButtonClose атын беруге болады. Әрбір бағдарлама құрушы өз бетімен қолайлы ережелер тағайындайды. Қарапайымдылық үшін біз жиі қолданылатын, үнсіз келісім бойынша қабылданатын атауларды, мысалы, Form1, Button1 немесе Edit1 пайдаланамыз.

Компонент өлшемдерімен және орналасуымен (мысалы, Left және Top) байланысқан қасиеттер мәні компоненттің тышқанмен орнын ауыстыру немесе оның өлшемін өзгерту кезінде автоматты түрде өзгереді.

Егер формада бірнеше компонентер ерекшеленген болса, онда Объектілер инспекторы терезесінде осы компоненттердің барлығы үшін орындалатын қасиеттер көрсетіледі. Сонымен қатар, Объектілер инспекторы терезесінде жасалған өзгерістер барлық ерекшеленген компоненттерге тән.

Объектілер инспекторында мәндердің қасиеттерін орнату үшін автоматты түрде қосылатын қасиеттер редакторы қолданылады:

❒ *қарапайым* (мәтіндік) —қасиеттер мәні кәдімгі жолдық символдар ретінде енгізіледі немесе редактрленеді, сандық немесе жолдық тип ретінде сипатталады; Caption, Left, Height және Hint үшін қолданылады;

❒ *саналушы* — қасиеттер мәні ашылған тізімнен таңдалады. Тізім қасиеттер мәні облысына тышқан сілтемесін апару кезінде пайда болатын бағыттауышқа тышқанды шерту кезінде ашылады. Қажетті мәнді таңдамай, пернетақтадан енгізуге болады, бірақ әдетте іс жүзінде бұл жасалмайды, себебі тек қана ұсынылатын мәндердің бірін енгізу керек. Сонымен қатар, қиындығы және қателік жіберу ықтималдығы артады. Мұндай қасиеттер FormStyle, Visible және ModalResult үшін қолданылады;

❒ *жиынтық* —қасиеттер мәні ұсынылған жиын мәндерінің комбинациясы болып саналады. Объектілер инспекторы терезесінде қасиеттер атының сол жағында "+" белгісі бар. Қасиеттер мәнін қалыптастыру қосымша тізімі көмегімен орындалады, ол тізім қасиеттер атына екі рет шерту арқылы ашылады. Бұл тізім қасиеттердің барлық мүмкін мәндерінің тізімін қамтиды, әрбір мәннің оң жағында true немесе false көрсету керек. true таңдауы берілген мән мәндер комбинациясын іске қосатынын білдіреді, ал false — жоқ. Мұндай қасиеттер BorderIcons және Anchors үшін қолданылады;

❒ *объект* — қасиеттер объект болып табылады және, өз кезегінде, басқа қасиеттерді (ішкі қасиеттер) қамтиды, олардың әрқайсысын бөлек редактрлеуге болады. Мұндай қасиеттер Font, Items және Lines үшін қолданылады.

Қасиеттер-объект мәні облысында жақша ішінде объект типі көрсетіледі, мысалы, (TFont) немесе (TSrings). Қасиеттер-объект үшін атының сол жағында "+" белгісі болуы мүмкін, бұл жағдайда оның қасиеттерін басқару жиынтық типті қасиеттер сияқты орындалады. Бұл тізімнің сол жақ бөлімінде ішкі қасиеттер аттары, ал оң жақта мәндері жазылады. Мәні облысында үш нүктелі батырма бейнеленуі мүмкін. Бұл берілген қасиеттер үшін арнайы редактор барын білдіреді.

Қосымшаның орындалуы кезінде компоненттер қасиеттері мәнін (Объектілер инспекторы терезесінде қолжетімді) меншіктеу көмегімен өзгертуге болады, мысалы, формаларда оқиғалар құру. Ендеше, Button1 батырмасының тақырыбын өзгертуді келесі түрде орындауға болады:

```
Button1->Caption = "OK";
```

Мұндай тәсіл үлкен жұмыс көлемін талап етеді, сонымен қатар, қосымшаның орындалу кезінде ғана күшіне енеді және өңдеу кезеңінде көрінбейді. Бұл кейбір жағдайларда визуалды компоненттермен басқаруды қиындатады.

Қосымшаның қызметін анықтау

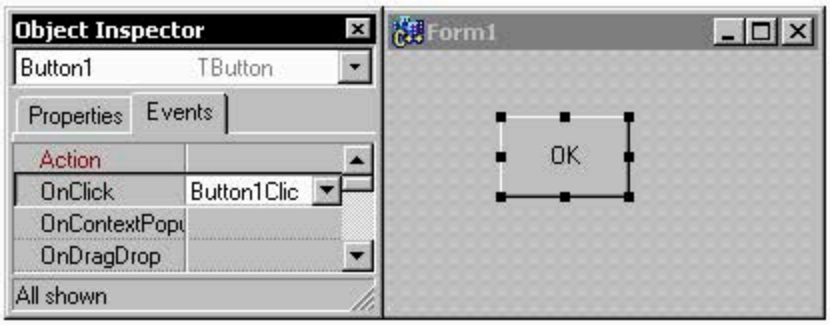
Қосымшалардың интерфейстік бөлімдерін өңдеудің кез-келген кезеңінде оны орындауға жүктеуге болады. Компиляциядан кейін экранда қосымшаның формасы

пайда болады. Форманың экрандағы орнын ауыстыруға, оның өлшемін өзгертуге, бүктеуге және ашуға, сонымен қатар жабуға болады.

Келтірілген әрекеттер әрбір формаға тән және қосымшаның тағайындалуы мен оның ерекшеліктерінен тәуелсіз. Формада, қағидаға сәйкес, қосымшаның интерфейсін құрайтын компоненттер орналасқан. Бағдарлама құрушының міндеті —осы компоненттердің пайдаланушының әрекетіне реакциясын анықтау, мысалы, батырманы басу кезінде. Бұл реакция қосымшаның *қызметін* анықтайды .

Қосымшаның интерфейсін құрған кезде формада терезені жабуға арналған Button батырмасы болсын. Үнсіз келісім бойынша бұл батырма аты және тақырыбы Button1, бірақ тақырыбы (Caption қасиеттері) көмегімен Объектілер инспекторы терезесі — OK белгілеумен алмастырылсын. Қосымшаның орындалуы кезінде Button1 батырмасын тышқанның немесе пернетақтаның көмегімен басуға болады. Батырма визуалды басуды бейнелейді, бірақ формаларды жабуға байланысты ешқандай әрекет орындамайды. Себебі, батырма үшін пайдаланушының қандай да бір әрекетіне реакциясы анықталмаған. Реакция болуы өңдеу үшін процедурасын құру немесе көрсету керек.

Оқиғаларды өңдеу процедураларын құру үшін, ең алдымен формада батырманы ерекшелеу және Объектілер инспекторы терезесінің **Events** (Оқиғалар) бетіне өту керек, мұнда берілген батырма үшін барлық мүмкін оқиғалар көрсетіледі (11.8-сурет).



10.8-сурет. Компонент оқиғаларына қатынас құру

Батырманы басқан кезде OnClick оқиғасы пайда болады. OnClick оқиғалары мәні облысында тышқанды екі рет шерту нәтижесінде C++ Builder автоматты түрде формалар модулінде процедуралар-өңдеуші дайын нұсқасын құрады. Сонымен қатар, Редакторы терезесі коды арт жаққа көшеді де, курсор процедуралар орнына орналасады, мұнда бағдарлама құрушы Button1 батырмасын басқанда орындалатын код жазуы керек. Себебі: **OK** батырмасын басқан кезде терезе жабылуы керек, онда Form1->Close() немесе жай ғана Close() деп жазылады. Толықтай келесі түрде жазылады:

Тақырыптық файл:

```
//---------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm1 : public TForm
{
```

```
__published:    // IDE-managed Components
    // атауы ОК болатын TButton типті батырма объектісін жариялау
     TButton *OK;
    // өңдеуші процедуралар тақырыбы
    void __fastcall OKClick(TObject *Sender);
private:// User declarations
public:       // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Файл жүзеге асуы:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::OKClick(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------------
```

Мұнда бағдарлама құрушы терген код қою жазумен ерекшеленіп көрсетілген; барлық қалғандарын C++ Builder ортасы автоматты түрде құрды.

Жағдайларды өңдеушіде Sender параметрі оқиғаны алған және өңдеуші шақырған компонентті көрсетеді. Кей жағдайда, әртүрлі объектілер үшін ортақ өңдеуші болған пайдалы. Мұндай жағдайда Sender параметрін қолдану өте пайдалы. Ескере кетейік, ол функциялар тақырыбында TObject *Sender түрінде көрсетілген.

TObject класы барлық C++ Builder кластарының базалық класы болып табылады, бірақ онда өңдеу кезінде қолдануға жарамды қасиеттер жоқ. Мысалы, өңдеушіден біржолдық Edit редакторына "компонентке шерту орындалды" мәтінін қосу талап етілсін. Компонент аты оның Name қасиеттер мәнімен анықталады. Себебі TObject класында бұл қасиеттер болмаса, онда параметра типін TComponent класына апару керек, онда бұл компоненттер алғаш рет пайда болады. Нәтижесінде аталған тапсырманы шешетін оператор келесі түрде жазылады:

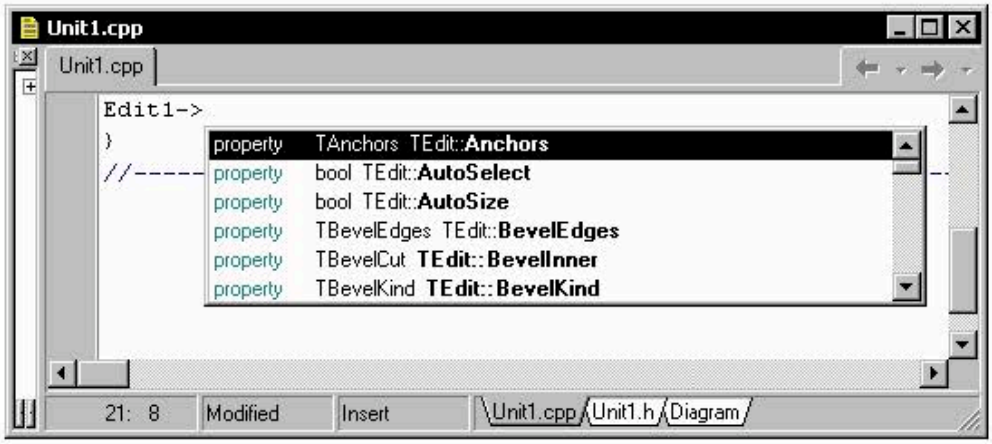
```
Edit1->Text = "компонентке шерту орындалды"+((TComponent *)Sender)->Name;
```

немесе

```
Edit1->Text = "компонентке шерту орындалды "+((TControl *)Sender)->Name;
```

Мұнда ((TComponent *)Sender)и ((TControl *)Sender) өрнектері Sender келтірілген тип параметріне сәйкес класс объектісі типін көрсетеді.

C++ Builder ортасы қасиеттер, объектілер әдісін және стандартты конструкциялар C++ тілі жазбасын шақыруды автоматтандыруды қамтамасыз етеді. Ендеше, объект атын көрсеткеннен кейін және -> белгісінен соң, осы объектінің қолжетімді қасиеттері мен әдістерін қамтитын тізім автоматты түрде пайда болады (10.9-сурет). Қажеттілік кезінде <Ctrl>+<бос орын> пернелер комбинациясының көмегімен бұл тізімді шақыруға болады. Аты таңдалған қасиеттер немесе әдістер -> белгісінің оң жағына автоматты түрде қосылады. Егер әдіс параметрлерді қамтыса, онда олардың құрамы және параметрлер типтері бейнеленеді.



10.9-сурет. Edit1 объектісінің әдістері және қасиеттерінің тізімі

Тізім <Ctrl>+<J> пернелер комбинациясымен шақырылады. Қажетті конструкцияны таңдағаннан кейін автоматты түрде оның коды қосылады. Мысалы, for операторын таңдау кезінде кодта келесі мәтін пайда болады:

for (; ;)

Өңдеуші аты TForm1::OKClick компонент атына оқиғалар атын қосу арқылы құрылады. Бұл атау өңдеуші құрылған оқиғалар мәні болып табылады, біздің жағдайда — OnClick оқиғалары үшін OK атты батырманы басу.

Басқа жағдайлар және басқа компоненттер өңдеуші үшін осыған ұқсас құрылады. Оқиғалар нағұрлым жан-жақты сәйкес компоненттерді меңгеру кезінде қарастырылады.

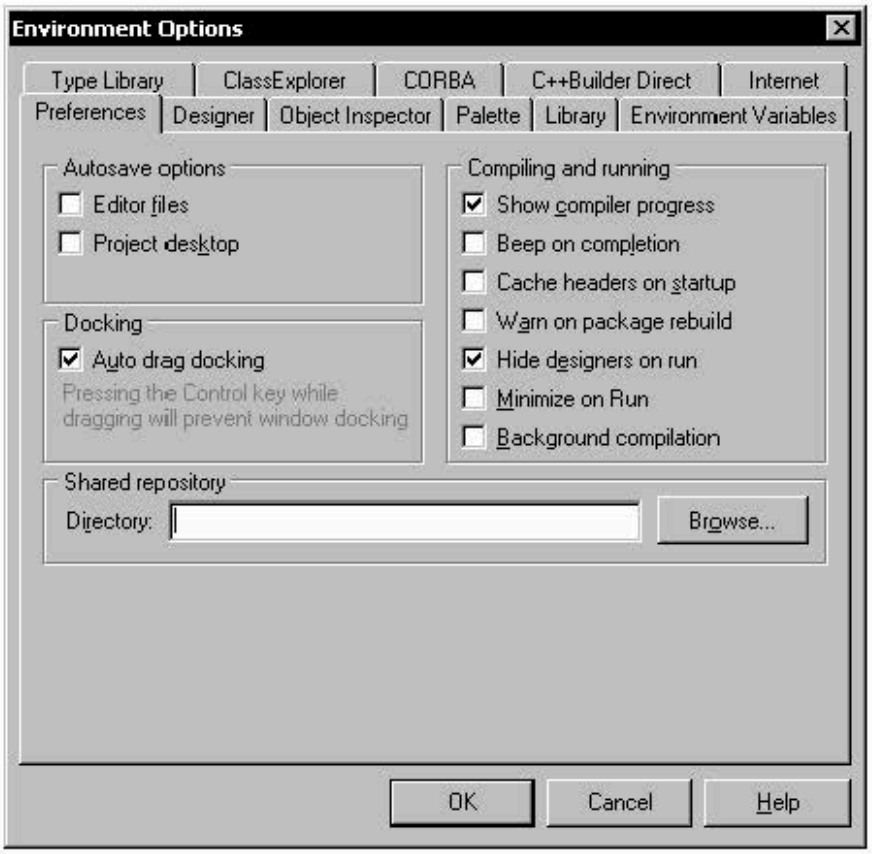
Процедуралар-өңдеушіні өшіру үшін бағдарлама құрушы өз бетімен енгізген кодты өшіру жеткілікті. Бұдан соң модульді сақтау немесе компиляциялау кезінде өңдеуші автоматты түрде жобаның барлық файлдарынан жойылады.

10.5. Бағдарламаны әзірлеудің біріктірілген ортасының құралдары

Өңдеудің интегралдық ортасы құрамында қосымшаны өңдеу үшін ыңғайлы және тиімді әртүрлі құралдар көп. Бұл бөлімде біз C++ Builder бағдарламаны әзірлеудің біріктірілген ортасының жалпы элементтерін қарастырамыз.

Ортаның басқару параметрлері

Пайдаланушы бағдарламаны әзірлеудің біріктірілген ортасын оның параметрлерін баптау арқылы басқару мүмкін. Параметрлерді орнату **Tools\Environment Options** (Құралдар\Орта параметрлері) командасымен шақырылатын **Environment Options** (Орта параметрлері) сұқбаттасу терезесінде орындалады. Барлық параметрлер жекелеген беттерде орналасқан топтар бойынша біріктірілген (10.10-сурет).



10.10-сурет. Өңдеу ортасы параметрлерін баптау терезесі

C++ Builder ортасы параметрлері үшін әрбір жоба конфигурация файлында (project configuration file) cfg кеңейтпесімен сақталады.

Кіріктірілген өңдеуші

Өңдеудің интегралдық ортасы кіріктірілген қосымшаны өңдеушіні іске қосады, ол айтарлықтай деңгейде қосымшадағы қателіктерді іздеу және жоюды жеңілдетеді. Өңдеуші құралдары **Run** мәзірі және **View\Debug Windows** (Көру\Терезе жүзеге асыру) ішкі мәзірінің командаларымен қосылады және келесі режимдерде жұмыс жасайды:

- көрсетілген жолдар кодына дейін орындау;
- қосымшаны қадаммен орындау;
- тоқтау нүктесіне дейін орындау (Breakpoint);
- тоқтау нүктесін қосу және ажырату;
- обьектілер мәндерін көру, мысалы, айнымалыларды көру;
- қосымшаның орындалуы кезінде обьекті мәндерін орнату.

Өңдеуші параметрлерін орнату **Tools** мәзірі командасымен шақырылатын **Debugger Options** (Өңдеуші параметрлері) сұқбаттасу терезесінде орындалады (10.11-сурет).

10.11-сурет. Өңдеуші параметрлер

Өңдеушіні қосу/ажыратуды үнсіз келісім бойынша орнатылған **Integrated debugging** (Біріктірілгенная орындау) белгішесімен басқарады және өңдеуші әрбір қосымшаға автоматты түрде қосылады.

Объектілерді сақтау қоймасы

C++ Builder жүйелері бір объектілерді қосымшаны өңдеу үшін бірнеше рет қолдануға мүмкіндік береді. Мұндай объектілерді сақтау үшін арнайы Объектілерді сақтау қоймасы немесе Репозиторий (Repository) Қолданылады.

Қосымшаларға жаңа объектіні сақтау қоймасынан таңдап, қосу үшін **New Items** (Жаңа элементтер) терезесінің **File\New\Other** (Файл\Жаңа\Басқа) командасы қолданылады. Бұл терезені Жоба менеджері аспаптар панелінің **New** батырмасын басу арқылы да ашуға болады .

Сақтау қоймасында әртүрлі объектілер орналасқан, мысалы, қосымша үлгісі, формалар, есеп берулер, сонымен қатар Форма шебері. Барлық объектілер жекелеген беттерде орналасқан топтарға біріктірілген, мысалы:

- **New** — қосымшаны құру кезінде қолданылатын кіріктірілген базалық объектілер;
- **ActiveX** —COM және OLE объектілері, ActiveX элементтері, ActiveX кітапханасы, белсенді серверлік беттер (ASP);
- **Project1** — құрылатын қосымшаның формалары;
- **Forms** — формалар;
- **Dialogs** — сұқбаттасу терезесі (стандартты, анықтамалық, құпия сөзді енгізу үшін);
- **Projects** — бір- және көпқұжатты қосымша жобалары;
- **Data Modules** —деректердің модулі;
- **Web Documents** — Web-құжаттар (HTML, XHTML, WML, XSL).

Project1 беттер атауы құрылатын жоба атауымен сәйкес келеді, ал беттің өзі үлгілер ретінде қосымшаның құрылған формаларын (алғашқыда бұл Form1 атаулы бір форма)

қамтиды. Жоба немесе формалар аттарын өзгерту кезінде сәйкесінше олардың Объектілерді сақтау қоймасындағы аттары өзгереді. Жобаға жаңа формалар қосу кезінде ол автоматты түрде жоба бетіне қосылады. Жобадан формаларды өшіру кезінде ол автоматты түрде Сақтау қоймасынан шығарылады.

Жобаға жаңа объектіні қосу үшін қажетті бетке өтіп, объектіні көрсету керек. **ОК** батырмасын басқан кезде объектіні кірістіру немесе қосу жүзеге асады. Объектілерді жобаға қосу әртүрлі әдістермен жүзеге асады.

❒ **Copy** — жобаға Сақтау қоймасындағы объект көшірмесі қосылады. Жобада бұл объектіні өзгертуге болады, бірақ барлық өзгерістер енгізу жоба шегінде *жергілікті* болады және Объектілерді сақтау қоймасында сақталатын түпнұсқаны өзгертпейді.

❒ **Inherit** — Сақтау қоймасындағы объектіден жаңа объектіге жіберіледі. Бағдарлама құрушы объектіге жаңа компоненттер қосуы мүмкін, сонымен қатар элементтердің қасиеттерін өзгертуге болады. Жобада бұл объектіні түрлендіру үшін оның құрамдас бөлімін (компонент) немесе атын өзгерту (Name қасиеті) мүмкін емес. Үнсіз келісім бойынша осылайша жобаға объектілер (әдетте формалар) қосылады.

