Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Физико-технологический институт

В. Ю. КАРА-УШАНОВ

SQL — ЯЗЫК РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Учебное пособие

Министерство образования и науки Российской Федерации
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

В. Ю. Кара-Ушанов

SQL — язык реляционных баз данных

Учебное пособие

Рекомендовано
методическим советом УрФУ для студентов,
обучающихся по направлению подготовки
230100 «Информатика и вычислительная техника»,
230400 «Информационные системы и технологии»

Екатеринбург
Издательство Уральского университета
2016

УДК 004.43:004.652.4(075.8)
ББК 32.973.2я73
К21

Рецензенты:
кафедра прикладной информатики Института урбанистики Уральской государственной архитектурно-художественной академии (заведующая кафедрой кандидат технических наук, доцент *Г. Б. Захарова*);
кандидат физико-математических наук, доцент *В. П. Кондратьев* (кафедра «Информационные системы и технологии» Уральского технического института связи и информатики (филиал Сибирского государственного университета телекоммуникации и информатики).

Научный редактор — проф., канд. физ.-мат. наук *В. И. Рогович*

Кара-Ушанов, В. Ю.
К21 SQL — язык реляционных баз данных : учебное пособие / В. Ю. Кара-Ушанов. — Екатеринбург : Изд-во Урал. ун-та, 2016. — 156 с.

ISBN 978-5-7996-1622-9

Учебное пособие предназначено для студентов, изучающих в дисциплинах «Базы данных» и «Управление данными» языковые и программные средства создания баз данных и манипулирования данными. Рассматривается реляционная модель данных. Приводятся содержательные определения основных понятий из области проектирования реляционных баз данных. Язык SQL изучается на примере версии Access SQL (Microsoft Jet SQL), поскольку учебное пособие рассчитано на начинающего пользователя. Изучаются подмножества SQL: язык определения данных (DDL), язык манипулирования данными (DML) и язык запросов (DQL). Рассматриваются примеры выполнения команд языка SQL и типы запросов к базе данных. Обсуждаются приемы работы и предлагаются контрольные задания для самостоятельных занятий.

Библиогр.: 13 назв. Рис. 12. Табл. 16. Прил. 1.

УДК 004.43:004.652.4(075.8)
ББК 32.973.2я73

ISBN 978-5-7996-1622-9

Введение

Как известно, для изучения нового языка требуется значительная затрата сил и времени, поэтому стоит остановиться на причинах, которые могут побудить к изучению языка SQL.

SQL (Structured Query Language — язык структурированных запросов) является языком реляционных баз данных, которые уже на протяжении многих лет заслуженно пользуются признанием у разработчиков информационных систем. Сочетание простоты и наглядности основных понятий с их строгим математическим обоснованием обеспечили широкое распространение реляционных баз данных. SQL — это первый и единственный язык работы с базами данных, который получил широкое распространение и поддерживается всеми производителями коммерческих реляционных СУБД.

SQL не является отдельным программным продуктом, подобным офисным приложениям, системам программирования или CASE-системам, его нельзя приобрести в компьютерном магазине. SQL является важнейшим компонентом методологии и технологии реляционных баз данных, неотъемлемой частью реляционной СУБД. Без SQL немыслимы разработка и эксплуатация реляционных баз данных.

В составе СУБД SQL выполняет множество функций. Его можно использовать в интерактивном режиме по принципу «запрос-ответ». SQL является подъязыком баз данных в обстановке прикладной программы, составленной на языке программирования. SQL — это язык запросов в приложениях многопользовательской клиент-серверной системы. SQL — это язык администрирования баз данных. Любому профессионалу, является ли он квалифицированным пользователем базы данных или прикладным программистом или администратором баз данных, необходимо знать язык SQL. Таким образом, *знание языка SQL является обязательным для специалиста в области разработки и сопровождения баз данных*.

SQL создавался в середине 1970-х годов сотрудниками фирмы IBM как высокоуровневый декларативный язык, основанный на реляционном исчислении. Предложения SQL описывают данные, которые необходимо получить, а не определяют способ поиска результата. Особенность предложений этого языка состоит в том, что они ориентированы в большей степени на конечный результат обработки данных, чем на процедуру этой обработки. Исходная версия языка называлась SEQUEL (Structured English QUEry Language — английский структурированный язык запросов) ясно говорит о намерении его авторов создать язык, близкий к естественному. И это им удалось. Команды SQL выглядят как обычные предложения английского языка, что упрощает их изучение и понимание.

Итак, *SQL достаточно легок и понятен для освоения*.

SQL является реляционным языком. Его осмысленное изучение предполагает знакомство с реляционной моделью данных. Основные концепции реляционной модели включают определение структуры и типов данных, определение множества операций, допустимых над элементами структуры данных, и определение правил целостности данных, регулирующих допустимые отношения между элементами структуры данных. Вследствие популярности реляционной модели в публикациях, посвященных методологии проектирования баз данных, всегда можно найти подробное описание реляционной модели (например, в [1–5]). SQL всегда был и остается актуальной проблемой методологии проектирования баз данных. На сей счет также имеется множество публикаций, в том числе учебного характера, как в печатных изданиях [6–8], так и в электронных ресурсах [9, 10]. Таким образом, *самостоятельное изучение SQL на основе обширного методического обеспечения — вполне реальное дело*.

Возникнув как фирменный стандарт, SQL по факту приобрел статус международного языка. Его первая полноценная версия известна как стандарт ANSI SQL-89. Этот стандарт оказался далек от совершенства, имел чрезвычайно общий характер и допускал очень широкое толкование. Опыт использования SQL позволил дополнить первый стандарт новыми возможностями, необходимыми для реализации языка в составе коммерческих СУБД. На этой основе был разработан стандарт ANSI SQL-92, который принято считать истинно реляционным. После его принятия стало возможным говорить про стандартную среду SQL-ориентированной СУБД. В настоящее время этот

стандарт поддерживается практически всеми современными реляционными СУБД. Хотя конкретные реализации, диалекты SQL, поддерживая стандарт в главном, отличаются от него, как правило, в сторону расширения. Начинающему *пользователю SQL полезно знать возможности стандарта*.

Для изучения языка SOL начинающему пользователю предлагается использовать его диалект Access SQL (Microsoft SQL Jet) в интерактивном режиме, свободном от проблем прикладного программирования и знания языков программирования. Выбор Access обусловлен, главным образом, доступностью и распространенностью этого популярного программного продукта.