❒ **Use** — жобаға Сақтау қоймасынан тікелей объектіні барлық өзінің файлдарымен іске қосады. Жобада бұл объектіні өзгерту кезінде Сақтау қоймасындағы объект, сонымен қатар осы объектіні қолданатын басқа жобалардағы объектілер де өзгереді.

Сақтау қоймасындағы объектілердің құрамын баптау қажеттілік кезінде **Tools\Repository** командасымен ашылатын **Object Repository** терезесінде орындалады.

Объектілерді сақтау қоймасында баптау процесінде беттер қосуға (**Add Page** батырмасы), өшіруге (**Delete Page** батырмасы) және беттердің атауын өзгертуге (**Rename Page** батырмасы), сонымен қатар редактрлеуге (**Edit Object** батырмасы) және объектілерді өшіруге (**Delete Object** батырмасы) болады.

Қосымшаның объектілері, формалар, фреймдер, деректердің модулі және код модулін **File\New** ішкі мәзірі арқылы жобаға кірістіруге (қосуға) болады, ол үшін Application, CLX Application, Data Module, Form, Frame және Unit командалары қолданылады.

10.6. Деректер қоры және олармен жұмыс жасау құралдары

Бұдан әрі C++ Builder жүйелерінде қолданылатын негізгі элементтер, реляциялық деректер қорын құраушылар (кестелер, кілттер және индекстер, кестелер арасындағы байланыстар) және кесте форматы қарастырылады. C++ Builder жүйелерінде деректер қорымен жұмыс жасау кезінде қолданылатын аспаптық құралдар, сонымен қатар компоненттер, деректер қоры үшін қосымшаны құру кезінде қолданылады.

Деректерге қатынас құру механизмдерінің сипаттамасы

Бір- және екіденгейлі C++ Builder қосымшасы деректер қорына жергілікті және қашықтықтан қатынас құруды келесі механизмдерді қолдану арқылы жүзеге асырады:

❒ BDE (Borland Database Engine — Borland фирмасының деректер қоры процессоры), деректер қорымен өзара әрекет жасау үшін дамыған API интерфейсін ұсынады (динамикалық кітапхана және драйверлер жинағы болып саналады);

❒ ADO (ActiveX Data Objects —ActiveX деректердің объектілері) ақпараттарға OLE DB көмегімен қатынас құруды жүзеге асырады (Object Linking and Embedding Data Base — деректер қорында объектілерді байланыстыру және енгізу);

❒ dbExpress деректер қорындағы ақпараттарға драйверлер жинағы көмегімен жылдам қатынас құруды қамтамасыз етеді;

❒ InterBase - InterBase деректер қорына қатынас құруды жүзеге асырады.

Деректер қорындағы ақпараттарға қатынас құру технологиясының нұсқасын таңдау, дайындау құрылған қосымшаның таралу қолайлылығын ескере отырып анықталады, сонымен қатар қосымшаның жадыда алатын орны ескеріледі.

Үшдеңгейлі C++ Builder қосымшасын DataSnap механизмы көмегімен құруға болады. Үшдеңгейлі (көпдеңгейлі) қосымшаны құрған кезде қолданылатын деректер қоры компоненттері **DataSnap** беттерінде және **Data Access** Компоненттер палитрасында орналасқан.

Деректерге қатынас құруды жүзеге асыратын динамикалық кітапхана және драйверлер жиынтығы ***BDE*** болып саналады. BDE процессоры ДҚ-мен жұмыс жасайтын C++ Builder қосымшасы орындалатын барлық компьютерлерде орнатылуы керек. Қосымшалар арқылы BDE деректер қорына сұраныс жібереді, ал кері қарай талап етілген деректерді алады.

ADO механизмі деректер қоры ақпараттарына қатынас құратын Microsoft фирмасының стандарты болып табылады. Бұл технологияны қолдану деректердің бапталатын драйверлерін қолдануды білдіреді. ADO технологиясы деректердің ақпарат көздеріндегі ақпараттарға қатынас құрудың әмбебап механизмін қамтамасыз етеді.

Қатынас құрудың ***dbExpress*** механизмі драйверлер жиынтығы, компоненттер, жалғастыру, транзакциялар, сұраныстар, деректердің жиыны және интерфейстерді құрайды. dbExpress технологиясы бойынша сервер мен деректер қорының өзара әрекетін қамтамасыз ету мамандандырылған драйверлерді қолдануға негізделген. Ол үшін SQL сұраныстар қолданады.

Деректер қоры кестелері

Ескере кетейік, реляциялық деректер қоры (ДҚ) өзарабайланысқан кестелерден тұрады. Әрбір кесте объектілер туралы бір типті ақпаратты қамтиды, ал барлық кестелердің жиынтығы біртұтас ДҚ береді.

ДҚ кестелері қатқыл дискісінде каталогта (бумада) орналасқан. Кестелер файлдарға сақталады және электрондық кестелерге (мысалы, Microsoft Excel кестелік процессоры) ұқсас. Бірақ, құжаттардан айырмашылықтары ДҚ кестелері *көппайдаланушылық* қатынас құру режимін сүйемелдейді, яғни, оларды бір уақытта бірнеше қосымша қолдана алады.

Бір кесте үшін деректер, индекстер, кілттер және т.с.с. қамтитын бірнеше файлдар құрылады. Олардың негізгісі деректермен файл болып табылады, бұл файлдың аты кесте атымен сәйкес келеді. Таңдау кезінде кестелер оның негізгі файлын таңдайды: dBase кестелері үшін бұл кеңейтпесі dbf файл, ал Paradox кестелері үшін - кеңейтпесі db файл. Қалған файлдардың аттары автоматты түрде тағайындалады — барлық файлдар кесте атына сәйкес келетін бірдей атауларға ие және сәйкес файлдың құрамын анықтайтын кеңейтпелері әртүрлі.

Кестені ұйымдастыру ерекшеліктері нақты ДҚБЖ-дан тәуелді. Оларды таңдау кезінде кестелер типін (форматын) ескеру керек, себебі олар тек қана ДҚ ұйымдастыруға ғана емес, осы ДҚ-мен жұмыс жасайтын қосымшаны тұрғызуға да әсер етеді. Бірақ, кестелердің айырмашылықтарына қарамастан, ДҚ құру және жұмыс жасаудың жалпы ережелері бар.

Кестелердің негізін оның өрісі құрайды, әрбір кестенің ең болмағанда бір өрісі болуы керек. Кестелермен келесі операцияларды орындауға болады: құру, құрылымына өзгерістер енгізу, атын өзгерту, өшіру.

Кестелер құрған кезде құрылымы және кестелер аты беріледі. Дискіге сақтау кезінде кестеге қатысты барлық қажетті файлдар құрылады. Олардың аты кесте атымен сәйкес. Кесте атын өзгерткен кезде кесте жаңа атауға ие болады, нәтижесінде барлық оның файлдары да жаңа атауға ие болады. Осы үшін ДҚ кестелерімен жұмыс жасауға

арналған сәйкес бағдарламалар (утилиттер) қолданылады, мысалы, Database Desktop немесе Data Pump.

Кестелерді өшіру кезінде барлық оның файлдары өшеді.

Кілттер және индекстер

Қарапайым кілт бір өрістен, ал құрама (күрделі) кілт бірнеше өрістен тұрады. Кілт тұрғызылған өрістер *кілттік* өрістер деп аталады. Кестеде тек қана бір кілт болуы мүмкін.

Кілт туралы ақпарат жеке файлда сақталады. Кілттің мәні белгілі бір ретпен орналасады. Әрбір кілттің мәні үшін бірегей сілтеме болады.

Кілттік файлда кестелердің сәйкес жазбаларына сілтемеден басқа кілттік өрістің мәні сақталады.

Индекс, кілт сияқты, кестелер өрісі бойынша құрылады, бірақ оның мәндері қайталануы мүмкін. Индекс тұрғызылған өріс *индекстік* деп аталады. Қарапайым индекс бір өрістен, ал құрама (күрделі) —бірнеше өрістен тұрады.

Бір кесте үшін бірнеше индекстерді құруға болады. Кілттік өрістер әдетте автоматты түрде индекстеледі. Paradox кестелерінде кілт негізгі (алғашқы) индекс болып табылады, оларға атау берілмейді. dBase кестелері үшін кілт құрылмайды және оның ролін индекстердің бірі орындайды.

Деректерге қатынас құру әдістері

C++ Builder ортасында кестелермен операциялар орындау кезінде келесі *деректерге қатынас құру әдістерінің* бірі қолданылады:

- навигациялық;
- реляциялық.

Навигациялық қатынас құру әдісінде кестелер жазбаларын өңдеу бөлек орындалады. Егер бірнеше жазбалармен жұмыс жасау қажет болса, онда олардың барлығын кезектесіп өңдейді.

Реляциялық қатынас құру әдісі бірден *жазбалар топтарын* өңдеуге негізделген. Өйткені реляциялық қатынас құру әдісі SQL-сұраныстарға негізделген.

Деректерге қатынас құру әдісін ДҚ-на қатынас құру құралдарынан тәуелсіз бағдарлама құрушы таңдайды.

Кестелер арасындағы байланыс

Кестелер арасындағы байланыстарды ДҚ құрған кезде және қосымшаның орындалуы кезінде тағайындауға болады. Екі немесе бірнеше кестені байланыстыруға болады. Реляциялық ДҚ-да, байланысқан кестеден басқа, ешбір кестемен байланыспаған ішінара кестелер болуы мүмкін.

Кестелерді байланыстыру үшін *байланыстар өрістері* қолданылады. Олар міндетті түрде индекстелген болуы керек. Индекстерді пайдалану ерекшеліктері байланыстырылатын кесте форматынан тәуелді.

Paradox кестесі үшін байланыстар ретінде негізгі кестелер өрісі өрістер кілтін, ал бағынышты кестелер үшін өрістер индексін қолданады. Сонымен қатар, бағынышты кестеде міндетті түрде кілт анықталуы керек. 11.3-суретте Paradox деректер қорының кестелері арасындағы байланыстар сызбасы көрсетілген.

10.12-сурет. Paradox деректер қорының кестелері арасындағы байланыстар сызбасы

Негізгі кестеде M_Code өрісі бойынша кілт анықталған. Бағынышты кестеде D_Num өрісі бойынша кілт және D_Code өрісі бойынша тұрғызылған индекс анықталған. Кестелер арасындағы байланыс D_Code және M_Code өрістері бойынша тағайындалады.

Жоғарыда байқалғандай, байланыстар өрістері индекстелген болуы керек, бірақ, бұл талап әрқашан міндетті болып табылмайды. Деректерге SQL тілі құралдарымен қатынас құрғанда кестелерді өзара байланыстыруға болады. Бірақ бұл жағдайда операциялардың орындалу жылдамдығы азаяды.

Кестелер арасындағы байланыс тәуелділік қатынасын анықтайды, онда бір кесте *негізгі* (түпнұсқалық немесе шебер — Master) болып табылады, ал екіншісі — *бағынышты* (ішкі немесе бөлшек — Detail). Байланыстың (қатынас) өзі "негізгі-бағынышты" байланыс деп аталады.

Негізгі кестелерден жазбаларды өшіру кезінде бағынышты кестеде сәйкес жазбалардың бар болуын тексеру керек. Егер ондай жазба бар болса, онда оларды да өшіру қажет немесе, керісінше, екі кестеден жазбаларды өшіруге тыйым салу қажет.

Кесте форматы

C++ Builder ортасы кестесінің өз форматы жоқ, бірақ dBase және Paradox кестелерінің форматын сүйемелдейді. Осы кестелердің әрқасйсысының өз ерекшеліктері бар.

dBase кестелері жеткілікті түрде қарапайым болады және дискіге сақтау үшін салыстырмалы түрде аз физикалық файлдар қолданады. Файлдардың кеңейтпесі бойынша деректер құрамын анықтауға болады:

- dbf — деректер кестесі.
- dbt — үлкен көлемді екілік объектілер деректері немесе BLOB-деректер (Binary Large OBject). Оларға екілік, Memo- және OLE-өрістер жатады.
- mdx — сүйемелдейтін индекстер.
- ndx — dBase форматымен тікелей сүйемелденбейтін индекстер. Мұндай индекстерді қолданған кезде бағдарлама құрушы оларды өзі өңдеуі керек .

dBase кестесінде өрістер аты әріптер мен сандардан тұрады, әдетте әріптен басталады. Ең үлкен ұзындығы 10 символды құрайды. Атында арнайы символдар мен бос орында қолдауға болмайды.

Paradox кестелері жеткілікті түрде дамыған және ДҚ құру үшін қолайлы болады. Олардың *артықшылықтары*: әртүрлі ақпараттық-бағдарламалық қамтамасыз етілуі; деректер тұтастығын сүйемелдеу; енгізілген деректерді тексеру; кестені құпия сөзбен қорғауды сүйемелдеу. Сондықтан, Paradox кестелері жиі қолданылады.

10.1-кестеде Paradox 7 кестелері үшін өріс типтері көрсетілген.

10.1-кесте. Paradox 7 кестелері үшін өріс типтері

Тип	**Белгіле-улер**	**Сипаттау мәні**
Alpha	A	Символдар жолы. Ұзындығы 255 символдан артық емес
Number	N	Диапазоны -10^{307} ... 10^{308} бүтін сан.
Money	$	Қаржылық қосынды. Онда қаржылық белгі көрсетіледі
Short	S	Бүтін сан. Диапазоны –32 768 ... 32 767
LongInteger	I	Бүтін сан. Диапазоны –2 147 483 648 ... 2 147 483 647
BCD	#	Екілік-ондық форматтағы сан
Date	D	Мерзімі. Диапазоны 01.01.9999. ... 31.12.9999
Time	T	Уақыт
Timestamp	@	Мерзім және уақыт
Memo	M	Символдар жолы. Ұзындығы шектелмеген. Алғашқы 240 символдар кестелер файлында, қалғандары кеңейтуі mb файлда сақталады
Formatted Memo	F	Символдар жолы. Memo типінен айырмашылығы - жол форматталған мәтіннен тұруы мүмкін
Graphic	G	Графикалық бейне. Форматтары BMP, PCX, TIFF, GIF және EPS.
OLE	O	OLE технологиясын сүйемелдейтін деректер форматы
Logical	L	Логикалық мән. Мәні - true (ақиқат) және false (жалған).
Autoincrement	+	Автоинкрементті өріс. Олардың мәні оқу үшін қолжетімді және әдетте кілттік өрістер ретінде қолданылады
Binary	B	Байттар тізбегі. Ұзындығы шектелмеген. Байты содержат еріктіое двоичное мәні. Алғашқы 240 байттар кестелер файлында, қалғандары кеңейтуі mb файлда сақталады
Bytes	Y	Байттар тізбегі. Ұзындығы 255 байттан көп емес

Paradox кестелер файлының кеңейтпелері:

- db — деректермен кесте;
- mb — BLOB-деректер;
- px — негізгі индекс (кілт);
- xg* және yg* — екінші индекстер;
- val — деректердің және сілтемелердің тұтастығын тексеру параметрлері;
- tv және fam —Database Desktop бағдарламасында кестелерді шығару форматы.

Көрсетілген файлдар қажетті кезде құрылады.

Құрал-саймандар

ДҚ-мен жұмыс жасауға арналған C++ Builder құралдарын екіге бөлуге болады:

- құрал-саймандар
- компоненттер

Құрал-саймандарға ДҚ қызметін жүзеге асыратын арнайы бағдарламалар және дестелер жатады.

Компоненттер ДҚ-мен операцияларды орындауға арналған қосымшаны құруды қамтиды.

C++ Builder келесі құрал-саймандар жиынын ұсынады:

- BDE Administrator — BDE әртүрлі параметрлерін баптау, деректер қоры драйверлерін баптау, ODBC драйверлерін құру және өшіру, псевдонимдер құру және қызмет көрсету.

❒ Database Desktop — бағдарлама құру және кестені өңдеу, SQL-сұраныстар және QBE сұраныстар.

❒ SQL Explorer — ДҚ және деректердің сөздігін көру және редактрлеуге мүмкіндік беретін ДҚ сілтеуіші.

❒ SQL Builder — SQL-сұраныстарды визуалды құрылымдау бағдарламасы.

❒ SQL Monitor — шеткері ДҚ-на SQL-сұраныстардың орындалу ретін бақылау бағдарламасы.

❒ Data Pump — ДҚ арасында деректердің орнын аустыруға арналған бағдарлама.

❒ IBConsole — шеткері ДҚ басқару бағдарламасы.

❒ InterBase Server Manager — InterBase серверін жүктеу бағдарламасы.

❒ SQL Links — қосымшаның шеткері ДҚБЖ-ға, Microsoft SQL Server немесе Oracle сияқты, қатынас құру драйверлері.

❒ dbExpress —деректер қорына қатынасқа құру SQL (InterBase, DB2, Oracle, MSSQL, MySQL) драйверлерінің жиыны.

❒ InterBase Server — InterBase серверінің клиенттік және серверлік бөлімдері.

Деректер қоры қосымшасының компоненттері

C++ Builder элементтерін басқару сияқты, ДҚ-мен байланысты компоненттерді визуалды және визуалды емес деп бөледі. *Визуалды емес* компоненттер деректерге қатынас құруды ұйымдастыруға арналған. Олар ДҚ кестесі деректері мен визуалды компоненттер арасындағы аралық бөлік.

Визуалды компоненттер қосымшаның интерфейстік бөлімдерін құру үшін қолданылады. Олардың көмегімен пайдаланушы деректерді көру немесе редактірлеу сияқты операцияларды орындауы мүмкін. Визуалды компоненттерді *деректерге сезімтал элементтер* деп те атайды.

ДҚ-мен жұмыс жасау үшін қолданылатын компоненттер **Data Access**, **Data Controls**, **dbExpress**, **DataSnap**, **BDE**, **ADO**, **InterBase**, **Decision Cube**, **QReport** және **InterBase Admin** Компоненттер палитрасы беттерінде орналасқан. Қайсібір компоненттер арнайы шеткері ДҚ-мен "клиент-сервер" архитектурасында жұмыс жасауға арналған. Маңызды компоненттерді қарастырайық.

Data Access (10.14-сурет) бетінде деректерге қатынас құруды ұйымдастыруға арналған визуалды емес компоненттер орналасқан, мысалы:

❒ DataSource (деректердің түпнұсқасы);

❒ ClientDataSet (деректердің клиенттік жинағы);

❒ DataSetProvider (провайдер деректерінің жинағы);

❒ XML Transform (XML құжатын деректердің дестеміне және кері қарай түрлендіргіштер).

❒

10.14-сурет. Data Access беті

10.15-сурет. Data Controls беті

Data Controls бетінде (10.14-сурет) орналасқан, деректерді басқаруға арналған визуалды компоненттер:

- DBGrid (торкөз, немесе кесте);
- DBNavigator (навигациялық интерфейс);
- DBText (жазу);
- DBEdit (біржолдық редактор немесе өрісті өңдеу);
- DBMemo (көпжолдық редактор немесе өңдеу панелі);
- DBImage (графикалық бейнелеу);
- DBListBox (тізім);
- DBComboBox (біріктірілген тізім);
- DBCheckBox (белгіше);
- DBRadioGroup (ауыстырып-қосқыштар тобы);
- DBLookupListBox (деректердің басқа жиынының өрісі бойынша қалыптасқан тізім);
- DBLookupComboBox (деректердің басқа жиынының өрісі бойынша қалыптасқан біріктірілген тізім);
- DBRichEdit (толық қызметтік мәтіндік редактор, немесе өңдеу өрісі);
- DBCtrlGrid (модификацияланған торкөз);
- DBChart (диаграмма).

BDE (10.16-сурет) беті BDE қолдану арқылы деректерді басқаруға арналған компоненттерді қамтиды:

- Table (ДҚ кестесіне негізделген деректердің жиыны);
- Query (SQL-сұранысқа негізделген деректердің жиыны);
- StoredProc (серверден сақталатын процедураларды шақыру);
- DataBase (ДҚ-мен байланыстыру);
- Session (ДҚ-мен жұмыс жасаудың ағымдық сеансы);
- BatchMove (жазбалар тобымен операциялар орындау);
- UpdateSQL (SQL-сұранысқа немесе сақталатын процедураларға негізделген деректер жиынының өзгерістері);
- NestedTable (кіріктірілген кесте)
- BDEClientDataset (деректердің клиенттік жинағы)

ADO бетінде (10.17-сурет) ADO (Active Data Objects) технологиясын қолдану арқылы деректерді басқаруға арналған компоненттер орналасқан:

- ADOConnection (байланыстыру);
- ADOCommand (команда);
- ADODataSet (деректердің жиыны);
- ADOTable (Table деректердің жиыны);
- ADOQuery (Query деректердің жиыны);
- ADOStoredProc (серверден сақталатын процедураларды шақыру);
- RDSConnection (RDS байланыс).

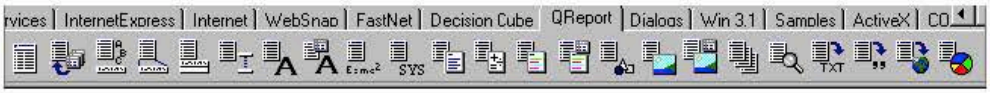
10.16-сурет. BDE беті

10.17-сурет. ADO беті

QReport бетінде (11.18-сурет) есеп беруді жүзеге асыруға арналған компоненттер орналасқан:

- QuickRep (есеп беру);

❒ QRSubDetail ("басты-бағынышты" қатынасымен байланысқан кестенің есеп беру жолағы);
❒ QRStringsBand (есеп берудің жолдық жолағы);
❒ QRBand (есеп беру жолағы);
❒ QRChildBand (есеп берудің ішкі жолағы);
❒ QRGroup (тобы);
❒ QRLabel (жазу);
❒ QRDBText (деректер жиынының мәтіндік өрісі);
❒ QRExpr (өрнек);
❒ QRSysData (жүйелік ақпарат);
❒ QRMemo (көпжолдық мәтін);
❒ QRExprMemo (көпжолдық өрнек);
❒ QRRichText (форматталған мәтін);
❒ QRDBRichText (деректер жиыны өрістерінің форматталған мәтіні);
❒ QRShape (геометриялық фигура);
❒ QRImage (графикалық тұрғыдан бейне);
❒ QRDBImage (графикалық тұрғыдан деректер жиыны өрістерінің бейнесі);
❒ QRCompositeReport (құрама есеп беру);
❒ QRPreview (есеп беруді көру терезесі);
❒ QRTextFilter (мәтіндік сүзгі);
❒ QRCSVFilter (CSV-сүзгі);
❒ QRHTMLFilter (HTML-сүзгі);
❒ QRChart (диаграмма).



10.18-сурет. QReport беті

Көптеген компоненттер аты префикстерді қамтиды, мысалы, DB, IB немесе QR. DB префиксі визуалды компонент деректермен байланысты екенін білдіреді. Ондай компоненттер формада орналасады және пайдаланушының деректерді басқаруына арналған. QR префиксі компонент QReport есеп беруді тұрғызу үшін қолданылатынын білдіреді.

Деректер қорымен жұмыс жасау кезінде шығарып тастау

C++ Builder деректер қорымен жұмыс жасау кезінде шығарып тастау үшін келесі қосымша кластарды ұсынады:

❒ EDatabaseError — ДҚ қатесі; олар:

• EDBEngineError —BDE қатесі (жергілікті ДҚ және "файл-сервер" архитектурасымен желілік ДҚ үшін);

• EDBClient — клиент қосымшаларындағы қателіктер ("клиент-сервер " архитектурасымен желілік ДҚ үшін); қате коды BDE, ADO, dbExpress немесе басқа деректерге қатынас құру механизмдеріне қайтып келеді;

• EUpdateError — жазбаларды жаңарту кезінде пайда болатын қателер;

❒ EDBEditError — өріске енгізілген мән өріс типіне сәйкес.

EDatabaseError класы қателерді өңдеуге арналған. Бұл класс тікелей Exception класынан жүзеге асады.

EDBEngineError класы жергілікті компьютерде BDE деректер қоры процессорымен жұмыс жасау кезінде пайда болған қателерді өңдеуге арналған. TDBError класы келесі қасиеттермен анықталатын ақпаратты қамтиды:

❒ string типті Message (қатені сипаттайтын мәтіндік хабарлама);

❒ DBIResult типті ErrorCode (қате коды), DBIResult типі Word типімен сәйкес келеді;
❒ Byte типті Category (Шығарып тастау деңгейі);
❒ Byte типті SubCode (Шығарып тастау тобы);
❒ Longint типті NativeError (сервермен қайтарылатын қате коды).

10.7. Деректер қоры кестесін құру

Ақпараттық жүйелер құру екі негізгі кезеңді іске қосады: ДҚ құру және қосымшаны құру. Қарапайым ақпараттық жүйелер құру кезінде C++ Builder ортасының ДҚ-мен жұмыс мүмкіндіктерін қарастырайық. Барлық қажетті операциялар Database Desktop бағдарламалары, Формалар конструкторы және Объектілер инспекторы көмегімен орындалады.

Қарапайым жағдайда ДҚ бір кестеден тұрады. Егер кестелер бар болса, онда бірінші кезең орындалмайды. Кестелермен жұмыс жасау үшін ДҚ жобалау кезінде қосымшаны тиімді қолдану бағдарламасы Database Desktop келесі мүмкіндіктерді береді:
❒ кестелер құру;
❒ құрылымын өзгерту;
❒ жазбаны редактірлеу.

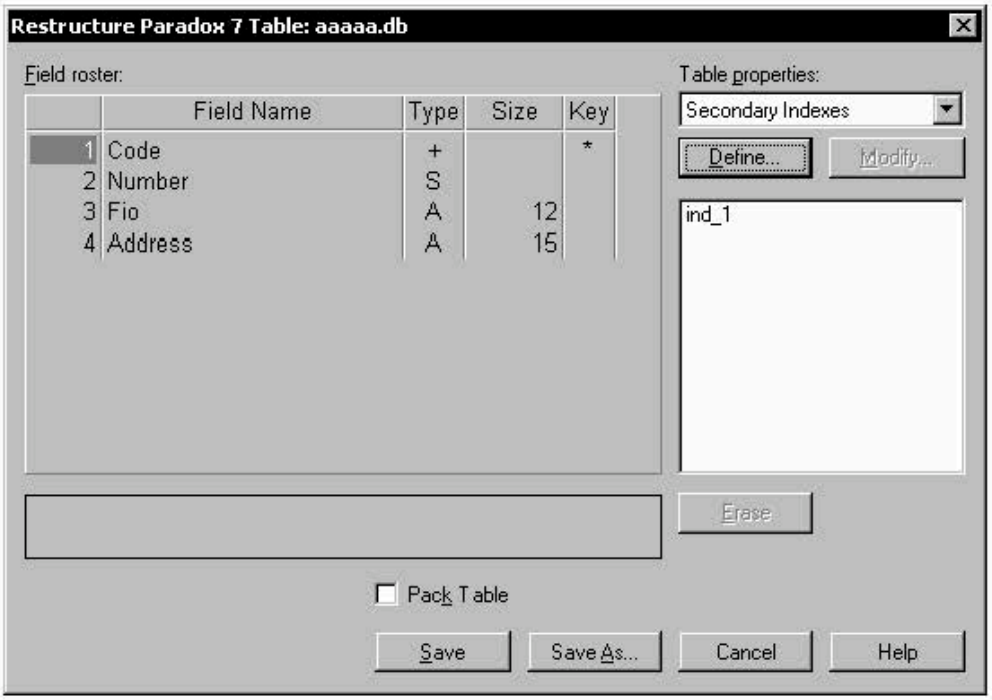
Сонымен қатар, Database Desktop көмегімен ДҚ-мен басқа да әрекеттер (құру, редактірлеу және визуалды орындау және SQL-сұраныстар, псевдонимдермен операциялар) орындауға болады.

Жаңа кестелер құру процессі **File\New\Table** (Файл\Жаңа\Кесте) командасын шақырумен басталады және интербелсенді режимде жүзеге асады. Сонымен қатар, бағдарлама құрушы міндетті:
❒ кестелер форматын (типін) таңдауға;
❒ кестелер құрылымын беруге.

Create Table (Кестелер құру) терезесінде жаңа кестелер құру кезінде ең алдымен, оның форматын таңдайды. Үнсіз келісім бойынша Paradox 7 кестелер форматы ұсынылады. Кестенің басқа форматтарының, мысалы dBase IV, кестелер құру әрекеті бойынша, практикалық тұрғыдан, айырмашылығы жоқ.

Кестелер форматын таңдағаннан кейін кестелер құрылымын анықтау терезесі (10.19-сурет) пайда болады, онда келесі әрекеттер орындалады:
❒ өрісті сипаттау;
❒ кілтті анықтау;
❒ индекстерді көрсету;
❒ өрістің мәндеріне шектеулер қою;
❒ сілтемелік тұтастық шарттарын (шектеулерді) анықтау;
❒ құпия сөзді енгізу;
❒ тілдің драйверлерін беру;
❒ кестелер үшін мәндерді таңдау.

Жоғарыдағы тізімдегі тек қана бірінші әрекет міндетті болып табылады, яғни әрбір кестеде ең болмағанда бір өріс болуы керек. Қалған әрекеттер қажеттілік кезінде орындалады.



10.19-сурет. Кестелер құрылымын анықтау

Кестелер құрылымын анықтағаннан кейін оны **Save As** батырмасын басу арқылы сақтау қажет. Нәтижесінде жаңа кесте дискіге жазылады.

Өрісті сипаттау

Кестелер құрылымын анықтау терезесінің орталық бөлімі **Field roster** (өрістер тізімі) тізімі болып табылады, онда кестелердің өрісі көрсетіледі. Әрбір өрістер үшін беріледі:

- өрістер аты — **Field Name** бағанасында;
- өрістер типі — **Type** бағанасында;
- өрістер өлшемі —**Size** бағанасында.

Өрістер аты кесте форматы үшін тағайындалған ережелер бойынша енгізіледі.

Өрістер типін сәйкес символды енгізу арқылы анықтауға болады. Тізім өрістің барлық типтерін қамтиды.

Өрістер өлшемі қажетті жағдайда ғана беріледі. Жолдық тип өлшемі өріске енгізуге болатын символдардың ең үлкен санын анықтайды. Кілт оның өрісін көрсетіп құрылады.

Кестелер құрылымын анықтау бойынша басқа әрекеттердің орындалуы үшін келесі пункттерді қамтитын біріктірілген тізім **Table properties** (кестелер қасиеттері) қолданылады:

- **Validity Checks** (өріске мәндерді енгізу дұрыстығын тексеру) —үнсіз келісім бойынша таңдалады;
- **Table Lookup** (кестені таңдау);
- **Secondary Indexes** (екінші индекс);
- **Referential Integrity** (сілтемелік тұтастық);
- **Password Security** (құпия сөздер);
- **Table Language** (кестелер тілі, тілдік драйвер);
- **Dependent Tables** (бағынышты кестелер).

dBase үшін кестелер тек қана **Indexes** және **Table Language** пунктерін қамтиды.

Индекстерді анықтау

Индекстерді анықтау келесілерді анықтауды қамтиды:

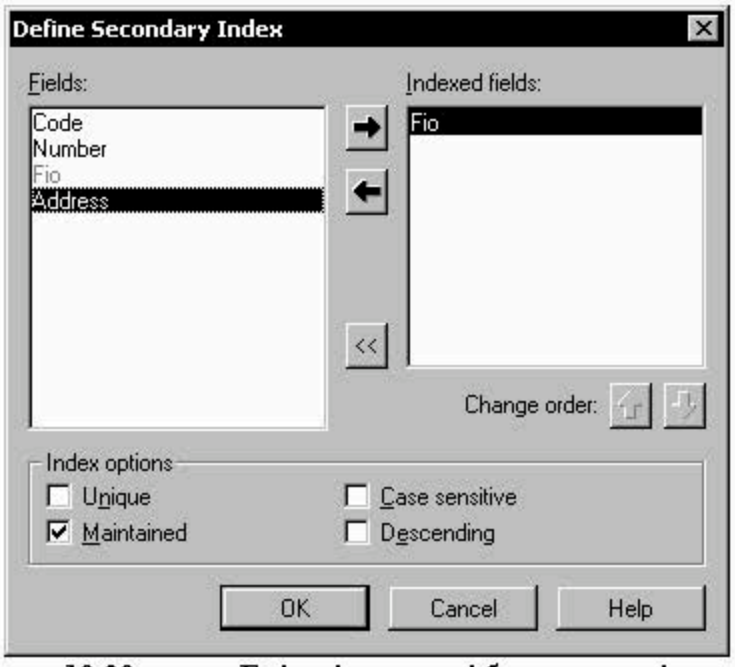
- Өрісің құрамы;
- Параметрлер;
- Аттары.

Бұл элементтер операцияларды құру, өзгерістер енгізу және индексті өшіру кезінде орнатылады немесе өзгереді.

Индекстерді анықтаумен байланысты операцияларды орындау үшін **Table properties** (Кестелер қасиеттері) тізімінен **Secondary Indexes** (Екінші индекстер) пунктін таңдау керек, сонда **Define** (Анықтау) және **Modify** (Өзгерту) батырмасы, индекстердің тізімі және **Erase** (Өшіру) батырмасы шығады.

Жаңа индекс құру **Define** батырмасын басудан басталады. Ол **Define Secondary Index** (екінші индексті беру) терезесін ашады, онда индекс құрамы және параметрлері беріледі (11.20-сурет).

Fields терезесінің тізімінде кестелердің барлық өрістерінің аттары беріледі. **Indexed fields** (Индекстік өрістер) тізімінде құрылатын индекс құрамын қамтитын өрістер болады. Индекс құрамын қамтуына болмайтын өріс аттары сол жақтағы тізімде сұр түспен бөліп көрсетіледі.



10.20-сурет. Екінші индексті беру терезесі

Индекте өрістер ретін өзгерту вертикаль бағыттауыштар бейнеленген батырмалар көмегімен орындалады, оларды **Change order** (Өзгерту реті) деп атайды. Өрістердің орнын ауыстыру үшін оны (оларды) ерекшелеп, аталған батырманы басу керек.

Терезенің төменгі бөлігінде орналасқан белгішелер келесі индекс параметрлерін береді:

- **Unique** — индекс бірегей мәндерді талап етеді;
- **Maintained** — индекске автоматты түрде қызмет көрсетіледі;
- **Case sensitive** — жолдық типті өріс үшін символдар регистрі ескеріледі;
- **Descending** — мәндерді сұрыптау кему бойынша орындалады.

Себебі dBase кестелерінде кілт жоқ, олар үшін **Unique** параметрін қолдану бағдарламалауды пайдаланбай, физикалық деңгейде жазбалардың бірегейлігін қамтамасыз етудің жалғыз мүмкіндігі болып табылады. **OK** батырмасын басқаннан кейін

Save Index As терезесі пайда болады, онда индекс атын көрсету керек. **ОК** батырмасын қайтара басқаннан кейін қалыптасқан индекс кестеге қосылады, және оның аты индекстердің тізімінде пайда болады.

Құрылған индексті өзгертуге болады. Индексті өзгерту практикалық тұрғыдан оны құрудан ерекшеленбейді.

Индексті өшіру үшін оны индекстердің тізімінен ерекшелеп, **Erase** батырмасын басу керек. Нәтижесінде индекс ескертпе хабарламаларсыз жойылады.

Modify және **Erase** батырмалары егер индекс тізімнен таңдалған болса ғана қолжетімді.

Өрістің мәндеріне шектеулер қою

Өрістің мәндеріне шектеулер қою үшін келесілерді көрсету керек:

- ❒ міндетті түрде енгізілетін мәнге талаптар;
- ❒ ең кіші мәні;
- ❒ ең үлкен мәні;
- ❒ үнсіз келісім бойынша мәні;
- ❒ енгізу маскасы.

Өрістің мәндеріне шектеулер қоюмен байланысты операцияларды орындау үшін **Table properties** біріктірілген тізімнен **Validity Checks** (мәндерді тексеру) пунктін таңдау керек (10.19-сурет).

Сілтемелік тұтастық

Сілтемелік тұтастық түсінігіне кестелердің өзара әрекетінің келесі байланыстары жатады:

- ❒ егер ол үшін бағынышты кесте жазбасы бар болса, онда өріс байланыстарын өзгертуге немесе негізгі кестелер жазбасын өшіруге тыйым салынады;
- ❒ негізгі кестеден жазбаны өшіру кезінде бағынышты кестедегі сәйкес жазбалар автоматты түрде өшіріледі.

Сілтемелік тұтастықпен байланысты операцияларды орындау үшін **Table properties** біріктірілген тізімнен **Referential Integrity** пунктін таңдау керек (11.19-сурет).

Құпия сөзді енгізу

Құпия сөзді енгізу пайдаланушылардың (қосымшаның) кестеге қатынас құру құқығын тағайындауға мүмкіндік береді. Егер кестелері үшін құпия сөз тағайындалса, онда ол автоматты түрде әрбір кестені ашу алдында сұралады. Құпия сөз физикалық деңгейде жүзеге асады және оның әрекеті кестеге қатынас құруды жүзеге асыратын барлық бағдарламалар үшін орынды.

Құпия сөзді енгізумен байланысты операцияларды орындау үшін **Table properties** біріктірілген тізімнен **Password Security** пунктін таңдау керек (10.19-сурет).

Тілдің драйверлерін беру

Тілдің драйверлерін беру үшін **Table properties** кестелер құрылымын анықтау терезесінің біріктірілген тізімінен **Table Language** (кестелер тілі) пунктін таңдау керек (10.19-сурет).

Кестелер құрылымын өзгерту

Кестелер құрылымын Database Desktop бағдарламалары терезесінен бастапқы кестелерді таңдағаннан кейін, **Table\Restructure** командасымен өзгертуге болады. Нәтижесінде кестелер құрылымын анықтау терезесі ашылады және келесі әрекеттердің кестелер құрған кездегі әрекеттерден айырмашылығы жоқ.

Егер тек қана кестелер құрылымымен танысу керек болса, онда **Table\Info Structure** командасы орындалады.

10.8. BDE қосымшасын құру

BDE қатынас құру механизмін пайдаланатын және ДҚ кестелерінің жазбалары бойынша орын ауыстыруға, өрістерді көруге және редактрлеуге, кестелерден жазбаны өшіруге, сонымен қатар жаңа жазбалар қосуға мүмкіндік беретін қосымшаны құру мысалын қарастырайық. Қосымшаны құрған кездегі негізгі міндеті формалар құрастыру, қарапайым жағдайда — бір форма болып табылады.

Жобалау кезеңіндегі қосымша формаларының түрі 10.21-суретте көрсетілген, мұндағы формада Table1, DataSource1, DBGrid1 және DBNavigator1 компоненттері орналасқан.

Table1 компоненті ДҚ кестесімен өзара әрекетті қамтамасыз етеді. Қажетті кестемен байланыстыру үшін ДҚ-на қатынасу жолын көрсететін DataBaseName қасиеттерінің сәйкес мәнін және кестелер атын көрсететін TableName қасиеттерін тағайындау керек. Бұдан соң деректердің Active қасиеттеріне true мәні тағайындалуы керек.

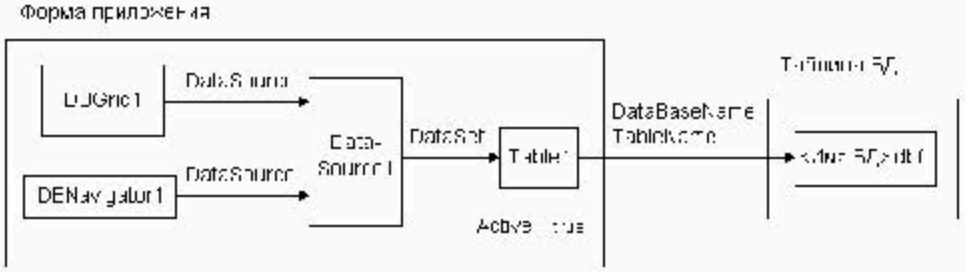
10.21-сурет. ДҚ-мен жұмыс жасау үшін қосымшаның формасы

Кестелер атын TableName қасиеттер мәнінің өрістері тізімінен таңдаған дұрыс. Егер ДҚ-на (DataBaseName қасиеттері) қатынас жолы дұрыс берілсе, онда бұл тізімде барлық қолжетімді кестелердің негізгі файлдары бейнеленеді.

DataSource1 компоненті нақты ДҚ кестесімен байланысқан Table1 компоненті мен пайдаланушының осы кестелерге өзара әрекетін жүзеге асыратын визуалды DBGrid1 және DBNavigator1 компоненттері арасындағы аралық звено болып табылады. DataSource1 компонентімен байланысты Table1 компонентінде DataSet қасиеттері көрсетіледі.

DBGrid1 компоненті ДҚ кестесінің құрамын торкөз түрінде бейнелейді, онда бағаналар өрістермен, ал жолдар — кесте жазбаларғымен сәйкес келеді. Үнсіз келісім бойынша пайдаланушы деректерді көруі және редактрлеуі мүмкін. DBNavigator1 компоненттері пайдаланушыға кестеде орынын ауыстыруға, жазбаны редактрлеуге, кірістіруге және өшіруге мүмкіндік береді. DBGrid1 және DBNavigator1 компоненттері DataSource қасиеттері арқылы өзінің түпнұсқасы — DataSource1 компонентімен байланысады.