Реляционная модель данных

Реляционная модель была предложена Э. Ф. Коддом в 1970 году [1] как средство структуризации данных, основанное на строгих математических принципах.

Что такое «модель данных» и в частности «реляционная модель данных»? Прежде чем давать определение понятию модели данных, отметим некоторые коллизии терминологического характера, связанные с термином «модель данных».

Дело в том, что понятие «модель данных» употребляется в двух разных контекстах, отличающихся той ролью, которую играет модель данных в моделировании [11].

Во-первых, модель данных рассматривается как *модель предметной области*, т. е. результат моделирования. И в ней должны быть учтены статические (структурные) и динамические (поведенческие) аспекты предметной области. При такой трактовке понятия «модель данных» объектом моделирования, строго говоря, являются не данные вообще, а конкретная база данных. В таком контексте правильнее говорить не о модели данных вообще, а о модели базы данных конкретной предметной области.

Во-вторых, модель данных часто рассматривается как *совокупность инструментальных средств*, с помощью которых происходит определение и разработка модели данных предметной области как результата моделирования. Определение модели данных в таком контексте выглядит так [2]:

> **модель данных** — это совокупность допустимых структур данных и операций над ними, поддерживаемая компьютерной средой (в т. ч. СУБД), для определения логической структуры базы данных и динамического моделирования состояний предметной области.

В определении понятия модели данных используется термин «допустимые». Речь идет о том, что реальный мир и его отображение в компьютерной среде находятся под влиянием некоторых ограничений (ограничений целостности), которые либо присущи природе реальной действительности, либо связаны с отображением модельных представлений в компьютерной среде.

Итак, в определение модели данных [3] в инструментальном смысле входят:

- определение типа структуры данных;
- определение ограничений целостности, определяющих допустимые отношения между элементами структуры данного типа;
- определение множества операций над данными этого типа.

Это определение модели данных в полной мере применимо и к реляционной модели данных. Итак, термин «реляционная модель данных» употребляется как совокупность инструментальных средств.

Структура данных

Любые данные, используемые в моделировании, имеют свои типы данных. Какие типы данных поддерживает реляционная модель?

В первом приближении принято различать типы данных *простые* и *структурированные* [4].

Простые, или атомарные, типы данных не обладают внутренней структурой. Данные такого типа называют **скалярными**. К простым типам данных относятся, например, следующие типы:

- логический;
- строковый;
- численный.

Этот список можно расширить и уточнить, добавляя другие типы:

- целый;
- вещественный;
- дата;
- время;
- денежный;
- перечислимый;
- интервальный и т. д.

Структурированные типы данных предназначены для задания сложных структур данных. Структурированные типы данных конструируются из составляющих элементов, называемых компонентами, которые, в свою очередь, могут обладать структурой. В качестве структурированных типов данных особенно распространены следующие типы данных:

- массивы;
- записи (структуры).

Общим для структурированных типов данных является то, что они *имеют внутреннюю структуру*. При работе с массивами или записями можно манипулировать массивом или записью как с единым целым (создавать, удалять, копировать целые массивы или записи), так с и их компонентами. Для структурированных типов данных есть специальные функции — конструкторы типов, позволяющие создавать массивы или записи из элементов более простых типов.

Работая же с простыми типами данных, например с числовыми, мы манипулируем ими как неделимыми целыми объектами.

Так вот, реляционная модель требует, чтобы типы используемых данных были простыми. Но что значит «простые»?

Для реляционной модели данных тип используемых данных сам по себе не важен. Требование, чтобы тип данных был *простым*, нужно понимать так, что *в реляционных операциях не должна учитываться внутренняя структура данных* [4]. Конечно, должны быть описаны действия, которые можно производить с данными как с единым целым, например данные числового типа можно складывать, для строк возможна операция конкатенации и т. д.

В реляционной модели данных с понятием типа данных тесно связано понятие домена, которое можно считать уточнением типа данных:

> **домен** — это семантическое понятие. Домен можно рассматривать как подмножество значений некоторого типа данных, имеющее определенный смысл.

Домен характеризуется следующими свойствами:

- имеет уникальное имя;
- определен на некотором простом (скалярном) типе данных или на другом домене;

- может иметь некоторое логическое условие, позволяющее описать подмножество данных, допустимых для данного домена;
- может быть задан перечислением множества допустимых элементов данных;
- несет определенную смысловую нагрузку.

Домен и тип данных — это разные понятия. Домен представляет собой подмножество допустимых элементов данных некоторого типа, имеющее семантически значимое имя. Отличие домена от подмножества данных некоторого типа состоит именно в том, что *домен отражает семантику*, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущих различный смысл. Например, на множестве неотрицательных целых чисел могут быть определены домены «табельный номер», «стаж», «отпуск», «возраст», но смысл этих доменов будет различным, и это будут различные домены.

Понятие домена помогает правильно моделировать предметную область, исключая некорректные сравнения семантически разнородных данных. Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип. В этом проявляется смысловое ограничение доменов.

Единственным средством структуризации данных в реляционной модели является отношение.

Отношение (по-английски relation, отсюда происходит название модели) — это множество со специфическими свойствами.

В теории множеств **отношением *R*** называется подмножество декартова произведения множеств D_j:

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n = \{\langle d_1, d_2, \ldots, d_n \rangle \mid d_1 \in D_1 \wedge d_2 \in D_2 \wedge \ldots \wedge d_n \in D_n\}.$$

Отношение представляет собой множество n-арных *кортежей* типа $\langle d_1, d_2, \ldots, d_n \rangle$ (n — число множеств-сомножителей D_j). Кортежи образуются из элементов множеств D_j по одному из каждого в заданном порядке.

Пример декартова произведения множеств $A = \{a_1, a_2\}$ и $B = \{b_1, b_2, b_3\}$:

$$A \times B = \{\langle a_i, b_j \rangle \mid a_i \in A \wedge b_j \in B\} =$$

$$= \{\langle a_1, b_1 \rangle, \langle a_1, b_2 \rangle, \langle a_1, b_3 \rangle, \langle a_2, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_2, b_3 \rangle\}.$$

В математике отношение — не более чем абстрактный объект. В моделировании данных отношение наполняется содержательным смыслом и применяется для определения объектов реальной действительности и связей между ними.

В теории данных отношение определено на доменах D_j. **Домен** представляет собой именованное множество элементов данных скалярного типа.

В моделировании данных домен играет роль **области определения атрибутов** A_j — спецификаторов свойств моделируемых объектов и связей между ними. Атрибутам как спецификаторам свойств сущностей или связей присваиваются семантически значимые имена, как правило, в форме существительного. Причем на одном домене могут быть определены несколько атрибутов, но любой атрибут может быть определен только на одном домене.

Если отношение моделирует тип объекта или тип связи, то его элементы — **кортежи** — представляют экземпляры объектов или связей.

Наглядным образом отношения является *реляционная таблица*. Отношение, подобно таблице, содержит заголовок и тело. Заголовок отношения представлен конечным множеством атрибутов. Атрибуты A_j определены на соответствующих доменах D_j и в заголовке представлены своими содержательными именами.

Тело отношения содержит множество кортежей-строк. В каждом i-кортеже для каждого j-атрибута имеется одно единственное значение, принадлежащее j-домену.

Максимальное число строк-кортежей называется **кардинальным числом** (мощностью) отношения. Число столбцов-атрибутов называется **степенью отношения**.

Отношение, точнее, переменная-отношение, имеет имя, допустим R. Атрибуты, определенные на доменах, также имеют имена, например $A_1, A_2, \ldots, A_n$. Заголовок отношения иначе называется схемой отношения. *Схема отношения* задается именем отношения и именами атрибутов $R(A_1, A_2, \ldots, A_n)$. *Экземпляр отношения*, или значение переменной-отношения, образует тело отношения (реляционную таблицу).

Набор взаимосвязанных отношений называется **реляционной базой данных.** А набор заголовков (схем) отношений входящих в базу данных называется **схемой реляционной базы данных**.

Манипулируя реляционными таблицами, следует помнить, что таблица и отношение — не синонимы. Отношение — это множество

со специфическими свойствами, а таблица — это наглядный образ отношения. Табличное представление реляционной базы данных будет корректно, если иметь в виду правила интерпретации элементов таблицы, задающие соответствие между реляционной и табличной терминологиями (табл. 1).

Таблица 1

Соотношение реляционной и табличной терминологии

Реляционная модель данных	Табличное представление
Отношение	Реляционная таблица
Атрибут	Имя столбца
Кортеж	Строка реляционной таблицы
Домен	Множество допустимых значений для столбца
Степень отношения	Число столбцов реляционной таблицы
Мощность отношения	Число строк реляционной таблицы
Реляционная база данных	Совокупность взаимосвязанных таблиц

Ограничения целостности

Целостность структуры данных является синонимом ее системности. Обеспечение целостности данных гарантирует их системную полноту, адекватность модели данных моделируемой предметной области.

Структура базы данных определяется не только составом образующих ее информационных элементов, но и характером связей между ними. Связи соответствуют зависимостям между компонентами предметной области. Всякие зависимости представляют собой ограничения на возможные отношения элементов системы.

Ограничения целостности представляют собой условия, которые определяют допустимые отношения между элементами структуры данных. Для конкретной модели данных эти условия выполняются или не выполняются. Средством спецификации ограничений целостности является язык математической логики.

Ограничения целостности данных принято классифицировать.

По происхождению ограничения целостности принято различать:

- **внутренние**, обусловленные особенностью типа структуры данных, в частности отношения;
- **семантические** (явные), обусловленные смыслом, значением взаимосвязанных данных конкретной предметной области.

По способу контроля целостности данных, который осуществляет реляционная система (СУБД) ограничения целостности, принято различать:

- **безотлагательные**, проверка которых осуществляется непосредственно в процессе манипулирования данными;
- **отложенные**, проверка которых совершается по завершении всех манипуляций со связанными таблицами.

Внутренние ограничения целостности

Внутренние ограничения целостности данных реляционной модели обусловлены свойствами отношения по определению как множества:

- у отношения не может быть одинаковых строк-кортежей;
- порядок следования кортежей значения не имеет;
- порядок следования атрибутов значения не имеет;
- все значения атрибутов атомарны (неделимы).

Уникальность строк-кортежей реляционной таблицы является следствием определения множества в математике. Действительно, тело отношения есть множество кортежей и, как всякое множество, не может содержать неразличимые элементы.

Неупорядоченность строк-кортежей также является следствием определения множества в математике. Тело отношения есть множество, а множество не упорядочено. Одно и то же отношение может быть представлено разными таблицами, отличающимися порядком следования строк. Иными словами, таблицы, отличающиеся только порядком следования строк, считаются эквивалентными.

Неупорядоченность атрибутов обусловлена тем обстоятельством, что каждый атрибут имеет уникальное имя в пределах отношения и порядок атрибутов не имеет значения. Одно и то же отношение может быть представлено разными таблицами, в которых столбцы идут в различном порядке. Иными словами, таблицы, отличающиеся

только порядком следования столбцов, считаются эквивалентными.

Условие атомарности атрибутов следует из определения реляционной модели. Атрибуты могут быть определены на простых типах данных (доменах), для которых в реляционных операциях не должна учитываться внутренняя структура данных.

Итак, из определения внутренних ограничений целостности следует, что две и более таблиц могут считаться эквивалентными при выполнении следующих условий:

- таблицы имеют одинаковое количество столбцов;
- таблицы содержат столбцы с одинаковыми наименованиями;
- столбцы с одинаковыми наименованиями содержат данные из одних и тех же доменов;
- таблицы имеют одинаковые строки с учетом того, что порядок столбцов может различаться.

Внутренние ограничения целостности являются *безотлагательными*. Их проверка реляционной СУБД выполняется автоматически.

Семантические ограничения целостности

Семантические ограничения целостности (или семантические условия) присущи самой предметной области и их учет основан на понимании смыслового содержания данных. Семантические ограничения целостности не выводятся. Это свойства данных, которые выполняются или не выполняются для рассматриваемого отношения элементов данных.

Семантические ограничения целостности для каждой предметной области будут свои, что представляет трудности для аналитика. Тем не менее классификация возможна и полезна. Семантические ограничения целостности различают:

- ограничения объектов;
- ограничения на ссылки (связи) между объектами.

В реляционном представлении ограничения целостности объектов реализуются в виде ограничений целостности доменов, атрибутов и отношений.

Ограничения ссылочной целостности или ограничения, основанные на связях между отношениями-объектами, проявляются как ограничения целостности реляционной базы данных в целом.

Ограничения целостности домена (атрибута)

Ограничениями целостности домена является определение множества элементов данных, которые образуют этот домен. В определение домена входит определение типа данных, образующих домен, а также множество допустимых значений элементов домена. Множество допустимых значений может быть задано либо перечислением (списком), либо при помощи порождающего условия, например:

- «пол» {мужской, женский};
- «фамилия» как множество строк символов длиной не более 20;
- «оценка» как множество целых чисел в диапазоне от 2 до 5 и т. д.