Қосымша және ДҚ кестелері компоненттерінің өзарабайланысы және қолданылатын компоненттер қасиеттері 10.22-суретте көрсетілген.



10.22-сурет. Қосымшаның компоненттері мен ДҚ кестелерінің өзарабайланысы

Қосымшаларды құру кезінде Объектілер инспекторы көмегімен компоненттер қасиеттерінің барлық мәнін беруге болады. Сонымен қатар, талап етілген мәнді өріске тікелей енгізеді немесе ашылған тізімнен таңдайды. Соңғы жағдайда қосымшалар тышқанның көмегімен құрылады. 10.2-кестеде ДҚ кестесімен жұмыс жасау үшін қолданылатын компоненттер, оларды негізгі қасиеттері және осы қасиеттердің мәндері келтірілген.

10.2-кесте. Компоненттер қасиеттерінің мәні

Компонент	**Қасиеттері**	**Мәні**
Table1	DataBaseName TableName Active	dbdemos Animals.dbf True
DataSource1	DataSet	Table1
DBGrid1	DataSource	DataSource1
DBNavigator1	DataSource	DataSource1

Формалар құру процесін автоматтандыру үшін **Database Form Wizard** (деректер қоры формасы шебері) қолдануға болады. Бұл Шебер Объектілерді сақтау қоймасының **Business** бетінде орналасқан.

Шебер бөлек кестелермен және байланысқан кестелермен жұмыс жасау үшін формалар құруға мүмкіндік береді, сонымен қатар Table немесе Query деректердің жиынын қолдануға болады.

10.9. Есеп берумен жұмыс

Есеп беру құжатты басып шығару болып саналады, олар ДҚ-на сұраныстың орындалу нәтижесінде алынатын деректерді қамтиды. Есеп берудің түрлері: қарапайым; "басты-бағынышты" қатынасымен байланысқан кесте үшін; бірнеше әртүрлі есеп берулерді біріктіретін құрама. C++ Builder ортасында есеп беру үшін QuickReport есеп берулер генераторы қолданылады.

Есеп беру компонентері

Есеп беруді құруға арналған компоненттер Компоненттер палитрасының **QReport** бетінде орналасқан. Көптеген есеп беру компоненттері визуалды болады. Олардың **Standard**, **Additional** және **Data Controls** беттерінің ұқсас компоненттерінен айырмашылығы аз. Мысалы, QRImage компонентіне Image және DBImage компоненттері сәйкес келеді.

Есеп берудің негізгі элементі QuickRep ***есеп беру компоненті*** болып табылады, онда басқа компоненттер орналасқан. QuickRep компоненті әдетте есеп беруге арналған жекелеген формада орналасады. Үнсіз келісім бойынша оған QuickRep1 атауы беріледі. Егер формада басқа есеп берулер (әдетте олай болмайды) орналасса, олар QuickRep2 және т.б. атауына ие болады.

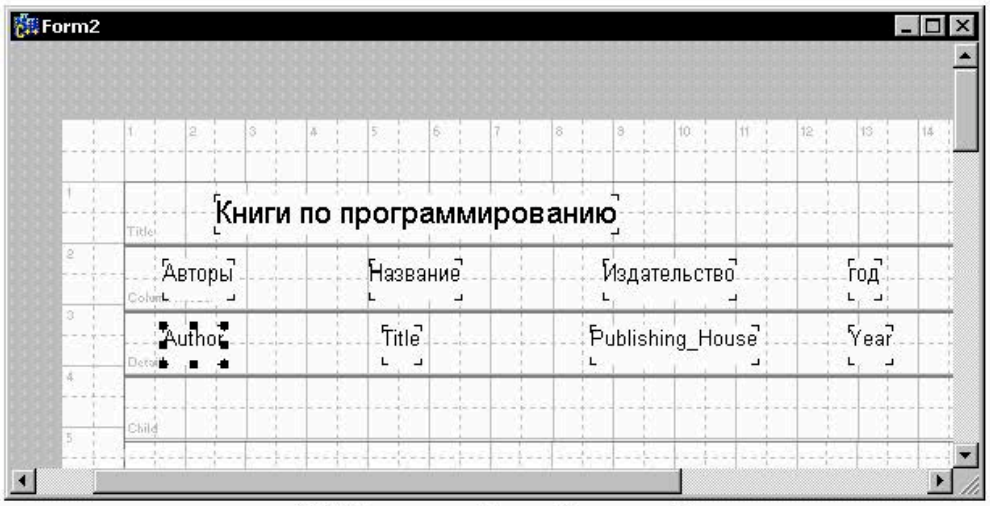
QuickRep компоненті DataSet қасиеттері көмегімен есеп берулер құрылатын Table немесе Query деректердің жиынымен байланыстырады. Сонымен қатар, Query деректердің жиыны әртүрлі кестелерден таңдалған жазбалардан тұруы мүмкін.

Мысалы. Есеп беруді деректердің жиынымен байланыстыру.

```
void __fastcall TForm2::FormCreate(TObject *Sender)
{
QuickRep1->DataSet=Form1->Table1;
}
```

Form2 формасында орналасқан QuickRep1 есеп беруі өңдеу кезінде осы форманың OnCreate оқиғаларын Form1 формасында орналасқан Table1 деректердің жиынымен

байланыстырады. Есеп беру жекелеген жолақтардан тұрады. Есеп беру элементі жолақ болып саналады. Әрбір элемент өз орнында орналасады және сәйкес есеп беру компоненттерін бейнелеуге және деректерді шығаруға арналған. Есеп беру түрі 10.23-суретте келтірілген.



10.23-сурет. Есеп беру түрі

Қарапайым есеп беру кезінде келесі жолақтар болуы мүмкін:

- HasPageHeader – жоғарғы колонтитул;
- HasTitle – есеп беру тақырыбы;
- HasColumnHeader – бағаналар тақырыптары;
- HasDetail – деректер облысы;
- HasSummary – есеп беру қорытындысы;
- HasPageFooter – төменгі колонтитул.

Есеп беруа беттері параметрлері TQRPage типті Page қасиеттерін анықтайды, онда келесі сипаттамаларды анықтауға болады:

- PaperSize – беттер форматы (үнсіз келісім бойынша A4);
- TopMargin, BottomMargin, LeftMargin және RightMargin – өрістің жоғарғы, төменгі, сол және оң жақ өлшемдері.

Мысалы. Бастапқы көру және есеп беруді баспаға шығару процедурасы.