Поскольку сами домены никогда не обновляются в процессе функционирования базы данных, то ограничения целостности доменов не проверяются. Обновляются значения атрибутов, определенных на доменах. Поэтому проверяются *ограничения целостности атрибутов* на предмет соответствия типу данных или множеству допустимых значений, причем проверяются всегда и немедленно.

Ограничения целостности отношения

Ограничения целостности отношения базируются на понятиях функциональных зависимостей атрибутов, потенциальных ключей, неопределенных так называемых Null-значениях атрибутов и др.

Зависимости одних атрибутов от других являются очень распространенным типом ограничений. Особое значение имеют зависимости от так называемых *потенциальных ключей*. Строки-кортежи отношения соответствуют реальным экземплярам объектов, а реальные объекты различимы, идентифицируемы. Это значит, что среди атрибутов отношения всегда найдется такое их подмножество (в том числе хотя бы один или наоборот все), которое однозначно определяет любой другой атрибут отношения, а значит, и все отношение в целом. Такие комбинации атрибутов, которые, как говорят, функционально полно определяют другие атрибуты, а значит, и весь кортеж в целом, называются потенциальными ключами отношения.

> **Потенциальным ключом** называется подмножество атрибутов, которое *функционально полно* определяет значение любого атрибута.

Термин «функциональная зависимость» означает однозначную зависимость от ключа. Термин «функционально полная зависимость» означает, что никакое подмножество ключа не может выступать в роли ключа. Среди потенциальных ключей выбирают по соображениям простоты манипулирования один *первичный ключ*, остальные ключи называются *альтернативными*.

Основными свойствами потенциального ключа являются:

- уникальность — не может быть в отношении двух и более кортежей с одинаковыми значениями ключа;
- неизбыточность — никакое подмножество ключа не может выступать в роли ключа;
- обязательность (определенность) — ни при каких условиях атрибуты ключа не могут принимать неопределенные (Null) значения.

Например, понятие «номер сотрудника» в роли потенциального первичного ключа как ограничение означает, что не может быть двух и более персон с одинаковыми учетными номерами.

Потенциальный ключ, состоящий из одного атрибута, например «номер сотрудника», называется *простым*. Потенциальный ключ, состоящий из нескольких атрибутов, например «серия паспорта» и «номер паспорта», называется *составным*. Атрибуты составного ключа могут быть неуникальными (допускаются дубли), но составной ключ в целом уникален всегда.

В качестве первичного ключа часто используется так называемый *искусственный ключ*, который никак не связан с содержанием данных. По существу это уникальный, неизменяемый идентификатор кортежа (порядковый номер строки реляционной таблицы). Преимущества искусственного ключа перед естественным проявляется в случае реструктуризации базы данных на основе связанных таблиц. Искусственный ключ всегда бывает один.

Значение Null используется в реляционной модели при решении проблемы отсутствующей информации. Null — это не одно и то же, что пробелы или числовые нули. Значения Null могут быть как значения атрибутов, для которых данные неизвестны. Null — значение для заданного атрибута может быть разрешено или запрещено.

Поскольку потенциальные ключи фактически служат идентификаторами объектов предметной области, то значения этих идентификаторов не могут содержать неизвестные значения. В противном

случае нельзя было бы определить, совпадают или нет два идентификатора. Отсюда следует правило целостности отношений: атрибуты, входящие в состав некоторого потенциального ключа, не могут принимать Null-значений.

Ограничения целостности отношений, как и ограничения целостности атрибутов, являются безотлагательными.

Ограничения ссылочной целостности

Бизнес-правила использования данных конкретной предметной области выражаются ограничениями ссылочной целостности. Последние контролируют корректность реляционной базы данных, образованной несколькими связанными отношениями (таблицами). Ограничения ссылочной целостности — это правила, контролирующие корректность связей между отношениями.

Связи между отношениями различают:

- по типу зависимости:
  - — категориальные, на основе абстракции обобщения;
  - — бинарные (в общем случае *n*-арные), на основе абстракции агрегации;
- по силе зависимости:
  - — неидентифицирующие;
  - — идентифицирующие.

По роли в образовании связи различают родительское отношение (parent) и отношение-потомок (child). Связь обычно имеет направление от родителя к потомку.

Связь между отношениями моделируется при помощи внешнего ключа.

Внешний ключ — это атрибут отношения-потомка, доставшийся ему по наследству от родительского отношения.

> **Внешний ключ** отношения-потомка — это потенциальный (часто первичный) ключ родительского отношения, мигрировавший в отношение-потомок и там используемый для моделирования связи.

По существу, внешний ключ представляет собой ссылку на родительское отношение.

Таким образом, потенциальный ключ родительского отношения и внешний ключ отношения-потомка представляют собой атрибуты связи. Часто они имеют одинаковые имена, хотя совпадение имен необязательно. Важно, чтобы атрибуты связи были определены на одном и том же или совместимых доменах.

Анализ структуры данных ведется на уровне схем отношений, при этом фактическое наполнение реляционных таблиц значения не имеет.

Бинарная связь со степенью «один ко многим» между родительским отношением, например, «Отдел» и отношением-потомком «Сотрудник» (рис. 1) называется **неидентифицирующей**, если кортежи потомка идентифицируются его собственным первичным ключом «номер сотрудника», а не ключом связи с родительским отношением «номер отдела». При этом внешний ключ отношения-потомка «номер отдела» не входит в состав первичного ключа «Сотрудник» и не используется для идентификации его кортежей, а используется только для моделирования связи.

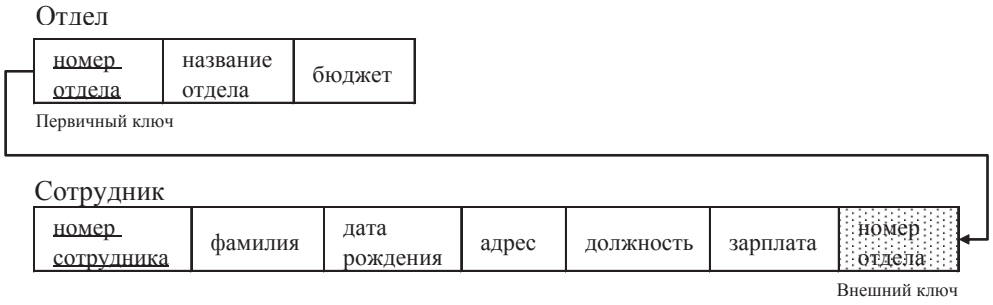

Рис. 1. Неидентифицирующая связь

Бинарная связь со степенью «один ко многим» между родительским отношением, например, «Проект» и отношением-потомком «Задание» (рис. 2) называется **идентифицирующей**, если кортежи потомка идентифицируются не только его собственным первичным ключом «номер задания», но и ключом связи с родительским отношением «номер проекта». При этом внешний ключ отношения-потомка «номер проекта» входит в состав первичного ключа «Задание» и используется не только для моделирования связи с родительским отношением, но и для идентификации его кортежей.

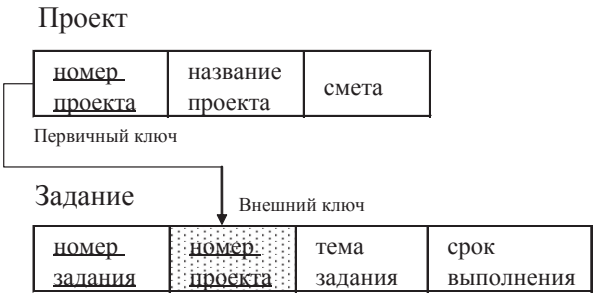

Рис. 2. Идентифицирующая связь

Категориальная связь со степенью «один к одному» на основе обобщения (связь с семантикой «есть» (is a)): «Сотрудник есть Штатный», «Сотрудник есть Совместитель», «Сотрудник есть Консультант» — осуществляется по атрибуту «номер сотрудника». Обобщающее отношение «Сотрудник» объединяет общие атрибуты, которые наследуются категориями.

Категории «Штатный», «Совместитель», «Консультант» содержат отличительные характеристики — атрибуты.

Категориальная связь в реляционном представлении моделируется при помощи трех (в данном случае) **идентифицирующих связей** (рис. 3). При этом внешний ключ каждого отношения-потомка «номер сотрудника» играет роль первичного ключа и используется не только для моделирования связи с родительским отношением, но и для идентификации кортежей отношений потомков.

Бинарные связи «многие ко многим» (*n*-арные связи в общем случае) моделируются при помощи дополнительного ассоциативного отношения. Отношение «Исполнитель проекта» представляет связь между отношениями «Сотрудник» и «Проект» (рис. 4).

Ассоциативное отношение-потомок «Исполнитель проекта» имеет двух родителей и связано с ними *идентифицирующими* связями, а в качестве первичного ключа имеет *составной ключ*, построенный из внешних ключей.

Ограничения ссылочной целостности — это ограничения на выполнение корректирующих операций вставки, обновления и удаления данных в связанных отношениях (таблицах). Ссылочная целостность означает одно из двух:

- реляционная база данных *не должна содержать несогласованных значений внешних ключей*, для которых не существует со-

ответствующего значения первичного ключа родительского отношения;

- *значение внешнего ключа может быть неопределенным* (т. е. ни на что не указывать).

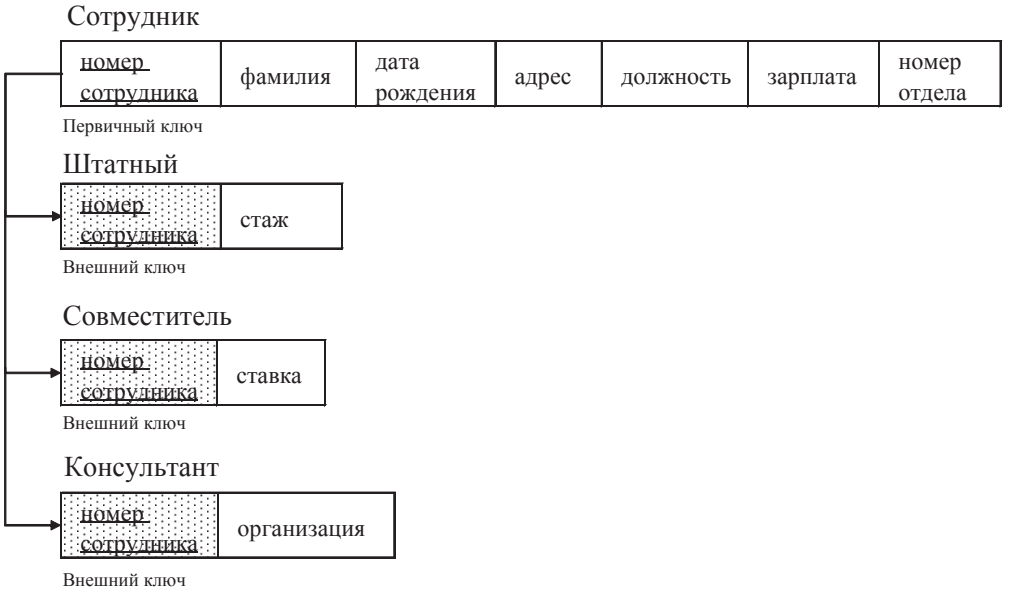

Рис. 3. Категориальная связь

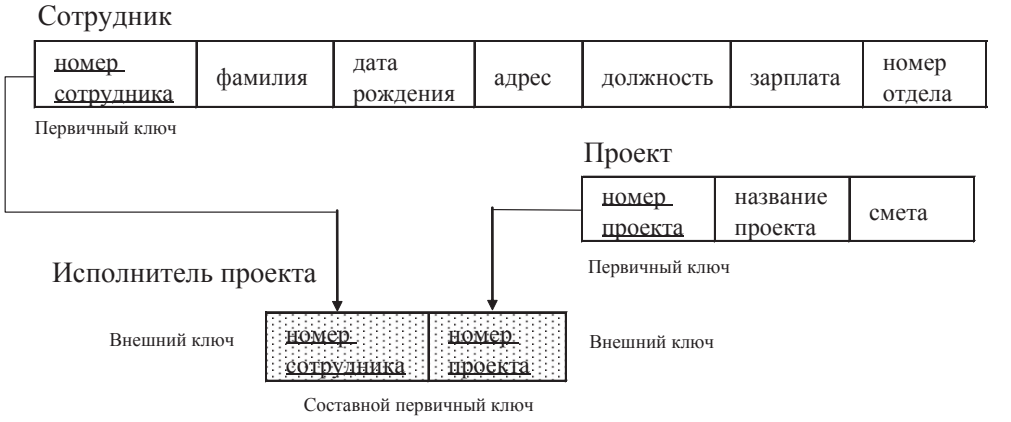

Рис. 4. Связь «многие ко многим»

Реакция на выполнение корректирующих операций над кортежами связанных отношений зависит от типа операции (вставка, обнов-

ление или удаление) и от типа связи (идентифицирующая, неидентифицирующая или категориальная) и может осуществляться в виде:

- запрета;
- каскадирования;
- установки в Null.