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
TBookmark bm;
bm=Table1->GetBookmark();
// есеп беруді көру
Form2->QuickRep1->Preview();
// есеп беруді баспаға шығару
Form2->QuickRep1->Print();
Table1->GotoBookmark(bm);
Table1->FreeBookmark(bm);
}
```

bm қосымшасы ағымдағы жазба сілтемесін есте сақтау және қалпына келтіру үшін қолданылады. Есеп беруді баспаға шығару орындалатын Form1 формалар модулінің бастапқы коды файлында есеп беру компоненті орналасқан Form2 тақырыптық формалар модулі файлына қатынас құру үшін препроцессор директивасын #include "Unit2.h" көрсету керек.

Есеп беруді баспаға шығару кезінде TQRBeforePrintEvent типті BeforePrint және AfterPrint оқиғалары орындалады. Есеп беруді баспаға шығару алдында көру үшін көру

терезесін жүктейтін Preview әдісі қолданылады. Бұл терезеде келесілерді орындауға болады: есеп беруді әртүрлі масштабта көру; есеп беруді файлда сақтау; алдын-ала сақталған есеп беруді жүктеу; есеп беруді баспаға шығаруға жіберу.

Preview әдісінің мүмкіндіктері Print әдісінің мүмкіндіктерінен жоғары, сондықтан оларда есеп беруді баспаға шығару алдында көру жиі орындалады.

Есеп беру жолағы (QRBand компоненті) есеп берудің негізгі құрама бөлімі (элементі) болып табылады.

Жолақ типі TQRBandType типті BandType қасиеттерімен анықталады, олар келесі мәндердің бірін қабылдайды:

❒ rbTitle – есеп беру тақырыбы (есеп берудің басында жоғарғы колонтитул астында көрсетіледі);

❒ rbPageHeader – егер есеп беру компонентінің Options қасиеттерінің FirstPageFooter параметрі қосулы (үнсіз келісім бойынша) болса, онда әрбір беттің жоғарғы бөлігінде баспаға шығарылатын жоғарғы колонтитул; егер бұл параметр қосылмаған болса, онда жоғарғы колонтитул баспаға шығарылмайды;

❒ rbDetail – деректер жиынының жазбалары; деректер жиынының әрбір жазбасы үшін шығарылады;

❒ rbChild – ішкі жолағы, өзімен байланысты жолақтан кейін баспаға шығарылады.

Есеп беруге жаңа жолақ кірістіру (қосу) келесі екі әдістердің бірімен анықталады:

❒ QRBand компонентін есеп беруге орналастыру және BandType қасиеттерінен талап етілген мәнді меншіктеу;

❒ QuickRep компонентінің Bands қасиеттерінен сәйкес келетін қасиетке true мәнін тағайындау; сонымен қатар есеп беруге жаңа жолақ қосылады, ал оның BandType қасиеттері автоматты түрде қажетті мәнді тағайындайды.

Әдетте есеп берудің келесі компоненттері қолданылады:

❒ QRLabel – мәтін жазу (жазу және Label аналогы), олар кез-келген жолақта орналасуы мүмкін, бірақ деректердің жолағы үшін әдетте қолданылмайды;

❒ QRDBText – жазба өрістерінің мәні (DBText компонентінің аналогы); әдетте деректердің жолағында орналасады;

❒ QRExpr – өрнектер негізінде қалыптасатын мәндер, онда жазбалар өрісі мәні қолданылуы мүмкін; әдетте деректердің жолағы және төменгі колонтитул үшін қолданылады;

❒ QRSysData – әдетте қорытынды жолағы және колонтитулдер жолағы үшін қолданылатын жүйелік ақпарат; оның түрі TQRSysDataType типінің келесі мәндерді қабылдайтын Data қасиетімен анықталады:

- qrsColumnNo – ағымдағы бағана номері;
- qrsDateTime – ағымдағы мерзім және уақыт;

❒ QRImage – QRImage компоненті көмегімен шығарылатын, Image графикалық бейнесіне ұқсас жүктелетін графикалық тұрғыдағы бейне, мысалы, логотип ұйымдастыру;

❒ QRShape – кез-келген жолақта орналасатын геометриялық фигура.

Бақылау сұрақтары және тапсырмалар

1. Қосымшаны құру кезінде жоба файлдарының өзара байланыс сызбасын келтіріңіз.
2. Формалар модулі құрамына қандай файлдар кіреді және олардың тағайындалуы қандай?
3. C++ Builder жүйелері интерфейсінің негізгі элементтерін сипаттаңыз.

4. Объектілер инспекторы және Код Редакторы терезесінің тағайындалуын сипаттаңыз.

5. Жоба құрамына қандай элементтер кіреді?

6. Көрініс формалары файлы нені қамтиды?

7. Қосымшаның қызметі қалай анықталады?

8. Run, Make және Build командалары әрекеттерінің айырмашылықтары неде?

9. Пайдаланушы интерфейсін құру процесін сипаттаңыз.

10. Компонент өңдеу үшінші оқиғалар қалай құрылады?

11. Бір- және екіденгейлі қосымшаларда қандай механизмдер деректерге қатынас құру сүйемелдейді?

12. Деректер қорын қалыптастыратын кестелерді сипаттаңыз.

13. Кестелерде индекстерді тұрғызудың Paradox және dBase-те айырмашылығын түсіндіріңіз.

14. Деректерге қатынас құру әдістерін сипаттаңыз.

15. Кестелерді байланыстыру дегеніміз не?

16. Paradox деректер қорынада кестелер арасындағы байланыстар сызбасын суреттеңіз.

17. dBase және Paradox кесте форматтарын сипаттаңыз.

18. Компоненттер палитрасының компоненттер орналасқан негізгі беттерін атаңыз.

19. QReport бетінде орналасқан компоненттердің тағайындалуын атаңыз.

20. Database Desktop бағдарламаларының тағайындалуы және мүмкіндіктері қандай?

21. Кестеде өрістер типі қалай тағайындалады?

22. Кестеде индекстерді анықтау қалай орындалады?

23. Кесте индексіне өзгерістер енгізіңіз.

24. Өрістің мәндеріне шектеулер қою нұсқаларын атаңыз.

25. Сілтемелік тұтастық түсінігін түсіндіріңіз.

26. Кестелер құрылымындағы өзгерістер қалай орындалады?

27. Деректер қоры үшін қосымша құрған кезде қолданылатын компоненттерді атаңыз.

28. DataSource компоненті не үшін арналған?

29. Қай роль қосымшалардағы играет компонент DBGrid?

30. Қосымшалардағы деректер қоры үшін қандай әрекет DBNavigator компонентін орындауға мүмкіндік береді?

31. Қосымшаның деректер қоры және кестелер үшін компоненттердің өзарабайланысын түсіндіретін сызбаны суреттеңіз.

32. Қандай компоненттер есеп беруді дайындау кезінде қолданылады?

33. Есеп беру құрамына қандай жолақтар кіреді?

11. MICROSOFT SQL SERVER 2000

Microsoft SQL Server "клиент-сервер" архитектурасымен ақпараттық жүйелер құруды жүзеге асыратын ДҚБЖ болып саналады. SQL Server ақпаратты өңдеудің үлестірімді жүйелеріне қойылатын талаптарды қанағаттандырады. Бұл ДҚБЖ сүйемелдейді: деректерді көбейту, параллель өңдеу, үлкен деректер қорын қымбат емес аппараттық платформаларда құру және өңдеу, басқару және пайдаланудың қарапайымдылығы.

Неғұрлым маңызды жаңалығы - деректер қорын Web-те интеграциялауы. XML тілін пайдаланып, пайдаланушылар ақпаратты деректер қорынан Интернетке оңай жариялай алады, сонымен қатар деректерге қатынас құруды әдеттегі шодушы (броузера) көмегімен қамтамасыз етеді.

11.1. SQL Server сипаттамасы

Біз SQL Server 2000 нұсқасын қарастырамыз. Алдыңғы нұсқалармен салыстырғанда жүйелер ядросы практикалық тұрғыдан сақталған болса да, онда көптеген өзгерістер бар. Осы өзгерістерді енгізуді қасқаша қарастырайық.

Сүйемелдеу жиын инсталляция. Енді бір компьютерге SQL Server 2000 ДҚ серверлерінің бірнеше көшірмесін (инсталляция) тағайындауға болады, олардың әрқайсысы басқару тұрғысынан қарағанда басқаларынан тәуелсіз болып табылады. Инсталляциялардың өзінің аттары бар (*инсталляцияның үнсіз келісім бойынша* атауы болмауы мүмкін).

Сәйкестендіру. SQL Server 7.0 сервері салыстыру және сұрыптау операцияларында Unicode сұрыптау және сәйкестендіру ретті кодтық беттер параметрлерімен беріледі. Сонымен қатар, бұл тек қана барлық серверлер деңгейінде орындалды, яғни барлық ДҚ серверінің барлық кестелерінің барлық бағаналарында бірдей жұмыс жасады.

Пайдаланушы әртүрлі деңгейде жүзеге асырады: сервер үшін, деректер қоры үшін, кестелер үшін және тіпті бөлек бағана үшін. *Пайдаланушымен анықталатын функциялар.* Бұрын MS SQL Server серверде өрнектерде қолданылатын кіріктірілген функциялардың белгілі бір жиыны болды. Күрделі алгоритмдерде пайдаланушылар сақталатын процедураларды шақыру үшін курсорды және басқа механизмдерді қолданатын. Енді сұраныс үшін шақырылатын өз функцияларын құру мүмкіндігіне ие.

Триггерлерді кеңейту мүмкіндіктері. Бұрын тек қана кесте үшін құрылған триггерлерді енді көріністер үшін де қолдануға болады, ол сервердің мүмкіндіктерін елеулі түрде артырады. Алдыңғы нұсқаларда триггердің сәйкес операциялардан кейін жүктелетін тек бір түрі ғана қолданылатын (AFTER типі). Жаңадан енгізілген триггер түрі (INSTEAD OF типі) триггерді жүктеуді қажет ететін пайдаланушының командасымен бірге орындалады.

Индекстердің жақсаруы. Кестені кәдімгі индекстеумен бірге көріністер мен есептелетін бағаналарды (computed columns) индекстеуге рұқсат береді. Бұл сервердің жұмыс өнімділігін арттыруға мүмкіндік береді. Сонымен қатар, серверді қолдану жүйелік ресурстар тұрғызған кезде және индекстерді жаңарту кезінде тиімді.

Каскадты өзгерістер енгізу. Алдыңғы нұсқаларда сыртқы кілттермен жұмыс жасау кезінде сілтемелердің дұрыстығына пайдаланушы жауапты болатын. Енді барлық қажетті өзгерістерді кілттер автоматты түрде орындайды.

Сенімділігі. Алдыңғы нұсқаларда сервер құрылғыларының тоқтап қалуы және істен шығуына жүйелердің тұрақтылығын арттыру үшін ДҚ администраторы екі негізгі технологияны пайдаланды: резервтік сервер (standby server) және кластерлеу (fail-over support).

Біріншісі желіге бір немесе бірнеше қосымша серверлерді орнатуды жобалайды, оларда оператор командалары бойынша негізгі сервер ақпараттарының периодты түрде көшірмесі алынады. Негізгі сервер істен шыққан жағдайда резервтік сервер негізгі сервер ретінде конфигурацияланады, содан соң пайдаланушылар өз жұмыстарын резервтік сервермен жалғастыра алады.

Басқа неғұрлым жетілдірілген технология ДҚ серверін виртуалдық сервер (кластер) ретінде пайдалануға мүмкіндік береді, физикалық тұрғыдан ол екі компьютердің базасында (түйіндер) құрылады

Жаңа деректер типі. Пайдаланушылар үшін деректердің үш жаңа типі пайда болды:

1. bigint, неғұрлым кеңейтілген диапазондағы бүтін сандарға арналған және көрініс үшін 8 байт қолданады, әдеттегі int 4 байт қолданатын;
2. sql_variant, көптеген басқа ақпараттық-бағдарламалық қамтамасыз ету мәндерін сақтауға мүмкіндік беретін мәні (сандық, символдық, қаржылық, мерзімдер және т.б.) және айнымалылармен, кестелер бағаналарымен жұмыс жасау кезінде қолданылатын мәндер;
3. table, кестеге ұқсас деректердің күрделі жинағын сақтауға мүмкіндік береді. Оны тек қана жергілікті айнымалылармен жұмыс жасау үшін қолдануға болады.

SQL Server 2000 құрал-саймандары

Құрал-саймандарды басқаруды сервердің өзін баптау кезінде және бөлек орнатылуы мүмкін. Бұл басқа компьютерде администратордың жұмыс орнын ұйымдастыруға мүмкіндік береді. Оларды жергілікті желіні кез-келген SQL Server 2000 немесе SQL Server 7.0 сервермен басқару үшін қолдануға болады.

SQL Server 2000 сервермен басқарудың негізгі құрал-саймандары:

1. SQL Server Enterprise Manager;
2. SQL Server Service Manager;
3. SQL Server Profiler;
4. Query Analyzer;
5. Upgrade Wizard (Жаңарту шебері);
6. Import and Export Data (экспорт/импорт шебері);
7. Client Network Utility және Server Network Utility утилиталары;
8. командалық жолдар утилиталары;
9. арнайы Шебер (Wizards).

SQL Server әкімшілік міндеттерінің үлкен бөлігі әртүрлі үш әдіспен орындалуы мүмкін: Transact-SQL құралдарын қолдану арқылы, SQL Server Enterprise Manager графикалық интерфейсі көмегімен және Шеберді пайдалану арқылы.

SQL Server Enterprise Manager келесі әрекеттерді орындауға мүмкіндік беретін маңызды құрал-сайман болып табылады:

- жүйелік қауіпсіздікті басқару;
- ДҚ және оның объектілерін құру;
- резервтік көшірмелерді құру және қалпына келтіру;
- SQL Server 2000 қызметінің жұмыс параметрлерін басқару;
- ішкі жүйелерді автоматтандыруды басқару;
- жүктеу, тоқтату және уақытша тоқтату қызметі;
- байланыс және қашықтықтағы серверлерді конфигурациялау.

SQL Server Service Manager пайдаланушыға SQL Server 2000 жүктеу, тоқтату және уақытша тоқтату қызметінің қолайлы механизмін ұсынады. Сонымен қатар, утилита автоматты түрдей жүктеуге рұқсат беру немесе тыйым салу мүмкіндіктерін береді.

SQL Server Profiler администратордың SQL Server 2000 жұмысын қадағалауына арналған графикалық құрал-сайман болып табылады. *Query Analyzer* сұраныстар және олардың нәтижелерін талдауды орындауға арналған.

Upgrade Wizard шебері SQL Server 6.5 деректер қорын SQL Server 2000 дейін жаңарту үшін қолданылады.

Import and Export экспорт/импорт шебері көмегімен қарапайым тапсырмаларды орындау оңай.

Client Network Utility және *Server Network Utility* утилиталары желілік параметрлерді клиенттік және серверлік бөлімдерде конфигурациялауға арналған. Желілік параметрлер арнайы желілік кітапхана көмегімен орнатылады.

Командалық жолдар утилиталары, қағидаға сәйкес, графикалық интерфейсі жоқ және әртүрлі тапсырмаларды орындауға арналған. Утилиттер SQL Server 2000 бастапқы орнату каталогтарының Binn каталогында орналасқан. Мысалы: console.exe (хабарламаларды көру бағдарламасы), dtsrun.exe (DTS дестелерімен басқару бағдарламасы), isqlw.exe (Query Analyzer), sqlserver.exe (MSSQLServer қызметінің жүзеге асуы бағдарламасы).

Шебер әртүрлі әкімшілік тапсырмалардың орындалуын автоматтандыру және ықшамдауға арналған. Көптеген шеберлердің мүмкіндіктері жеткілікті дәрежеде шектеулі.

Жеткізілім нұсқалары

SQL Server жеткізілім нұсқалары:

1. Enterprise (мекемелік);
2. Standard (стандартты);
3. Desktop Engine (үстелдік);
4. Windows CE (Microsoft Windows CE ортасында жұмыс жасау үшін);
5. Personal (дербес);
6. Developer (қосымшаны өңдеу және тестілеу үшін);
7. Evaluation (бағалаушылық).

SQL Server орнату әдеттегі пайдаланушы үшін жоғары, ал өндірістік серверлер үшін орындауға болатын талаптар қояды (13.1-кестені қараңыз).

13.1-кесте. SQL Server 2000 серверінде аппараттық қамтамасыз етуге қойылатын талаптар

Сипаттамасы	Минимальды талаптар	Ұсынылатын талаптар
Процессор	Pentium 166, Pentium Pro	Pentium II немесе Pentium III
Жедел жады	32 Мбайт (Enterprise Edition үшін 64 Мбайт)	64-128 Мбайт (Enterprise Edition үшін 128-256 Мбайт)
Дискілік жады	Ең аз орнату үшін 65 Мбайт; тек қана утилиттерді басқаруды орнату кезінде 90 Мбайт; үйреншікті орнату үшін 170 Мбайт; толық орнату үшін 180 Мбайт; OLAP орнату үшін +50 Мбайт; English Query орнату үшін +12 Мбайт	Минимальды талаптармен ұқсас, бірақ пайдаланушы ДҚ өлшемдерін ескеру керек

SQL Server 2000 неғұрлым әлсіз компьютерлерде де жұмыс жасайды, бірақ одан жоғары өнімділікті күтудің қажеті жоқ.

Windows операциялық жүйесінің әртүрлі нұсқалары үшін SQL Server 2000 қайсыбір редакциясын ғана орнатуға болады (13.2-кесте).

13.2-кесте. Редакциялар және операциялық жүйелерді таңдау

Операциялық жүйе	Enterprise Edition	Standard Edition	Personal Edition	Developer Edition	Desktop Engine	SQL Server CE	Enterprise Evaluation Edition
Windows 2000 Datacenter/Advanced Server/Server	+	+	+	+	+	-	+
Windows 2000 Professional	-	-	+	+	+	-	+
Windows NT 4.0 Server/Enterprise Edition	+	+	+	+	+	-	+
Windows NT 4.0 Workstation	-	-	+	+	+	-	+
Windows 98	-	-	+	-	+	-	-
Windows CE	-	-	-	-	-	+	-

11.2. Transact-SQL сұраныстар тілі

SQL Server деректер қорын құру және жұмыс жасау үшін Transact-SQL деп аталатын SQL тілін қолданады. Алғашқы SQL тілімен салыстырғанда Transact-SQL тіліне қосымша кілттік сөздер енгізілген. Олар деректермен операцияларды таңдау, сақтау және орындау кезінде қолданылады.

Негізгі операторы INSERT, DELETE, UPDATE және SELECT. Transact-SQL қосымша мүмкіндіктері көбінесе ақпараттардың ағымын басқарумен және операторлардың орындалу ретін анықтаумен байланысты.

Transact-SQL операторларын ISQL (Interactive Structured Query Language – құрылымдық сұраныстардың интербелсенді тілі) утилитасының көмегімен беруге болады. Windows үшін ISQL нұсқасы сұраныстарды талдау (Query Analyzer) деп аталады.

Transact-SQL операторлары Query (Сұраныс) терезесінде енгізіледі. Сұраныстың орындалу нәтижелері Results (Нәтижелері) терезесінде бейнеленеді.

11.3. Жүйелік деректер қоры және кестелер

SQL Server серверде ***деректер қоры*** кестелер және индекстер орналасқан логикалық объект болып саналады. Физикалық деректер қоры операциялық жүйелердің бір немесе бірнеше файлдарын қамтиды.

Кесте (table) жиын өрісі және жазбалар болып саналады. Кестелердің екі ақпараттық-бағдарламалық қамтамасыз етуі бар: *тұрақты* және *уақытша*. Тұрақты кестелер оларды өшіргенге дейін сақталады. Уақытша кестелер жергілікті және ауқымды болып бөлінеді. *Жергілікті* уақытша кестелер ағымдағы жұмыс кезеңінде қолданылып, содан соң жойылады. *Ауқымды* уақытша кестелер оларды қолданатын барлық жұмыс кезеңдері аяқталғанға дейін сақталады.

Транзакциялар журналы жұмыс облысы болып саналады, онда SQL Server ақпаратты транзакциялар орындалғанға дейін және орындалғаннан кейін жазады. Транзакциялар журналы деректер қорын құрған кезде автоматты түрде құрылатын жеке файлда орналасады.

Деректер қорын сақтау үшін келесі үш типті файлдар қолданылады:

1. *Негізгі* (Primary) файл біреу ғана құрылады және инициализациялау үшін талап етілген ақпаратты қамтиды. Үнсіз келісім бойынша файл кеңейтуі mdf;
2. *Көмекші* (Secondary) файлдар деректерді қамтиды; оларды қолдану міндетті емес, бірақ ол ДҚ бірнеше жинақтауыштарда орналастыруға мүмкіндік береді. Үнсіз келісім бойынша файлдар кеңейтуі ndf;
3. *Транзакциялар журналы* файлында ДҚ қалпына келтіру үшін қажетті ақпараттар сақталады. Үнсіз келісім бойынша файл кеңейтуі ldf.

Деректерді сақтау үшін деректер қорында орналасқан кестелер қолданылады. Microsoft SQL Server деректер қорын екіге бөледі — *жүйелік* және *пайдаланушылық*. ***Жүйелік*** деректер қорында жүйені басқару үшін қолданылатын метадеректер орналасады. Microsoft SQL Server келесі жүйелік деректер қорын құрады: *master*, *model*, *tempdb* және *msdb*.

Жүйелік деректер қоры ***master*** пайдаланушылық деректер қорын басқаруды және Microsoft SQL Server жұмысын қамтамасыз етеді. Онда келесі деректер болады:

- Пайдаланушылардың тіркелу жазбалары;
- ағымдағы процестер туралы мәліметтер;
- жүйелік қателер туралы хабарлама;
- сервердегі деректер қоры туралы мәліметтер;
- деректер қорының ерекшеленген өлшемі;
- шектеу қою туралы мәліметтер;
- деректер қорының қолжетімді және резервтік құрылғылары туралы мәліметтер;
- жүйелік басқару процедуралары.

Жүйелік деректер қоры ***model*** деректер қоры үшін ағымдағы серверде құрылатын үлгі болып саналады. Ол деректер қорының әрбір пайдаланушысына қажетті жүйелік кестелерді қамтиды. *model* деректер қорына деректер қорына қажетті объектілер орналастырылады. Олар келесі объектілер:

- пайдаланушылармен анықталатын деректер типі;
- енгізуді тексеру ережелері;
- үнсіз келісім бойынша мәні;
- сақталатын процедуралар;
- деректер қорына қатынас құруға рұқсат етілген пайдаланушылар туралы ақпарат;
- рұқсатнамалар.

Жүйелік деректер қоры ***tempdb*** дискіге әртүрлі уақытша объектілерді: кесте, топтау және реттеудің аралық нәтижелері, курсорлар және басқаларды орналастыруға арналған. SQL Server жұмысын аяқтағаннан кейін бұл деректер қорынан барлық деректер жойылады.

Жүйелік деректер қоры ***msdb*** SQLServerAgent қызметін қамтамасыз ету үшін қолданылады. Онда SQL Server 2000 басқаруды автоматтандыру және басқару

ақпараттары, операторлар және оқиғалар туралы ақпарат, сонымен қатар тапсырмаларды автоматты түрде жүктеу туралы ақпарат сақталады.

msdb деректер қорында келесі жүйелік кестелер болады: *sysalerts*, *sysoperators*, *sysnotifications*, *sysjobhistory, sysjobs* және басқалар. ***Деректер қоры каталогы*** (database catalog) әрбір деректер қорында болатын 20-ға жуық жүйелік кестелер жиынтығы болып саналады. Барлық жүйелік кестелер атаулары sys сөзінен басталады, мысалы: *sysalternates, syscolumns, syscomments*. Оларда пайдаланушылар, триггерлер және сақталатын процедуралар, кестелер, кестелер индекстері, пайдаланушылардың қатынас құру құқығы, деректердің түрі, шектеулер, публикациялар, репликациялар туралы және басқа ақпараттар сақталады. Жүйелік кесте атауы онда сақталатын ақпараттардың сипатын көрсетеді. Мысалы, *sysindexes* жүйелік кестеде индекстер туралы деректер болады.

Жүйелік каталог немесе *деректер сөздігі* ***master*** деректер қорында орналасқан жүйелік кестелердің қайсыбір жиынтығы болып саналады. Жүйелік каталог құрамына келесі жүйелік кестелер кіреді: syscharsets, sysconfigures, syscurconfigs, sysdatabases, sysdevices, syslanguages, syslocks, syslockinfo, sysxlogins, sysmessages, sysprocesses, sysservers және басқалар.

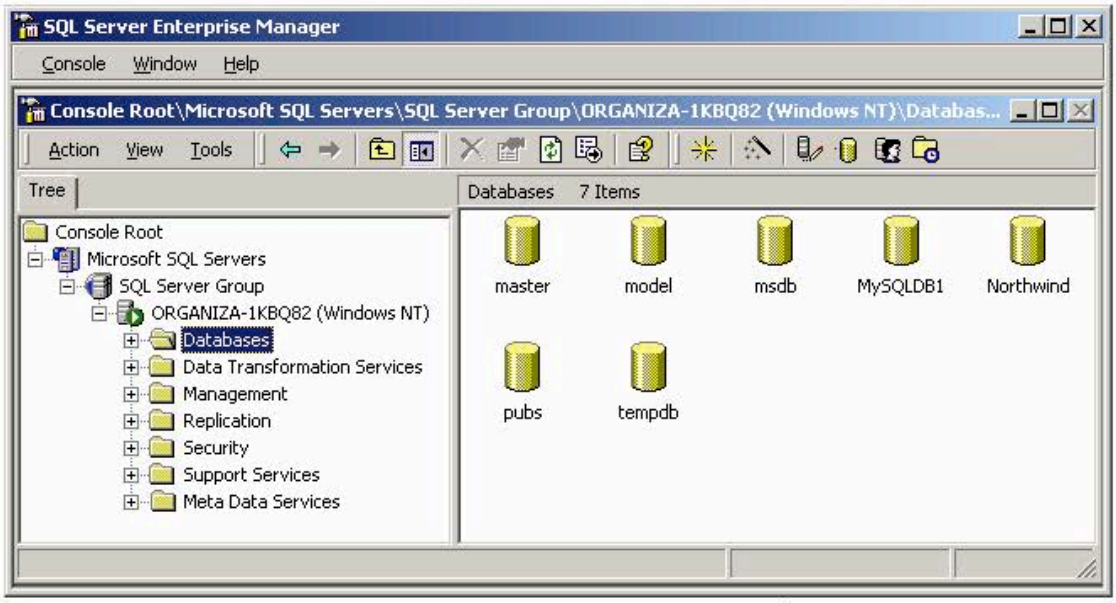
11.4. Деректер қорын құру

Серверде орналасқан деректер қорымен жұмыс жасау кезінде келесі кезеңдерді ерекшелеуге болады:

- деректер қорын және кесте құру;
- көріністер және сақталатын процедуралар құру;
- кестелермен жұмыс;
- деректерді қалпына келтіру;
- басқару.

Деректер қорын құру үшін SQL Server Enterprise Manager немесе Transact-SQL CREATE DATABASE командасын қолдануға болады. Деректер қорын SQL Server Enterprise Manager көмегімен құру үшін келесілерді орындаймыз:

1. Аталған бағдарламаны жүктеу, сұқбаттасу терезесінде деректер қоры серверін және Databases бумасын таңдау (11.1-сурет).
2. Action | New Database (Әрекет | деректер қорын құру) командасын орындау.
3. Ашылған сұқбаттасу терезесінде деректер қоры параметрлерін енгізу: атауы, файлдардың орналасқан жері, бастапқы өлшемі, өлшемін автоматты түрде көбейту мүмкіндіктері.



11.1-сурет. SQL Server Enterprise Manager сұқбаттасу терезесі

Деректер қорын құруға арналған Transact-SQL командасы келесі түрде жазылады:

CREATE DATABASE **ДҚ аты**
[ON [< файлдың мамандануы> [,…n]] [, <файлдар тобын сипаттау> [,…n]]]
[LOG ON <файлдың мамандануы> [,…n]]
[COLLATE сәйкестендіру аты]
[FOR LOAD | FOR ATTACH]
<файлдың мамандануы> : : = [PRIMARY]
(NAME = файлдың логикалық аты ,
FILENAME = ' операциялық жүйелер файлы аты '
[, SIZE = өлшемі]
[, MAXSIZE = { ең үлкен өлшемі | UNLIMITED }]
[, FILEGROWTH = өлшемін ұлғайту])
<файлдар тобын сипаттау> : : = FILEGROUP <файлдар тобы атауы> <файлдың мамандануы> [,…n]

CREATE DATABASE командасы параметрлерінің мағынасы:

1. ON деректер қорына ақпараттардың орналасуы үшін дискідегі файлдар тізімін анықтайды;
2. n - бұл символдың орнына ДҚ орналасу үшін қосымша файлдарды көрсетуге болады;
3. LOG ON транзакциялар журналында орналасу үшін дискідегі файлдар тізімін анықтайды;
4. COLLATE деректер қорында болатын салыстыруларды көрсетеді;
5. FOR LOAD алдыңғы SQL Server нұсқаларымен үйлесімділік үшін қолданылады;
6. FOR ATTACH жаңа ДҚ құруды емес, құрылған ДҚ-на қосуды орындау;
7. COLLATE деректер қорында болатын салыстыруларды көрсетеді;
8. PRIMARY бастапқы файлды анықтайды. Егер параметр берілмесе, онда тізімдегі бірінші файл бастапқы файл болып табылады;
9. SIZE бастапқы файл өлшемін анықтайды, параметрдің ең аз мәні 512 Кбайт; егер ол көрсетілмесе, онда үнсіз келісім бойынша 1 Мбайт мәнін қабылдайды;
10. MAXSIZE деректер қоры файлының ең үлкен өлшемін анықтайды.

Деректер қорын құру алдында USE master командасының көмегімен *master* ағымдағы деректердің қорын ашу және тағайындау керек.

1-мысал. Деректер қорын құру.

```
CREATE DATABASE owndbase266
ON PRIMARY
(NAME  = owndbase266_data,
FILENAME = ' C:\mssq117\data\ ' owndbase266_data.mdf ',
        SIZE = 4 MB,
        FILEGROWTH = 1 MB)
LOG ON
(NAME  = owndbase266_log,
FILENAME = ' C:\mssq117\data\ ' owndbase266_log.ldf ',
        SIZE = 2 MB,
        FILEGROWTH = 1 MB)
```

ДҚ құрған кезде model деректер қоры үлгісі қолданылады.

11.5. Кестелермен жұмыс

Кестелермен жұмыс жасау кезінде кесте құру, кестеге деректер кірістіру немесе қосу, таңдау, өшіру және кестедегі деректерді өзгерту жобаланады.

Enterprise Manager графикалық интерфейсі көмегімен немесе Transact-SQL тілінің CREATE TABLE командасын пайдаланып ***кесте құруға*** болады. SQL Server 2000 серверде кесте құру шебері, өкінішке орай, жоқ.

Жергілікті уақытша кестелер құру үшін оның атауында алдымен # символын көрсетеді. Олар жұмыс аяқталғаннан соң автоматты түрде жойылады. Жергілікті уақытша кестелер атауы # символымен бірге 116 символға дейін қамтуы мүмкін.

Ауқымды уақытша кестелер құру үшін оның атауында екі # символын көрсетеді.

Кестелер құрған кезде әрбір оның өрістері үшін өрісте сақтауға болатын ақпараттардың типін анықтайтын ***деректердің типі*** беріледі. Деректердің типі анықталғаннан кейін бағана оны тұрақты сипаттама ретінде сақтайды. Деректердің типін ALTER TABLE операторы көмегімен өзгертуге болады. SQL Server 2000 кестелердегі деректер типі (жүйелік) 11.3-кестеде келтірілген.

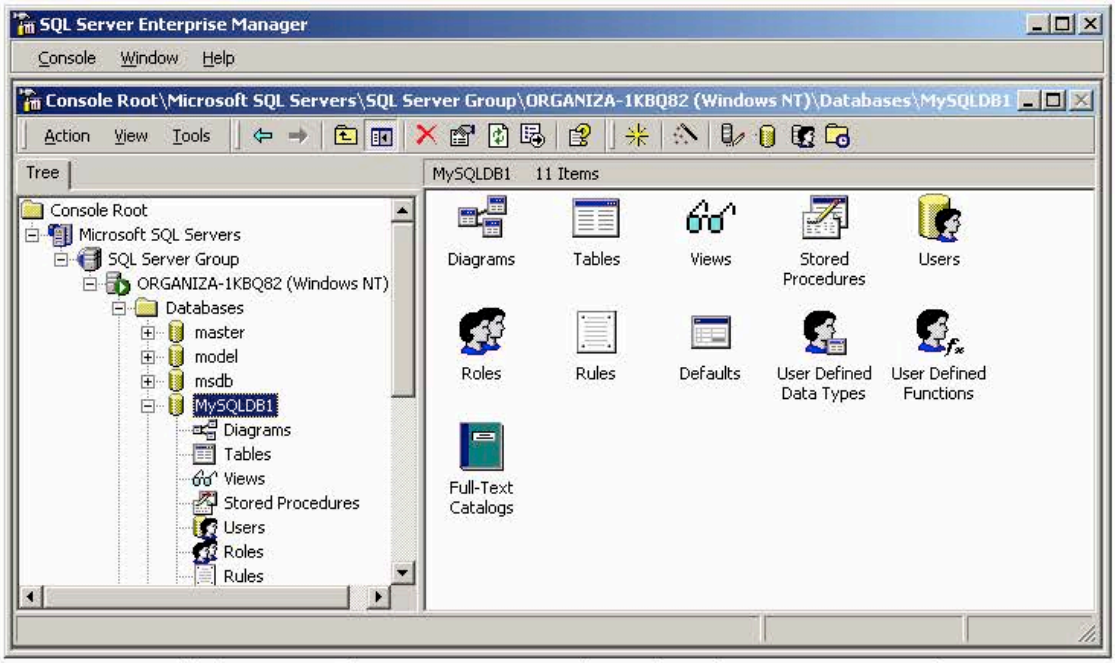
11.3-кесте. Деректер типі

Типі	Өлшемі, байт	Қысқаша сипаттамасы
binary(n)	8 Кбайт дейін	Ұзындығы белгіленген екілік деректер
varbinary(n)	8 Кбайт дейін	Ұзындығы айнымалы екілік деректер
char(n)	8 Кбайт дейін	Ұзындығы белгіленген символдық жол
nchar(n)	8 Кбайт дейін	Ұзындығы белгіленген Unicode символдық жолы
varchar(n)	8 Кбайт дейін	Ұзындығы айнымалы символдық жол
nvarchar(n)	4 Кбайт дейін	Ұзындығы айнымалы Unicode символдық жолы
datetime	8	Жоғары дәлдікпен мерзім және уақыт
smalldatetime	4	Төменгі дәлдікпен мерзім және уақыт
decimal(p,s)	1-17	p цифр және үтірден кейінгі s цифр жалпы саны
numeric(p,s)	1-17	p цифр және үтірден кейінгі s цифр жалпы саны
float	8	Өзгермелі нүктелі сан
real	4	Өзгермелі нүктелі сан

bigint	8	Бүтін сан
int	4	Бүтін сан
smallint	2	Бүтін сан
tinyint	1	Бүтін сан (0-ден 255-ке дейін)
money	8	Қаржылық мән
smallmoney	4	Қаржылық мән
bit	1	Булевтік мән (бит, мәні 0 немесе 1)
timestamp	8	Уақытша белгілеу немесе ДҚ жаңартудан кейін ерекшелеу
text	2 Гбайт дейін	Мәтіндік деректер
ntext	1 Гбайт дейін	Мәтіндік деректер Unicode
sql_variant		text, ntext, timestamp, image және sql_variant мәндерінен басқа әртүрлі ақпараттық-бағдарламалық қамтамасыз ету мәндерін сақтау
image	2 Гбайт дейін	Екілік деректер (MS Office файлдары, бейнелеулер және т.с.с.)
table		Сұраныстардың орындалу нәтижелерін сақтау
uniqueidentifier		Ауқымды ерекше идентификатор (globally unique identifier, GUID)

SQL Server Enterprise Manager көмегімен *тұрақты* кестелер құру үшін келесілер орындалады:

- SQL Server Enterprise Manager жүктеу;
- ашылған сұқбаттасу терезесінде топтар атауларының сол жағындағы "+" белгісін басу арқылы серверлік топтарды таңдау;
- кестелер құрылатын деректер қорын таңдау (11.2-сурет);
- деректер қорынан Tables (кестелер) объектісін таңдау және контексті мәзірден немесе Action (әрекет) пунктінің негізгі мәзірінен New Table (Жаңа кесте) командасын таңдау;
- ашылған сұқбаттасу терезесінде кесте бағаналары атауларын көрсету, әрбір бағана үшін деректердің типін, өлшемін, Null мүмкін мәндерін және үнсіз келісім бойынша (Default) мәнін анықтау;
- Specify Table Name терезесінде атауын көрсетіп, кестені сақтау (Save Table батырмасы).



11.2-сурет. Деректер қоры объектілерімен терезе түрі

Кестелер құруды Transact-SQL тілінің CREATE TABLE командасының көмегімен орындауға болады. Команда келесі түрде жазылады:

```
CREATE TABLE [ДҚ_аты.[ДҚ иесі] | ДҚ иесі] кестелер_аты
({<бағананы_анықтау> | бағана_аты AS есептелетін_өрнек |
<кестелер_шектеулері>} [,…n])
[ON {файлдар_тобы | DEFAULT}]
[TEXTIMAGE_ON {файлдар_тобы | DEFAULT}]

<бағананы_анықтау>::= бағана_аты  деректердің_типі
[COLLATE <атын_сәйкестендіру>]
[  [DEFAULT өрнек] |
   [IDENTITY [(бастапқы_мәні, өзгеруі)
     [NOT FOR REPLICATION]]]
]
[ROWGUIDCOL]
[<бағанаға_шектеулер>]
```

Кестелер құрған кезде деректер қорын және ДҚ иесін көрсетуге болады. USE деректер_қоры командасының көмегімен ағымдағы деректердің қорын алдын-ала анықтау тиімді. Сонымен қатар, барлық келесі командалар осы деректер қоры үшін орындалады, және оның атын көрсетпеуге болады. Кестелер аты және кесте бағаналары 128 символға дейін қамтиды.

1-мысал. Кестелер құру.

Үш бүтін ақпараттық-бағдарламалық қамтамасыз ету бағаналары бар кестелер құру операторын қарастырайық.

```
CREATE TABLE table1
(int1 TINYINT, int2 INT, int3 SMALLINT)
```

2-мысал. Қызметкерлер туралы жеті бағаналардан тұратын кесте құру талап етілсін. Бағаналарда келесі мәліметтер берілген: қызметкердің цифрлық коды, фамилиясы, аты, әкесінің аты, жынысы (М – ер адам, Ж – әйел адам), туған күні және қызметі. Сонымен қатар, қызметкер коды кестелердегі бастапқы кілт ролін атқарады. Аты және фамилиясы бос болмауы және кестелердегі фамилиясы және әкесінің аты деректері алфавит бойынша сұрыпталған болуы керек. Жасы бойынша да өспелі түрде реттеу қажет.

Сипатталған кестелерге оператор құру келесі түрде жүзеге асады:

```
CREATE TABLE tab2
(emp_id int IDENTITY (1,1) PRIMARY KEY,
 second_name varchar(30) NOT NULL,
 first_name varchar(15) NOT NULL,
 middle_name varchar(20) NULL,
 sex char(1),
 birth_date datetime,
 position varchar (50)
 CONSTRAINT const_tab2 UNIQUE
  (second_name ASC,first_name ASC, birth_date DESC)
)
```

IDENTITY қасиеттері кесте бағаналарының бастапқы мәнін анықтауға мүмкіндік береді. Олар автоматты түрде бағанада бірінші жолда орналасқан.

Кесте ***деректермен жұмыс жасау*** кезінде ақпараттарды таңдау, қосу, өшіру және өзгерту сияқты негізгі әрекеттерді орындайды.

Деректерді кестеге ***кірістіру немесе қосуды*** INSERT операторы көмегімен орындауға болады:

```
INSERT INTO кестелер_аты
(1_бағана,…,n_бағана)
VALUES ('1_жол',…,'n_жол')
```

Оператордың екінші жолында кесте бағаналары атаулары беріледі. Келесі жолда VALUES кілттік сөзімен дөңгелек жақша ішінде кесте бағаналарына қосылатын мәндер көрсетіледі. Әрбір INSERT операторы кестеге бір жазба (жол) кірістіруге (қосуға) мүмкіндік береді.

3-мысал. Кестеге жазба кірістіру немесе қосу.

Кестенің үш бағаналары үшін үш бүтін мәндерден тұратын жолдар кірістіру немесе қосуды Қарастырайық.

```
INSERT INTO table1
(int1, int2, int3)
 VALUES (255, 32767, 50000)
```

4-мысал. Кестеге жазба кірістіру немесе қосу.

```
INSERT INTO table2
(Name, Sity, Year)
 VALUES ('Salivan', 'New York', 2004)
```

NULL және NOT NULL атрибуттары. NULL атрибуты кестеге деректер мәнін енгізу кезінде бұл бағананы бос қалдыруға мүмкіндік береді. NOT NULL атрибуты жағдайында бұл бағана мәнін енгізу міндеттеледі. Үнсіз келісім бойынша бағанаға NOT NULL атрибуты тағайындалады.

Кестеден ***деректерді таңдау*** үшін SELECT операторы қолданылады. Оның көмегімен бағаналар және жолдарды көрсете отырып, кестеден деректерді таңдаудың әртүрлі сұраныстарын құруға болады. Кестелерден деректерді таңдауға арналған SQL-сұранысты дайындау үшін келесі құралдарды қолдануға болады:

1. Query Analyzer утилитасы;
2. ISQL командалық жолдар утилитасы;
3. OSQL командалық жолдар утилитасы.

Неғұрлым тиімдісі Query Analyzer бағдарламасы.

Индекстер мен кілттер

Индекстер деректерді шығаруды және таңдауды жылдамдату үшін қолданылады. Индекстер деректерді кестеден таңдауды жылдамдатуға мүмкіндік беретін қажетті деректерге сілтеуіштер ролін атқарады. Индекс құрған кезде SQL Server 2000 кестелерді сканерлеуді орындайды, индекстелетін бағана мәнін таңдайды және индекстелетін бетке беттер деректеріне сілтемелер жазады.

SQL Server 2000 индекстердің екі түрін қолданады: кластерлік және кластерлік емес.

Кластерлік типті индекстерде жолдардың орналасу реті деректердің орналасуының физикалық ретімен сәйкес келеді. Кластерлік индекс тікелей деректерді қамтиды, сондықтан кестеде кластерлік типті бір индекс болуы мүмкін. Бұл индекс деректерге қатынас құрудың жоғары жылдамдығын қамтамасыз етеді.

Кластерлік емес типті индекстер деректермен логикалық байланысты болады, сондықтан бір кесте үшін бірнеше мұндай индекстерді беруге болады.

Индекс құру кезінде келесілерді ескеру қажет:

1. деректердің типі BIT, TEXT немесе IMAGE болатын бағаналар үшін индекс құруға болмайды;

2. деректердің типі CHAR, VARCHAR, NCHAR, NVARCHAR, BINARY және VARBINARY болатын бағаналар үшін индекс құруға болмайды;

3. индекс құрудың негізгі кандидаты - алғашқы кілт;

4. нәтижелердің үлкен жиынын беретін сұраныстар құрылатын кесте үшін индекстеудің барлық мүмкіндіктерін пайдаланған жөн;

5. индекстеуде ORDER BY (бойынша реттеу) және GROUP BY (бойынша сұрыптау) қолдану тиімді.

Индекстерді екі әдіспен құруға болады: SQL Server Enterprise Manager бағдарламалары көмегімен және Transact SQL тілінің CREATE INDEX операторы көмегімен.

SQL Server Enterprise Manager бағдарламалары көмегімен индекстер құру үшін келесілер орындалады:

1. Microsoft SQL Server топтарынан аталған бағдарламаны жүктеу.

2. индекс құру орындалатын сервер, деректер қоры және кестелерді таңдау.

3. Action | All tasks | Manage Indexes (Әрекет | Барлық тапсырмалар | Индекстермен жұмыс) командасын орындау.

4. Ашылған Manage Indexes (Индекстермен жұмыс) сұқбаттасу терезесінде New (Құру) батырмасын басу.

5. Create New Index (Жаңа индекс құру) сұқбаттасу терезесінде Index Name (Индекс аты) өрісінде индекс атын беру.

6. Index Options (Индекс параметрлері) тобында индекс параметрлерін тағайындау және ОК батырмасын басу.

Деректерді таңдау операторының қарапайым түрі:

SELECT…

FROM кестелер_аты (INDEX = n)

…

Мұндағы INDEX кілттік сөзі SQL Server n мәнімен анықталған индексті қолдануы керектігін білдіреді. Егер n мәні нөлге тең болса, онда кестелерді сканерлеу орындалады. Егер n мәні бірге тең болса кластерлік индекс қолданылады. Қалған n мәндері индекстердің реттік номерін анықтайды.

Индекстерді *өшіру* екі әдіспен жүзеге асады: SQL Server Enterprise Manager бағдарламалары көмегімен және Transact SQL тілінің DROP INDEX операторы көмегімен.

Кілттер деректер қоры кестелерінде деректер тұтастығын сүйемелдеу үшін қолданылады (3.1-3.2 бөлімдерді қараңыз). Ескере кетейік, *алғашқы кілт* бірегей өріс болып саналады.

SQL Server алғашқы және сыртқы кілттер құруды үш әдіспен жүзеге асырады:

1. SQL Server Enterprise Manager бағдарламалары көмегімен;
2. Transact SQL тілінің ALTER TABLE…ADD CONSTRAINT операторы көмегімен;
3. CREATE TABLE операторының PRIMARY/FOREIGN KEY параметрі көмегімен.

SQL Server Enterprise Manager бағдарламалары көмегімен *алғашқы кілтті* қосу үшін келесілер орындалады:

1. Microsoft SQL Server топтарынан аталған бағдарламаны жүктеу.
2. Сервер, деректер қоры және кестелерді таңдау.
3. Action | Design Table (Әрекет | Кесте конструкторы) командасын орындау.
4. Алғашқы индекстер ретінде анықталатын бағаналарды бөліп алу, және кестелердің контексті мәзірінен Set Primary Key (алғашқы кілтті тағайындау) командасын орындау.
5. Save (Сақтау) батырмасы арқылы сақтау.

Алғашқы және сыртқы кілттерді *өшіру* SQL Server Enterprise Manager бағдарламалары көмегімен және Transact SQL тілінің ALTER TABLE…DROP CONSTRAINT операторы көмегімен жүзеге асады.

11.6. Сақталатын процедуралар мен триггерлер

Сақталатын процедуралар Transact SQL тілінің серверде орындалатын бағдарламалық коды фрагменттерін білдіреді. Сақталатын процедуралар пайдаланудың негізгі *құндылықтары:* қуатты компьютер-серверді қолданғандықтан, ақпаратты өңдеудің жоғары жылдамдығы және деректер қоры серверде орналасқандықтан, деректерге қатынас құру үшін артық уақыт кетпейді

Сақталатын процедуралардың артықшылықтар: жоғары өнімділігі, жергілікті және қашықтықтағы серверде орындалу мүмкіндігі.

Сақталатын процедуралардың ақпараттық-бағдарламалық қамтамасыз етулері: жүйелік (System), жергілікті (Local), уақытша (Temporary) және қашықтықтағы (Remote).

Жүйелік сақталатын процедуралар сервер және администратор қолданатын *Master* деректер қорында орналасады. Жүйелік процедура аттары sp_ символынан басталады.

Жергілікті сақталатын процедуралар пайдаланушылық деректер қорында орналасады.

Уақытша сақталатын процедуралар # немесе ## символдарынан басталатын атауға ие. Процедуралар бір # символымен басталса, онда уақытша *жергілікті* процедуралармен жұмыс жасайды. Процедуралар екі ## символымен басталса, онда уақытша *ауқымды* процедуралармен жұмыс жасайды.

Қашықтықтағы сақталатын процедураларды серверден шақыруға болады.

Сақталатын процедураларды CREATE PROCEDURE операторы көмегімен және SQL Server Enterprise Manager бағдарламалары көмегімен құруға болады .

Сақталатын процедуралар аты 128 символға дейін қамтиды. Сақталатын процедураларды SQL барлық операторлары (CREATE операторынан басқа) қолданылады. Сақталатын процедураларды *құру* операторы келесі түрде жазылады:

```
CREATE PROCEDURE [иесі.]процедуралар_аты [;нқсқасы]
[(@параметр_аты типі [=default] [OUTPUT]
. . .
[@параметр_аты типі [=default] [OUTPUT]])]
[FOR REPLICATION | WITH RECOMPILE][, ENCRYPTION]
AS
<Transact-SQL_операторлары>
```

Сақталатын процедураларды шақыру олардың аты бойынша орындалады. Егер сақталатын процедура топтар бөлімі болып табылса, онда оны шақыру EXEC командасының көмегімен жүзеге асады. Мысалы,

```
EXEC proc; 3
```

1-мысал. Сақталатын процедураларды құру және қолдану.

Employee кестелерден жазбаларды таңдайтын процедура құрайық, таңдау шарты екі параметр көмегімен анықталады.

```
CREATE PROCEDURE procSelect
(@p1 char(30), @p2 char (20))
AS
SELECT Name, Department, Cost FROM Employee
WHERE  @p1=@p2
```

Процедураға қатынас құру келесі түрде болуы мүмкін:

```
procSelect Name, 'Kooper'
```

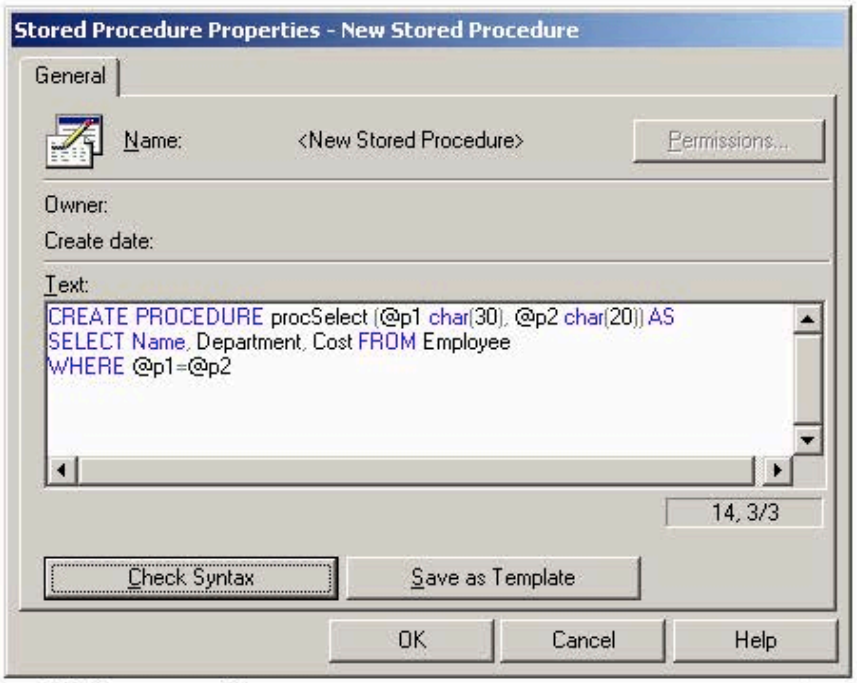
нәтиже келесі түрде болуы мүмкін:

```
Name          Department          Cost
- - - - - - - - - - - - - - - - - - - - - - - - - - -
Kooper        Trade               3000
```

Сақталатын процедураларды SQL Server Eterprise Manager бағдарламалары көмегімен құру келесі әрекеттермен орындалады.

- Microsoft SQL Server бағдарламалар тобынан аталған бағдарламаны жүктеу.
- Ашылған сұқбаттасу терезесінен сервер бағдарламаларының деректер қорын және деректер қорын таңдау.

- Stored Procedures (Сақталатын процедуралар) элементін таңдау және оның контексті мәзірінен New Stored Procedure (сақталатын процедура құру) командасын орындау.
- Ашылған Stored Procedure Properties (сақталатын процедуралар қасиеттері) сұқбаттасу терезесінде Text (Мәтін) өрісінде құрылатын процедуралар Transact-SQL операторын енгізу және процедура атын <PROCEDURE NAME> орнына жазу (11.3-сурет).



11.3- сурет. Сақталатын процедуралар құру терезесі

- Синтаксистік қателіктерді тексеру үшін және қажеттілік кезінде қателіктерді жою мақсатында операторларға түзетулер енгізу үшін Check Syntax (Синтаксистік тексеру) батырмасын басу.
- OK батырмасын басу, нәтижесінде сақталатын процедура құрылады және сақталады.

Триггерлер кестелер деректеріне өзгерістер енгізу кезінде автоматты түрде жүктелетін *сақталатын процедуралардың* бір түрі. Триггерлерді күрделі деректер қорында неғұрлым жиі қолданады, оларды сүзгі ретінде қарастыруға болады.

Триггер және оны шақыратын Transact-SQL операторы біртұтас транзакция ретінде қарастырылады. Әдетте триггер құрамына деректермен қайсібір операцияларды орындау немесе белгілі бір шартты тексеру командаларының жиыны кіреді. Деректерді өңдеу мүмкін емес немесе шарттардың орындалуы мүмкін емес жағдайда транзакцияны болдырмау жүзеге асады.

Триггерлер құру үшін CREATE операторы қолданылады:

```
CREATE TRIGGER [иесі.]триггер_аты
ON [иесі.]{кестелер_аты | көрініс_аты}
[WITH ENCRYPTION]
FOR [{AFTER | INSTEAD OF}]
{INSERT, UPDATE, DELETE}
AS
<SQL_операторлары>
```

WITH ENCRYPTION (шифрлеумен) параметрі триггер мәтінін серверге орналасқаннан кейін оқу мүмкіндіктерін шектейді.

FOR кілттік сөзінен кейін триггер типін көрсетеді: Стандартты (AFTER), пайдаланушымен деректерге өзгерістер енгізгеннен кейінгі жүктелетін, немесе триггерді жүктейтін команданың орнына орындалатын (INSTEAD OF). Үнсіз келісім бойынша AFTER қолданылады.

INSERT (Кірістіру), UPDATE (Жаңарту) және DELETE (Өшіру) кілттік сөздері триггер орындауы қажет операцияларды анықтайды.

SQL Server триггер мәтінін syscomments жүйелік каталогы кестесінде сақтайды.

11.7. Қауіпсіздікті қамтамасыз ету

SQL Server қауіпсіздік жүйелерін шартты түрде екі деңгейге бөлуге болады: сервер және деректер қоры. *Сервер* деңгейінде пайдаланушылардың серверге қатынас құру мүмкіндіктері анықталады. *Деректер қоры* деңгейінде серверді пайдаланушылар үшін деректер қоры объектілеріне қатынас құру құқығы орнатылады.

SQL Server сервер деңгейінде қауіпсіздікті қамтамасыз етудің келесі құралдарын қолданады:

1. Жүйеге кіру кезінде пайдаланушылардың аты бойынша сәйкестендіру (identification);
2. аутентификация (authentication) – пайдаланушының көмегімен құпия сөздің дұрыстығын тексеру;
3. тіркелу жазбасы (login);
4. сервердің кіріктірілген немесе ерекшеленген ролі (fixed server roles).

SQL Server серверде жүйелерді қорғау келесі екі режимде жүзеге асуы мүмкін:

- *стандартты* (Mixed Mode, аралас аутентификация режимі) – SQL Server және Windows NT/2k қорғау құралдарының комбинациясы;
- *біріктірілген* (Windows Authentification Mode, Windows аутентификация режимі) – тек қана Windows NT/2k қорғау құралдарын қолдану.

11.8. Клиент-сервер өзара әрекетін ұйымдастыру

Ескере кетейік, клиент-сервер технологиясын қолданған кезде қосымшалар екі (немесе неғұрлым көп) бөлімдерге бөлінеді. Клиенттік бөлім (Front-end) графикалық тұрғыдан қолайлы интерфейсті қамтамасыз етеді және пайдаланушының компьютерінде орналасады. Серверлік бөлім (Back-end) деректерді басқару, ақпараттарды бөлу, ақпараттардың қауіпсіздігін қамтамасыз етуді жүзеге асырады. Клиенттік қосымшалар деректер қоры серверіне сұраныстар қалыптастырады. Сұраныстың орындалу нәтижелері клиентке жіберіледі.

Үлестірімді ақпараттық жүйелер құру кезінде клиенттік және серверлік бөлімдердің өзара әрекетін ұйымдастыруда келесі маңызды тапсырмаларды ерекшелеу керек:

- болашақта бірлескен деректер қоры ретінде коллективтік пайдалану үшін *дербес деректер қорын серверге көшіру;*
- компьютер-клиент тарапынан серверде орналасқан бірлескен деректер қорына *сұраныстар ұйымдастыру*;
- *клиенттік қосымша құру* үшін компьютер-клиент тарапынан бірлескен деректер қорына шалғайдан қатынас құру;
- клиент тарапынан серверді *басқару.*

Аталған тапсырмаларды неғұрлым жан-жақты Қарастырайық.

Дербес деректер қорын серверге көшіру

Дербес деректер қорын серверге көшіру тапсырмасы деректер қорына коллективтік қатынас құруды қамтамасыз ету талап етілген жағдайда пайда болады. Мұндағы деректер қоры Microsoft Acces немесе Microsoft Visual FoxPro сияқты дербес ДҚБЖ көмегімен құрылған. Microsoft Acces 2002 бағдарламасында Access деректер қорын SQL Server форматына түрлендіруге арналған Upsizing Wizard ("өсіру" шебері) бар. Microsoft Visual FoxPro 8.0 дестесі құрамында да Visual FoxPro деректер қорын SQL Server немесе Oracle форматына түрлендіретін Шебер бар.

Деректер қорына сұраныстар дайындау кезінде өзара әрекеттер әдістері

Серверде (SQL тілінде) деректер қорына сұраныстар дайындау клиенттік бөлімдер тарапынан қайсібір утилиталар, мысалы, Query Analyzer көмегімен орындалуы мүмкін. Сұраныстар құру кезінде пайдаланушыға үлкен мүмкіндіктер және қолайлылық беру үшін клиенттік қосымша құрылады. *Серверлік деректер қорына сұраныстар* ұйымдастыру үшін әртүрлі өзара әрекеттер әдістері қолданылады, олар оның тиімділігін арттырады. Өзара әрекеттердің негізгі әдістеріне келесілерді қолдануға негізделген әдістер жатады:

- DB-Library немесе DB-LIB (деректер қоры кітапханасы) интерфейсі;
- ODBC (ашық деректер қоры үйлесімділігі) технологиясы;
- OLE DB (деректер қоры объектілерін байланыстыру және кірістіру) интерфейсі;
- DAO (Data Acces Object – деректерге қатынас құру объектілері) технологиялары;
- ADO (ActiveX Data Object – деректердің ActiveX объектілері) технологиялары.

Аталған интерфейстер және технологияларды бірлесе қолдану мүмкін, мысалы, ODBC және DAO немесе ODBC, ADO және OLE DB.

Клиент тарапынан серверді басқару

Клиент тарапынан серверді басқару үшін MS SQL Server 2000 клиенттік бөлімдер компоненттері қолданылады, мысалы, SQL Server Enterprise Manager бағдарламасы. Сонымен қатар, осы мақсатта SQL-DMO технологиясы қолданылуы мүмкін.

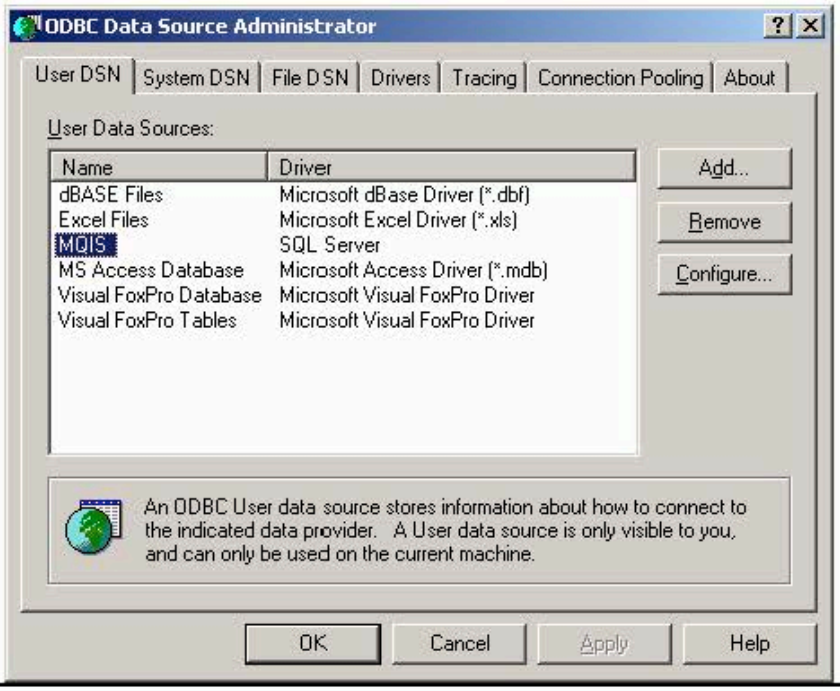
Сондықтан SQL-DMO пайдаланып, қосымша SQL Server Enterprise Manager бағдарламаларымен орындалатын барлық функцияларды орындай алады. SQL-DMO технологиясын қолданған кезде объектілермен, коллекциялармен, SQL-DMO объектісі қасиеттерімен және әдістерімен жұмыс жасалады.

11.9. ODBC көмегімен деректерді өңдеу

SQL Server сервердің деректер қорында өңдеуді ODBC технологиясы көмегімен ұйымдастыруды қарастырайық. ODBC құралдарын қолданған кезде деректердің түпнұсқасына, ал ол арқылы – ДҚБЖ қатынас құру жүзеге асады.

Құру және баптау Деректердің түпнұсқасы

Деректердің түпнұсқасын құру басқару панелі (Control Panel) терезесінен шақырылатын ODBC Data Source Administrator утилитасының көмегімен орындалады. Оның көмегімен (11.4-сурет) жаңа байланыс құруға және байланыстардың параметрлерін өзгертуге болады.



11.4-сурет. Байланыс параметрлерін баптау терезесі

Жаңа байланыс құру үшін ODBC Data Source Administrator утилитасының терезесінде (11.4-сурет) Add… (Кірістіру (қосу)…) батырмасын басып, деректердің түпнұсқасын құру Wizard (Шебер) шеберін шақыру және оның сұқбаттасу терезесінде параметрлерді көрсету керек. Оларға жататындар: деректердің түпнұсқасының аты және сипаттамасы; байланыс орнатылатын сервер; аутентификация әдісі; деректер қоры атауы; үнсіз келісім бойынша деректер қоры атауы және басқалар.

Параметрлерге өзгерістер енгізу үшін ODBC Data Source Administrator утилитасының терезесінде Configure… (Баптау…) батырмасын басып, ашылған сұқбаттасу терезесінен қажетті параметрді таңдау жеткілікті (13.4-сурет).

Бақылау сұрақтары және тапсырмалар

1. MS SQL Server 2000 негізгі жаңалықтарын атап шығыңыз және олардың мағынасын түсіндіріңіз.
2. MS SQL Server 2000 қызметін атаңыз және оларды сипаттаңыз.
3. MS SQL Server 2000 жұмыс режимдері қандай?
4. MS SQL Server 2000 қандай құрал-саймандары бар және олар не үшін арналған?
5. MS SQL Server 2000 жеткізілім нұсқаларын және аппараттық-бағдарламалық қамтамасыз етуге қойылатын талаптарды атаңыз.
6. Домен және топтық жұмыс түсінігін сипаттаңыз.
7. MS SQL Server 2000 жүйелік деректер қорының тағайындалуын түсіндіріңіз және қысқаша сипаттама беріңіз.
8. Деректер қоры каталогы және жүйелік каталог не үшін арналған?
9. MS SQL Server 2000 кесте құру әдістерін атаңыз.
10. Жергілікті және ауқымды уақытша кесте құру және қолдану айырмашылықтары неде?

11. MS SQL Server 2000 деректер типін сипаттаңыз.
12. Кесте құру әдістерін түсіндіріңіз.
13. Атаңыз ережелер неявного меншіктеу мәні полю.
14. Кестелерге деректерді кірістіру немесе қосу және кестеден таңдау қалай жүзеге асады?
15. MS SQL Server 2000 кесте индекстері типтерін сипаттаңыз.
16. SQL Server Enterprise Manager бағдарламалары көмегімен индекс құруды сипаттаңыз.
17. Сақталатын процедураларды сипаттаңыз.
18. Сақталатын процедуралар түрлерін атаңыз және сипаттаңыз.
19. Сақталатын процедураларды құру, қолдану және өшіру қалай жүзеге асады?
20. Триггер дегеніміз не? Олар қалай құрылады?
21. Объектілерге қатынас құру құқығын сипаттаңыз.

12. WEB-ҚОСЫМШАЛАР ТУРАЛЫ НЕГІЗГІ МӘЛІМЕТТЕР

Web-қосымша дегеніміз Web-интерфейсі бар қосымша. Бұл бөлімде біз Web-қосымша архитектурасын, бағдарламалық құралдардың өзара әрекетін және Web-қосымша үшін деректермен алмасуда негізгі хаттама болып табылатын HTTP хаттамасын қысқаша қарастырайық.

12.1. Web-қосымша құрылымы

WEB-қосымша құрылымы бағдарламалық компоненттер құрамы және олардың арасындағы өзара әрекеттер сызбасымен анықталады. Web-қосымшаның жұмысы кезінде пайдаланушының компьютерінде орналасқан шолушы және Web-сервер арасында деректермен алмасуы жүзеге асады.

Шолушы жағынан және Web-сервер жағынан, сәйкесінше шолушының және сервердің мүмкіндіктерін кеңейту бағдарламалары орнатылады. Бұл бағдарламалар Web-қосымшаның қажетті қызметімен қамтамасыз етеді.

12.2. Шолушы және Web-сервердің өзара әрекеттері құралдары

Шолушы және Web-сервердің өзара әрекетін жүзеге асыратын бағдарламалық құралдарға жататындар:

- JavaScript (Netscape), JScript және VBScript (Microsoft) тілдеріндегі сценарийлер (скрипттер);
- CGI интерфейсі негізінде консольдық бағдарламалар;
- WinCGI интерфейсі негізінде орындалатын бағдарламалар;
- ISAPI интерфейсі көмегімен жүзеге асқан динамикалық кітапханалар;
- IDC/HTX динамикалық беттер;
- ActiveX басқару элементтері;
- Java тіліндегі апплеттер және сервлеттер;
- ASP белсенді серверлік беттер.

Сценарийлер Web-құжатты интерфейстік объектілермен (компоненттермен) динамикалық басқару үшін қолданылады. Сценарийлер тілдері *түсіндірілетін* болады, яғни код талдаулар процесінде орындалады. Сценарийлерді талдау және орындауды шолушы немесе Web-сервер жүзеге асырады.

Сценарийлер HTML тілінің кеңейтпесі ретінде қарастырылады және HTML-құжатын қамтуы мүмкін. Сценарий бөлімдері құжатты жүктеу кезінде, ал функциялар түрінде жүзеге асқан сценарий бөлімдері пайдаланушының әрекетіне жауап ретінде орындалуы мүмкін. Сценарийлер тілін таңдау Web-шолушыны ақпараттық-бағдарламалық қамтамасыз етумен анықталады.

ActiveX басқару элементтері клиент жағынан және сервер жағынан қолданылатын кеңейту модулін білдіреді. Модульдер DLL динамикалық кітапхана көмегімен жүзеге асады және Web-құжатқа қосымша интерфейстік элементтер ретінде кіріктіріледі. ActiveX басқару элементтерінің жұмыс механизмі компьютердің жергілікті ресурстарына шектелмеген түрде қатынас құруға мүмкіндік береді.

CGI интерфейсі (Common Gateway Interface – жалпы шлюздік интерфейс) Web-сервер және кеңейту модульдері арасындағы өзара әрекеттердің стандартты хаттамасы болып табылады. CGI интерфейсіне DOS операциялық жүйелерінің *консольдық* қосымшалары сәйкес келеді.

ISAPI/NSAPI (Internet Server API/Netscape Server API) интерфейстері сәйкесінше Microsoft және Netscape фирмаларымен құрылған. Олар Web-сервер үшін кеңейту модульдерін DLL кітапхана түрінде өңдеуге арналған.

Апплеттер (applet – *ағылш.* кішігірім қосымшалар) пайдаланушының динамикалық интерфейсін құру үшін қолданылады. Java тілінің мүмкіндіктері пайдаланушылардың жергілікті ресурстарына қатынас құруы жағынан шектеулі, сондықтан оны желі үшін пайдалану қауіпсіз. Апплеттер кез-келген платформаларда орындалуға арналған.

Сервлеттердің апплеттерден айырмашылықтары - олар сервер жағынан орындалады және шолушыдан жөнелтілетін сұраныстарды өңдеуге арналған. Сервлеттердің артықшылықтары:

- жоғары тиімділігі;
- Java тілін қолданғандықтан, платформалық тәуелсіздік;
- шектеулер қоюмен байланысты жоғары дәрежелі қауіпсіздік;
- Java кластары кітапханаларының қызметтік мүмкіндіктерінің қолжетімділігі.

Сервер және кеңейту модулі арасындағы өзара әрекет арнайы объектілер (Request, Response) көмегімен жүзеге асады. Сервер кеңейту модуліне сұраныс параметрлерін жібереді және сұраныс нәтижесінде қалыптасқан Web-құжат алады, оны шолушыға қайтарады.

12.3. HTTP хаттамасы

HTTP хаттамасы (HyperText Transfere Protocol) шолушы және Web-сервер арасында әртүрлі форматтарда деректер алмасуға арналған.

Компьютер-жөнелтуші және компьютер-қабылдаушы арасындағы байланыс төменгі деңгейдегі TCP/IP (Transport Control Protocol/Internet Protocol – жөнелтулерді басқару хаттамасы/ желі аралық хаттама) хаттамасы көмегімен жүзеге асады.

Шолушы URL көмегімен қажетті беттерді алуға сұраныс қалыптастырады (шолушы терезесінде адрес жолын көрсетеді).

HTTP хаттамасы үшін URL құрылымы:

http://<хост>/<порт>:<жолы>?<іздеу>

Мұнда http – ресурсқа қатынас құру үшін қолданылатын хаттама аты;

<хост> – интернеттен талап етілген Web-түйінді іздеу үшін қолданылатын домен аты;

<порт> – интернетте байланыстардың логикалық каналының номерін көрсететін порт номері;

<жолы> – файлға жергілікті жол;

<іздеу> – сұраныстың қосымша параметрлері.

HTTP хаттамасының негізгі әдістері:

- GET – URL көмегімен көрсетілген ақпаратты алады.
- POST – сервердің URL көмегімен көрсетілген ресурсқа сұранысты өңдеу үшін жөнелтуін талап етеді;
- HEAD – GET сұранысы бойынша қайтарылатын жауап тақырыбын ғана қайтарады;
- PUT – берілген URL сұранысы бойынша сақталатын деректерді серверге береді; POST әдісінен негізгі айырмашылығы - сервер бұдан әрі PUT сұранысын өңдеуге тиіс емес, ол көрсетілген URL бойынша тек қана сайтайды;
- DELETE – URL көмегімен көрсетілген ресурсты өшіреді;
- TRACE – серверден сұраныс хабарламасын қайтаруды талап етеді.

Сұраныс тақырыптары сұраныс немесе клиент туралы қосымша ақпараттарды жөнелтуге арналған. Сұраныс тақырыбы қос нүктемен бөлінген өрістер

идентификаторынан және оның мәнінен тұрады және келесі ақпараттық-бағдарламалық қамтамасыз етулердің бірі болады.

From қамтиды e-mail – пайдаланушының адресі, мысалы:

From: webmaster@yandex.ru

Accept, Accept-Encoding, Accept-Charset және Accept-Language тақырыбында сұраныс жауаптарының форматтары (кодировкалар, символдар және тілдер кестелері) беріледі, мысалы:

Accept: text/plain, text/html, text/x-dvi; q=.8; mxb=100000; mxt=5.0

User-Agent клиенттің бағдарламалар атауын қамтиды.

Referer сұраныс алынған ресурс адресін көрсету үшін қолданылады.

If-Modified-Since желіге түсірілген жүктемені азайту мақсатында, көрсетілген мерзімнен бұрын құжаттарды жөнелтуді болдырмау үшін қолданылады.

Сұраныс мысалы:

GET / HTTP/1.1
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xml
Accept-Encoding: gzip,deflate,sdch
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4
Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) Chrome/17.0

HTTP-сұранысқа *сервер жауабы* жауап кодын қамтиды. Тақырыбы жоқ жауап жазуға болады:

<Response> ::= (<Content>)? | <HTTP-Version> <Code> <Explanation> CrLf <Resp-Header>* (<Content>)?

<Resp-Header> ::= <Field> : <Value> CrLf

CrLf ::= '\r''\n'

Жауап кодтарының мүмкін нұсқалары 12.1-кестеде келтірілген.

12.1-кесте. Жауап кодтары

Тобы	Коды	Код мәні	Түсініктеме
1xx			Ақпараттық хабарлама
2xx			Табысты өңдеу
	200	OK	Бәрі дұрыс
	201	Created	Құжат құрылған
3xx			Сұранысты қайтадан бағыттау
	301	Moved Permanently	Ресурстың орны ауыстырылған
	302	Moved Temporarily	Ресурстың орны уақытша ауыстырылған
4xx			Клиенттің қатесі
	400	Bad Request	Дұрыс емес сұраныс
	401	Unauthorized	Аутентификация қажет
	403	Forbidden	Ресурсқа қатынас құруға тыйым салынған

	404	Not Found	Ресурс жоқ
5xx			Сервер қатесі
	500	Internal Server Error	Сервердің ішкі қатесі

Келесі жауап тақырыптары болуы мүмкін:

Allowed – көрсетілген URL бойынша пайдалануға болатын сұраныс әдістері бос орын арқылы беріледі.

Public – сұраныстардың барлық қолжетімді әдістері беріледі.

Content-Length, Content-Type, Content-Encoding және Content-Language – оның құрамы, форматы немесе MIME-типі, кодировка және тілінің байт түріндегі өлшемі.

Date − құжат немесе объектінің құру мерзімі.

Last-Modified – объектіге соңғы өзгерістер енгізу мерзімі.

Expires − объект ескірген болып саналатын мерзім.

URI – объектінің URI.

Title – құжат тақырыбы.

Server − серверлік бағдарламаларды сипаттау.

Retry-After – сұранысты өңдеуге кететін уақыт аралығы, сол уақыт өткенге дейін жауап болмаса, сұранысты қайталау керек емес.

HTTP-жауап мысалы:

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Content-Type: text/html;charset=utf-8

Expires: 0

Content-Length: 2109

Last-Modified: Tue, 10 Jan 2012 10:11:12 GMT

<!doctype html><html> <head> <title> … </title> </head>

<body> …

</body>

</html>

Келтірілген жауапта құрамы ұзындығы және типі, объектінің есчкіру мерзімі, соңғы өзгерістер енгізілген уақыт көрсетілген.

13. БЕЛГІЛЕУЛЕР ЖӘНЕ СТИЛЬДЕР КЕСТЕСІ ТІЛІ

HTML, XML белгілеулер тілдері және каскадты стильдер кестесі Web-қосымша құру кезінде маңызды роль атқарады – Web-беттердің жекелеген элементтерінің бейнелену форматын анықтауда, Web-қосымша параметрлерін енгізу кезінде және т.б.

13.1. HTML гипермәтіндік белгілеулер тілі

HTML тілі (HypertText Markup Language) интернетте қолданылатын құжаттарды белгілеу тілі болып табылады. HTML-құжат Web-құжаттың бір түрі және жалпы жағдайда мәтіндік құжат (қарапайым ASCII-файлы) болып саналады. HTML-құжатты кез-келген мәтіндік редактордың көмегімен өзгертуге болады. Ол арнайы басқарушы операторлары бар (тегтер) мәтіндік деректерден тұрады.

HTML-құжаттың құрамы және тегтері

Тегтер шолушы үшін қажетті қызметтік ақпараттарды қамтиды және әртүрлі форматтау режимінде берілуі мүмкін. Тегтер бұрыштық жақшаларға алынып жазылады. Мысалы, құжаттың алғашқы тегі келесі түрде жазылады:

<HTML>

Тегтер *жұптасқан* немесе *жұптаспаған* болуы мүмкін. Жұптасқан тегтер *ашатын* және *жабатын* болып бөлінеді. Мұндай тегтердің жұбы *контейнер* деп аталады.

Мысалы, HTML-құжат келесі тегтермен берілсін:

<HTML>

…

<!—Бұл шолушы бейнелемейтін комментарий. --!>

HTML-құжат мәтіні

…

</HTML>

Мұнда <HTML> – ашатын тег, </HTML> – жабатын тег, олардың арасында қайсібір мәтіндік деректер орналасады.

<!– Бұл шолушы бейнелемейтін комментарий. --!> – мысалы, комментарий болып табылатын, бірақ шолушы бейнелемейтін жұптық тег.

HTML-құжат екі бөліктен тұрады – құжаттың тақырыбы және денесі.

Құжаттың тақырыбы міндетті элемент емес болып табылады, ол бірнеше құжаттар арасындағы қатынастарда құжат атауын беру үшін қолданылуы мүмкін және т.б. Құжаттың тақырыбы және денесі арнайы тегтермен қоршалады. Құжаттың денесінде құжат мәтіні және әртүрлі басқарушы тегтер орналасқан.

Тағайындалуы бойынша тегтердің келесі негізгі топтарын ажыратады: құрылымдық, мәтінді форматтау, графикалық бейнелеулер, сілтемелік және дескриптор тегтері.

Негізгі тегтерді қолдануды түсіндірейік және оларды практикада пайдалануға мысалдар келтірейік.

13.2. Құрылымдық тегтер

Құрылымдық тегтер HTML-құжатты негізгі бөлімдерге бөлуге жауапты. Оларға құжат шегі тегтері (<HTML> ... </HTML>), құжаттың тақырыбы тегтері (<HEAD> ... </HEAD>), құжат денесі тегтері (<BODY> ... </BODY>) жатады.

<HTML> контейнерлік типті тег, HTML-құжаттың басын, ал </HTML> тегі құжаттың соңғы жолын көрсетеді. Бұл тегтер олардың арасындағы жолдар ортақ HTML-құжат екендігін білдіреді.

Құжаттың тақырыбын анықтайтын <HEAD> ... <\HEAD> тегтердің ішінде құжат атауын және жалпы ақпаратты беретін басқа тегтер де болуы мүмкін. Мысалы, әдетте шолушы терезесінің тақырыбында көрсетілетін құжат атауы <TITLE> және </TITLE> тегтері арасында мәтіндік жолдар түрінде көрсетіледі.

Құжат денесі <BODY> ... </BODY> тегтері көмегімен келесі түрде беріледі:

<BODY атрибуты>

… құжаттың құрамы…

</BODY>

Мұнда атрибуттар міндетті емес және құжаттың сыртқы түрін анықтайды; құжаттың құрамы – HTML-құжаттың кез-келген ұйғарымды элементтері. Мысалы, тақырыбын және мәтіннің бір жолын қамтитын HTML-құжаты келесі түрде жазылады:

<HTML>
<HEAD> <!-- Бұл құжаттың тақырыбы. --!>
<title> Бұл қарапайым HTML-құжат тақырыбы! </title>
</HEAD>
<BODY> <!-- Бұл құжат денесі. --!>
Бұл қарапайым HTML-құжата мәтіні!
</BODY>
</HTML>

Келтірілген мысал HTML-құжаттың ең аз тегтер жиынын сипаттау болып саналады.

13.3. Мәтінді форматтау тегтері

Мәтінді форматтау тегтері HTML-құжат неғұрлым көрнекі болуы үшін қолданылады. Бұл бөлімде осы мақсатта неғұрлым жиі қолданылатын тегтерді қарастырамыз.

Мәтінді азат жолдарға бөлу <P ALIGN=атрибут>…</P> контейнерлік типті тегі көмегімен жүзеге асады. Бұл тегті қолдану шолушы терезесі параметрлерінің тағайындалуынан тәуелсіз мәтінді азат жолдарға бөлуге мүмкіндік береді. Азат жол тегінде ALIGN атрибуты LEFT, CENTER және RIGHT мәндерін қабылдауы мүмкін, олар мәтінді сәйкесінше сол жағынан, оң жағынан және ортасынан туралайды. Үнсіз келісім бойынша (атрибут берілмеген кезде) мәтінді сол жағынан туралау жүзеге асады.

HTML-құжаттың кез-келген жерінде каретканы (каретка мәтінді енгізу жүзеге асатын ағымдағы орынды көрсетеді) келесі жолға жылжыту
 жолдар айырымы тегімен жүзеге асады. Азат жол тегінен айырмашылығы
 тегі жолды жібермейді.

HTML тақырыбының алты деңгейі бар (қаріп өлшеміне қатысты): H1 (ең жоғарғы), H2, H3, H4, H5 және H6 (ең төменгі). <H1> тегі тапсырмалар тақырыбы, ашатын (<H1 ALIGN=атрибут>) және жабатын (</H1>) тегтері болып табылады. Мысалы, екінші деңгейлі тапсырма тақырыбы қарпінің өлшемі екінші деңгейлі болады:

<H2> файл мәтіні </H2>.

Тақырыптарды туралау азат жолдарды туралау сияқты ALIGN атрибуты көмегімен жүзеге асады.

<HR> тегін көрсеткен кезде сызықтарға дейін және сызықтардан кейін автоматты түрде бос жол қойылады.

<I> тегі контейнерлік болып табылады және мәтінді қиғаш қаріппен ерекшелеу үшін қолданылады, мысалы:

<I> ерекшеленген мәтін </I>

<B> тегі контейнерлік болып табылады және мәтінді жартылай қалыңдатылған қаріппен ерекшелеу үшін қолданылады, мысалы:

<B>Бұл мәтін жартылай қалыңдатылған қаріппен ерекшеленген!</B>

<TT> тегі контейнерлік болып табылады және мәтінді өлшемі белгіленген қаріппен шығару үшін қолданылады, мысалы:

<TT> Бұл мәтін өлшемі белгіленген қаріппен шығарылған! </TT>

<U> тегі контейнерлік болып табылады және мәтіннің астын сызу үшін қолданылады. Мысалы:

<U>асты сызылған мәтін</U>

<FONT> тегі контейнерлік болып табылады және мәтін қарпінің әртүрлі атрибуттарын тағайындау үшін қолданылады. Тег ашылғаннан кейін міндетті түрде атрибуттарды көрсету керек.

FACE атрибуты шолушының сәйкес мәтінін бейнелейтін қаріп түрін тағайындау үшін қолданылады. Қаріп түрі көрсетілмесе, онда үнсіз келісім бойынша тағайындалған қаріп қолданылады, мысалы:

<FONT FACE= "Arial"> Бұл мәтін Arial қарпімен шығарылады </FONT>

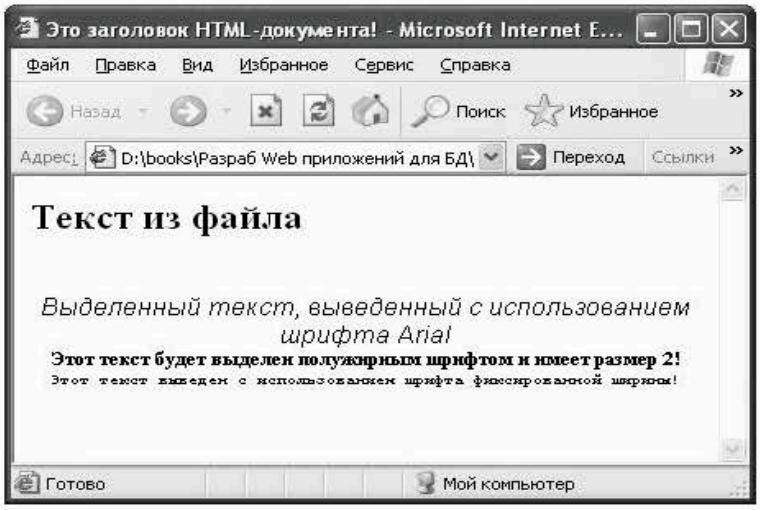
SIZE атрибуты қаріп өлшемін 1-ден 7-ге дейінгі бірліктермен көрсету үшін қолданылады, мысалы:

<FONT SIZE=l> Мәтіннің қолданатын қаріп өлшемі = 1 </FONT>

Қарапайым HTML-құжат мысалы:

```
<HTML>
<HEAD> <!—Бұл құжаттың тақырыбы. --!>
<title> Бұл HTML-құжаттың тақырыбы! </title>
</HEAD>
<body> <!—Бұл құжат денесі. --!>
<H2> Файл мәтіні </H2>
<BR>
<p ALIGN=CENTER>
<FONT FACE= "Arial">
<I> Бұл мәтін Arial қарпімен шығарылады </I>
</FONT>
<BR>
<FONT SIZE=2>
<B> Бұл мәтін жартылай қалыңдатылған қаріппен ерекшеленген және өлшемі 2-ге тең!</B>
<BR>
<TT> Бұл мәтін өлшемі белгіленген қаріппен шығарылған! </TT>
</p>
</body>
</HTML>
```

Сипатталған HTML-құжаттың Microsoft Internet Explorer терезесінде бейнеленуі 13.1-суретте көрсетілген.



13.1-сурет. Шолушы терезесіндегі құжаттың түрі

13.4. Кестелік тегтер

Кесте құру үшін <TABLE> контейнерлік кестелер тегі қолданылады. Бұл тег-контейнерде кестенің құрамы орналасады. <TABLE> тегінде келесі атрибуттар қолданылуы мүмкін:

- BORDER – кестелер жиегін анықтау үшін; жиек ені пиксельмен тағайындалады, мысалы BORDER=2 (үнсіз келісім бойынша 1);
- ALIGN – шолушы терезесінде кестелерді туралау үшін; LEFT (үнсіз келісім бойынша), CENTER және RIGHT мәндерін қабылдайды.

Кестелер құрылымы келесі тег-контейнерлер көмегімен анықталады:

- <TH> – бағаналар (жолдар) тақырыптары беріледі, оларды жартылай қалыңдатылған қаріппен ерекшелеп, ұяшықта ортасынан туралап көрсетеді;
- <TD> – ұяшықтарда деректер беріледі.

Кестелер тапсырмаларының 3 жолдан және 3 бағанадан тұратын мысалдарын келтірейік,:

<TABLE ALIGN=LEFT BORDER=1>

<TR> <TH> 1-бағана тақырыбы </TH><TH> 2-бағана тақырыбы </TH><TH> 3-бағана тақырыбы </TH></TR>

<TR> <TD> 2-жол және 1-бағана деректері.</TD><TD> 2-жол және 2-бағана деректері.</TD><TD> 2-жол және 3-бағана деректері.</TD> </TR>

<TR> <TD> 3-жол және 1-бағана деректері. </TH><TD> 3-жол және 2-бағана деректері.</TD><TD> 3-жол және 3-бағана деректері.</TD> </TR> </TABLE>

13.5 Кадрларды анықтау тегтері

Кадр (фрейм – *ағылш.* frame) дегеніміз шолушы терезесінің қайсібір бөлігі. Кадр FRAMESET (қайсыбір кадрлар блогы) деп аталатын ішкі кадрлар құрылымымен анықталады.

Кадрлар блогын анықтау үшін <FRAMESET> контейнерлік тегі қолданылады:

<FRAMESET атрибут1=мәні… атрибутN=мәні > …Кадрлар тегтері… </FRAMESET>.

Бұл тег-контейнер ішінде бір кадрды ғана анықтайтын тек қана <FRAME> тегтері немесе басқа <FRAMESET> контейнерлері орналасуы мүмкін. Қарастырылған тег

<BODY> тегін алмастырады. FRAMESET құрылымы бар HTML-құжатта <BODY> тегі болмауы керек, әйтпесе кадрлық құрылым жүзеге аспайды.

<FRAMESET> тегінің ROWS және COLS маңызды атрибуттарын қарастырайық. HTML-құжатта кадрлар ретінде жолдар мен бағаналардың кез-келген санын анықтауға болады. ROWS және COLS атрибуттарымен <FRAMESET> тегі келесі түрде жазылады:

<FRAMESET ROWS="мәндер тізімі" COLS=" мәндер_тізімі "> </FRAMESET>

Мұндағы мәндер_тізімі үтір арқылы ажыратылған параметрлерді қамтиды, олар кадрлар өлшемін пиксель, пайыз немесе салыстырмалы бірлікпен анықтайды.

Шолушы кадрлық құрылымды бейнелеуі үшін тізімде ROWS немесе COLS атрибуттары параметрлерінің мәндерін беру керек, мысалы:

<FRAMESET ROWS="50,50,100,100">

…

<FRAMESET COLS="50%,50%">

Мұнда бірінші жол төрт жолдан тұратын, биіктіктері сәйкесінше 50, 50, 100, 100 пиксельді құрайтын кадрлар жиынын анықтайды. Ал, екінші жол екі бағанадан тұратын кадр жиынын анықтайды.

Жолдар биіктігін пиксельмен беру тиімсіз, себебі шолушы терезесінің әртүрлі өлшемі болуы мүмкін. Салыстырмалы бірлікпен жолдар биіктігін анықтау әрқашан орынды, мысалы:

<FRAMESET ROWS="25%,50%,25%">

Егер қосынды 100 % болмаса, онда кадрлар масштабы пропорциональ түрде өзгереді.

Кадрлар параметрін салыстырмалы түрде беру:

<FRAMESET COLS="*,2*,3*">

Мұнда * символы терезені пропорциональ бөлуді білдіреді. Осылайша, терезе үш вертикаль кадрға бөлінген, біріншісінің ені шолушы терезесінің енінің 1/6 бөлігі, екіншісі – 2/6 (немесе 1/3) және үшіншісі –3/6 (немесе 1/2) бөлігі.

Құжатты үш кадрға бөлетін main.htm құжатының кодын қарастырайық:

<HTML>
<HEAD>
</HEAD>
<FRAMESET ROWS="50%,50%">
<FRAME SRC="file1.htm">
<FRAMESET COLS="25%,75%">
<FRAME SRC="file21.htm"> <FRAME SRC="file22.htm">
</FRAMESET>
</FRAMESET> <NOFRAMES>
Шолушы кадрлық құрылымды сүйемелдемейді.
</NOFRAMES> </HTML>

file1.htm файлы келесі ақпаратты қамтиды:

<Font Size=10>Бірінші кадр</Font>

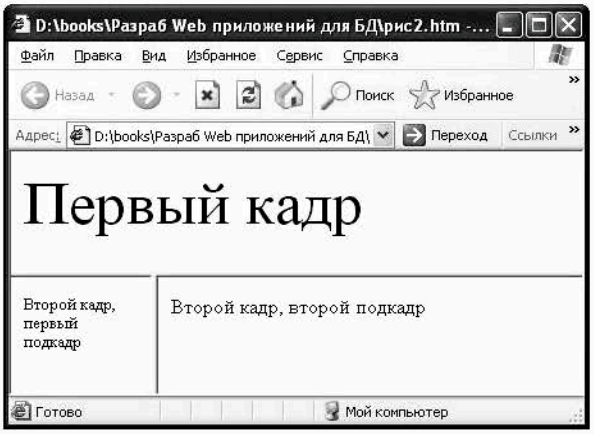
file21.htm файлы келесі ақпаратты қамтиды:

<Font Size=2>Екінші кадр, бірінші ішкі кадр</Font>

file22.htm файлы келесі ақпаратты қамтиды:

<Font Size=3>Екінші кадр, екінші ішкі кадр</Font>

Шолушы терезесінің сыртқы түрі 13.2-суретте келтірілген.



13.2-сурет. main.htm файлын қамтитын шолушы терезесінің түрі

Суретте көрінгендей, құжат үш кадрдан тұрады. Жоғарғы кадр шолушы терезесінің биіктігі бойынша жартысын алып тұр. Төменгі бөлік тағы екі кадрға бөлінген (25% және 75%).

Пайдаланылған әдебиеттер тізімі

1. Кузнецов С. Д. Введение в СУБД: часть 2 // Системы управления базы данных, № 2, 1995. – С. 116-124.
2. Основы современных компьютерных технологий: учебник / Под ред проф. Хомоненко А.Д. – СПб: КОРОНА принт, 2005. – 672 с.
3. Системы управления базами данных и знаний: Справ. изд. / Наумов А.Н., Вендров А.М., Иванов В.К. и др; Под ред. Наумова А.Н. – М.: Финансы и статистика, 1991. – 352 с.
4. Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. База данных : учебник для высших учебных заведений/ Под ред. проф. А. Д. Хомоненко. 5-е изд., доп. – СПб. : КОРОНА принт, 2006. – 736 с.
5. Четвериков В. Н. и т.с.с. Базы и банки данных: учебник для вузов по спец. “АСУ” / Под ред. В. Н. Четверикова. – М.: Высш. шк., 1987. – 248 с.
6. Перри Б. У. Java. Сервлеты және JSP : сборник рецептов / Б. У. Перри. – М. : Кудиц-Пресс, 2006. – 768 с.
1. Дейтел Х. М. Технологии бағдарламалау на Java 2. Кітап 3. Бірлескеные жүйелер, сервлеты JSP, Web-сервис ы / Х. М. Дейтел, П. Дж. Дейтел, С. И. Сантри. – М. : ООО «Бином-Пресс», 2003. – 672 с.
7. Шмитт Ксуреттофер. CSS. Рецепты бағдарламалау = CSS. Cookbook / Шмитт Ксуреттофер. – СПб. : БХВ-Петербург, 2007. – 592 с.
8. Машнин Т. Заманауи JAVA-технологии іс жүзінде / Т. Машнин. – СПб. : БХВ-Петербург, 2010. – 560 с.
9. Мещеряков Е. В. Публикация деректер қоры в Интернете / Е. В. Мещеряков, А. Д. Хомоненко. – СПб. : БХВ-Петербург, 2001. – 560 с.
10. Монахов В. Бағдарламалау тілі Java және орта NetBeans / В. Монахов. – 2-е изд. – СПб. : БХВ-Петербург, 2009. – 712 с.
11. Ноутон П. JAVA 2. Неғұрлым толық руководство / П. Ноутон, Г. Шилдт. – СПб. : БХВ-Петербург, 2002. – 1072 с.
12. Холл М. Бағдарламалау Web. Кітапхана профессионала / М. Холл, Л. Браун ; пер. с англ. – М. : Издательский дом «Вильямс», 2002. – 1264 с.
13. Хомоненко А.Д., Рогальчук В.В., Тырва А.В. Құру Web-қосымша жұмыс жасау деректер қоры үшінмен: Учебное пособие. СПб.: ПГӘӨС, 2012. – 88 с.
14. Хорстман К. Java 2. Кітапхана профессионала. Т. 2. Тонкости бағдарламалау / К. Хорстман, Г. Корнелл. – М. : ООО «И. Д. Вильямс», 2010. – 992 с.
15. Beginning Java EE 5 From Novice to Professional. Kevin Mukhar and Chris Zelenak, with James L. Weaver and Jim Crume. Apress. 2006. – 641 p.

Куандыкова Д.Р., Утепбергенов И.Т.,
Хомоненко А.Д.

Ақпараттық жүйелердегі деректер қоры.

Оқулық

Подписано в печать 24.04.2017 г.
Формат 60х84, 1/16.
Плотность 80 г/м2.
Усл.печ.л. 11,16.
Тираж 1000 шт.

Отпечатано в ТОО «ColorCity».
Наурызбай батыра, 9.
Тел. +7 727 329 24 40