Для родительского отношения возможны следующие результаты корректирующих операций:

- при вставке кортежа в родительское отношение возникает новое значение потенциального ключа. Так как допустимо существование кортежей в родительском отношении, на которые нет ссылок из отношения-потомка, то *вставка кортежа в родительском отношении не нарушает ссылочной целостности*;
- при обновлении кортежа в родительском отношении может измениться значение потенциального ключа. Если есть кортежи в отношении-потомке, ссылающиеся на обновляемый кортеж, то значения их внешних ключей станут некорректными. *Обновление кортежа в родительском отношении может привести к нарушению ссылочной целостности*, если это обновление затрагивает значение потенциального ключа;
- при удалении кортежа в родительском отношении удаляется значение потенциального ключа. Если есть кортежи в отношении-потомке, ссылающиеся на удаляемый кортеж, то значения их внешних ключей станут некорректными. *Удаление кортежей в родительском отношении может привести к нарушению ссылочной целостности*.

Для отношения-потомка возможны следующие результаты корректирующих операций:

- нельзя вставить кортеж в отношение-потомок, если вставляемое значение внешнего ключа некорректно. *Вставка кортежа в отношение-потомок может привести к нарушению ссылочной целостности*;
- при обновлении кортежа в отношении-потомке есть возможность некорректно изменить значение внешнего ключа. *Обновление кортежа в отношении-потомке может привести к нарушению ссылочной целостности*;
- удаление кортежа в отношении-потомке при условии, что оно не является родителем в другой связи, выполняется без про-

блем. *Удаление кортежа в отношении-потомке не нарушает ссылочную целостность.*

Итак, к нарушению ссылочной целостности могут привести:

- обновление кортежа в родительском отношении;
- удаление кортежа в родительском отношении;
- вставка кортежа в отношение-потомок;
- обновление кортежа в отношении-потомке.

Выбор адекватного способа поддержания целостности при выполнении операций манипулирования данными определяется семантикой связей, бизнес-правилами предметной области.

При обновлении кортежа в родительском отношении возможны:

- *запрет* — не разрешается обновление первичного ключа в кортеже, если имеется хотя бы один кортеж в отношении-потомке, ссылающийся на обновляемый кортеж;
- *каскадирование* — разрешается изменять значение первичного ключа в кортеже родительского отношения, если в отношении-потомке есть ссылающиеся на него кортежи. При этом будут изменены «по каскаду» значения внешних ключей во всех кортежах отношений-потомков, ссылающихся на обновляемый кортеж;
- *установка в Null* — актуальна для неидентифицирующих связей, когда внешний ключ отношения-потомка является необязательным атрибутом и может быть определен вне зависимости от значения первичного ключа соответствующего кортежа родительского отношения. В случае обновления первичного ключа в кортеже родительского отношения внешний ключ кортежей отношения-потомка может быть установлен в Null.

При удалении кортежа в родительском отношении возможны:

- *запрет* — не разрешается удаление кортежа, если имеется хотя бы один кортеж в отношении-потомке, ссылающийся на удаляемый кортеж;
- *каскадирование* — разрешается удаление кортежа родительского отношения, если в отношении-потомке есть ссылающиеся на него кортежи. При этом будут удалены «по каскаду» все кортежи отношений-потомков, ссылающихся на удаляемый кортеж;
- *установка в Null* — актуальна для неидентифицирующих связей, когда внешний ключ отношения-потомка является необя-

зательным атрибутом и может быть определен вне зависимости от значения первичного ключа соответствующего кортежа родительского отношения. В этом случае разрешается удаление кортежа родительского отношения, но во всех кортежах отношения-потомка, ссылающихся на удаляемый кортеж, внешний ключ может быть установлен в Null.

При вставке кортежа в отношении-потомке возможны:

- *запрет* — не разрешается вставка, если внешний ключ во вставляемом кортеже не соответствует ни одному значению первичного ключа родительского отношения;
- *установка в Null* — актуальна для неидентифицирующих связей, когда внешний ключ отношения-потомка является необязательным атрибутом и может быть определен вне зависимости от значения первичного ключа соответствующего кортежа родительского отношения. В этом случае разрешается вставка кортежа, но в качестве значения внешнего ключа отношения-потомка следует установить Null-значение.

При обновлении кортежа в отношении-потомке возможны:

- *запрет* — не разрешается обновление, если внешний ключ в обновляемом кортеже становится не соответствующим ни одному значению первичного ключа родительского отношения;
- *установка в Null* — актуальна для неидентифицирующих связей, когда внешний ключ отношения-потомка является необязательным атрибутом и может быть определен вне зависимости от значения первичного ключа соответствующего кортежа родительского отношения. В этом случае разрешается обновить кортеж, но в качестве значения внешнего ключа отношения-потомка следует установить Null-значение.

Операции над отношениями

Доступ к реляционным данным осуществляется при помощи реляционной алгебры или эквивалентного ему реляционного исчисления. Реляционная алгебра представляет собой набор операторов, использующих отношения в качестве операндов и возвращающих результат также в виде отношений. Это свойство, называемое зам-

кнутостью реляционной алгебры, позволяет в реляционных выражениях использовать вложенные выражения сколь угодно сложной структуры.

Реляционная алгебра состоит из восьми операторов, составляющих две группы по четыре в каждой.

- теоретико-множественные:
  — объединение;
  — пересечение;
  — вычитание;
  — декартово произведение;
- реляционные:
  — проекция;
  — селекция;
  — соединение;
  — деление.

Теоретико-множественные операции объединения, пересечения и вычитания имеют ту специфику, что они определены на отношениях. Это значит, что элементами множеств являются кортежи, причем кортежи должны быть совместимыми по типу.

Совместимыми по типу называются отношения, которые имеют одну и ту же структуру (идентичные заголовки), а именно:

- каждое отношение имеет одно и то же множество имен атрибутов (одну и ту же степень);
- одноименные (соответствующие) атрибуты определены на одном и том же домене (должны иметь один и тот же тип и размер).

Объединение

Объединением двух совместимых по типу отношений R_1 и R_2 называется отношение R с тем же заголовком, что и у отношений R_1 и R_2, и телом, состоящим из кортежей, принадлежащих или R_1, или R_2, или обоим отношениям (рис. 5).

Пусть имеются два подмножества: таблица, представляющая список сотрудников, работающих в отделе 102 (R_1), и список сотрудников-инженеров (R_2). Оба подмножества должны быть совместимы по типу. Тогда результатом объединения этих двух списков будет спи-

сок сотрудников, которые или работают в отделе 102, или являются инженерами по должности, или удовлетворяют обоим условиям одновременно.

Синтаксис операции

$$R = R_1 \cup R_2 = \{\langle r \rangle \mid r \in R_1 \vee r \in R_2\}$$

Диаграмма Венна

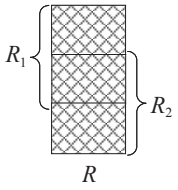

Рис. 5. Операция объединения

Сотрудники-инженеры (R_1)

номер сотрудника	**фамилия**	**должность**	**номер отдела**
1001	Иванов	инженер	101
1002	Петров	инженер	102
1004	Волков	инженер	101

Сотрудники 102 отдела (R_2)

номер сотрудника	**фамилия**	**должность**	**номер отдела**
1002	Петров	инженер	102
1005	Белов	техник	102
1006	Кравцов	оператор	102

Сотрудники (R)

номер сотрудника	**фамилия**	**должность**	**номер отдела**
1001	Иванов	инженер	101
1002	Петров	инженер	102
1004	Волков	инженер	101
1005	Белов	техник	102
1006	Кравцов	оператор	102

Вычитание

Вычитанием двух совместимых по типу отношений R_1 и R_2 называется отношение R с тем же заголовком, что и у отношений R_1 и R_2, и телом, состоящим из кортежей, принадлежащих отношению R_1 (уменьшаемому) и не принадлежащих отношению R_2 (вычитаемому, рис. 6).

Синтаксис операции

$$R = R_1 - R_2 = \{\langle r\rangle \mid r \in R_1 \wedge r \notin R_2\}$$

Диаграмма Венна

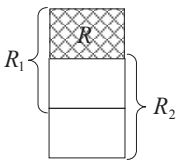

Рис. 6. Операция вычитания

Допустим, исходные данные те же, что и в предыдущем примере, и нужно сформировать список сотрудников, которые являются инженерами, но не работают в отделе 102.

Сотрудники-инженеры не из отдела 102

номер сотрудника	фамилия	должность	номер отдела
1001	Иванов	инженер	101
1004	Волков	инженер	101

Пересечение

Пересечением двух совместимых по типу отношений R_1 и R_2 называется отношение R с тем же заголовком, что и у отношений R_1 и R_2, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям R_1 и R_2 (рис. 7).

Синтаксис операции

$$R = R_1 \bigcap R_2 = \{\langle r\rangle \mid r \in R_1 \wedge r \in R_2\}$$

Диаграмма Венна

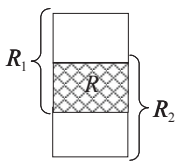

Рис. 7. Операция пересечения

Операция пересечения является зависимой и может быть выражена через другие операции:

$$R = R_1 \cap R_2 = R_1 - (R_1 - R_2).$$

Допустим, исходные данные те же, что и в предыдущем примере, и нужно сформировать список сотрудников, которые являются инженерами и работают в отделе 102.

Сотрудники-инженеры отдела 102

номер сотрудника	**фамилия**	**должность**	**номер отдела**
1002	Петров	инженер	102

Декартово произведение

Декартовым произведением двух отношений R_1 (со степенью n_1 и мощностью k_1) и R_2 (со степенью n_2 и мощностью k_2) называется отношение R, заголовок которого является сцеплением заголовков отношений R_1 и R_2, а тело состоит из кортежей, являющихся сцеплением кортежей отношений R_1 и R_2.

Синтаксис операции декартового произведения:

$$R = R_1 \times R_2 = \{\langle r_1, r_2 \rangle | r_1 \in R_1 \wedge r_2 \in R_2\}.$$

Степень отношения R:

$$n = n_1 + n_2.$$

Мощность отношения R:

$$k = k_1 \cdot k_2.$$

Таким образом, кортежи декартового произведения представляют собой сцепление кортежей отношений-операндов во всех возможных сочетаниях.

Само по себе декартово произведение в моделировании данных не имеет никакого содержательного смысла, но представляет интерес как источник данных, комбинируемых из нескольких связанных отношений.

Допустим, что при описании некоторой предметной области таблица «Сотрудники» представляет список сотрудников, а таблица «Отделы» — список отделов, где они работают.

Сотрудники (R_1)

номер сотрудника	фамилия	должность
1001	Иванов	инженер
1002	Петров	инженер
1003	Смирнов	техник

Отделы (R_2)

номер отдела	название отдела
101	ОГК
102	ОГТ

Декартово произведение (R)

номер сотрудника	фамилия	должность	номер отдела	название отдела
1001	Иванов	инженер	101	ОГК
1001	Иванов	инженер	102	ОГТ
1002	Петров	инженер	101	ОГК
1002	Петров	инженер	102	ОГТ
1003	Смирнов	техник	101	ОГК
1003	Смирнов	техник	102	ОГТ

Проекция

Проекцией отношения R_1 (A, B, C, D, E) на атрибуты A, C, E, где множество $\{A, C, E\}$ является подмножеством полного списка атрибутов заголовка отношения R_1 $\{A, B, C, D, E\}$, называется отношение R с заголовком A, C, E и телом, содержащим кортежи отношения R_1 без дублей (рис. 8).

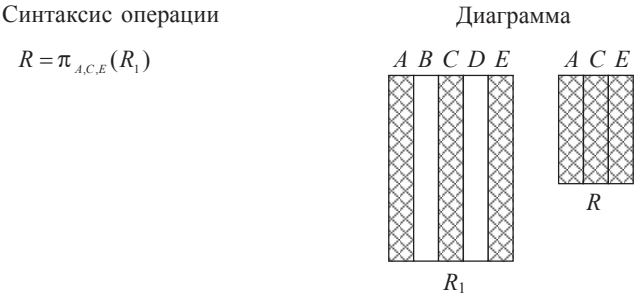

Рис. 8. Операция проекции

Операция проекция порождает результат как вертикальную выборку, выборку столбцов-атрибутов реляционной таблицы без дублей строк-кортежей.

Допустим, что из отношения «Сотрудники» формируется как проекция отношение «Штатное расписание».

Сотрудники (R_1)

номер сотрудника	**фамилия**	**должность**	**зарплата**	**номер отдела**
1001	Иванов	техник	8000	101
1001	Иванов	инженер	15000	101
1002	Петров	инженер	15000	102
1003	Смирнов	техник	8000	101
1003	Смирнов	инженер	15000	101
1004	Волков	инженер	15000	101

Штатное расписание (R)

должность	**зарплата**
техник	8000
инженер	15000

Селекция

Селекцией отношения R_1 по формуле F называется отношение R, имеющее тот же заголовок, что и отношение R_1, и тело, содержащее кортежи отношения R_1, которые удовлетворяют истинности логического выражения, заданного формулой F (рис. 9).

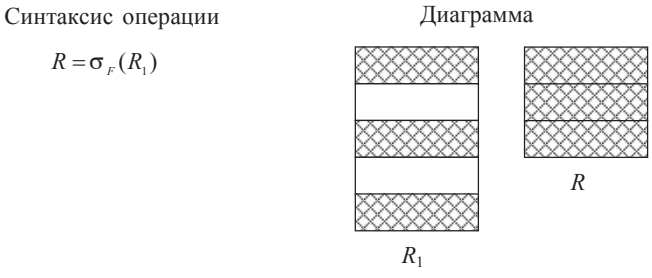

Рис. 9. Операция селекции

Операция селекция порождает результат как горизонтальную выборку, выборку строк-кортежей реляционной таблицы, удовлетворяющих заданному условию. Ввиду многообразия средств спецификации условий отбора операция селекции является одной из наиболее распространенных на практике операций над реляционными таблицами. Например, если из списка, заданного отношением-таблицей «Сотрудники» (R_1), нужно выбрать сотрудников отдела 101, результат можно получить при помощи операции селекции по условию «номер отдела = 101».

Сотрудники отдела 101 (*R*)

номер сотрудника	**фамилия**	**должность**	**зарплата**	**номер отдела**
1001	Иванов	инженер	15000	101
1003	Смирнов	инженер	15000	101
1004	Волков	инженер	15000	101

Соединение

Операция соединения отношений, наряду с операциями выборки и проекции, является одной из наиболее важных реляционных операций, так как лежит в основе большинства реальных запросов, адресуемых к данным нескольких связанных реляционных таблиц.

Обычно рассматривается несколько разновидностей операции соединения.

Наиболее общим является так называемое θ-соединение, которое формирует результат на основе сцепления кортежей отношений-операндов, чьи атрибуты связи находятся друг с другом в отношении θ ($=, \neq, >, <, \geq, \leq$).

Частным случаем θ-соединения является эквисоединение или соединение на основе условия равенства атрибутов связи.

Частным случаем эквисоединения является естественное, или внутреннее, соединение. Дадим неформальное определение этой важной операции.

Естественным (внутренним) соединением отношений R_1 (A, B, C) и R_2 (B, C, D) называется отношение R (A, B, C, D), полученное по условию равенства (эквисоединение) общих (часто одноименных) атрибутов связи (B, C), из которого исключаются дубли атрибутов.

Атрибуты связи (*B* и *C* в данном примере), принадлежащие разным отношениям-операндам, могут иметь разные имена, но должны быть определены на совместимых доменах.

Операция естественного соединения является зависимой и может быть выражена через другие операции. Синтаксис операции естественного соединения:

$$R = R_1 \triangleright\triangleleft R_2 = \pi_{A,B,C,D}\left(\sigma_{R_1.B=R_2.B \wedge R_1.C=R_2.C}\left(R_1 \times R_2\right)\right).$$

Результирующее отношение *R* состоит из кортежей, которые представляют собой сцепление только тех кортежей отношений-операндов R_1 и R_2, которые содержат совпадающие значения общих атрибутов (*B*, *C*), без дублей столбцов.

Допустим в кадровой информационной системе между отношениями «Сотрудники» (R_1) и «Дети» (R_2) имеется идентифицирующая связь типа «один ко многим». Атрибуты первичных ключей подчеркнуты. Семантика связи такова, что не у всех сотрудников могут быть дети. Причем у любого сотрудника может быть более одного ребенка. Тогда список сотрудников и их детей может быть получен при помощи естественного соединения.

Сотрудники» (R_1)

номер сотрудника	**фамилия**	**должность**	**номер отдела**
1001	Иванов	инженер	101
1002	Петров	инженер	102
1003	Смирнов	техник	101
1004	Волков	инженер	101
1005	Белов	техник	102
1006	Кравцов	оператор	102

Дети (R_2)

номер сотрудника	**имя ребенка**	**дата рождения**
1001	Иван	10.04.1998
1001	Татьяна	18.05.2004
1002	Сергей	22.01.1995
1004	Наталья	05.10.2005

Здесь атрибутами связи являются атрибуты «Сотрудники.номер сотрудника» и «Дети.номер сотрудника», а результатом является отношение «Сотрудники с детьми».

Сотрудники с детьми (*R*)

номер сотрудника	**фамилия**	**должность**	**номер отдела**	**имя ребенка**	**дата рождения**
1001	Иванов	инженер	101	Иван	10.04.1998
1001	Иванов	инженер	101	Татьяна	18.05.2004
1002	Петров	инженер	102	Сергей	22.01.1995
1004	Волков	инженер	101	Наталья	05.10.2005

В случае, когда для атрибутов, по которым осуществляется соединение, нет совпадающих значений атрибутов связи, между отношениями может быть выполнено внешнее соединение. *Внешнее соединение* предполагает формирование результирующего отношения, в том числе и из кортежей, которым нет соответствия в одном из отношений-операндов соединения. Различают левое и правое внешнее соединение.

Левым соединением называется соединение, в котором результирующее отношение состоит из кортежей, образованных либо сцеплением кортежей операндов, содержащих совпадающие значения общих атрибутов, либо кортежами левого операнда, не имеющими совпадающих значений общих атрибутов.

Правым соединением называется соединение, в котором результирующее отношение состоит из кортежей, образованных либо сцеплением кортежей операндов, содержащих совпадающие значения общих атрибутов, либо кортежами правого операнда, не имеющими совпадающих значений общих атрибутов.

Для обозначения несовпадающих значений кортежей используется определитель Null.

Результат операции левого соединения зависит от типа связи и роли отношений в качестве левого или правого операнда. В нашем примере между отношениями «Сотрудники» и «Дети» имеется *идентифицирующая* связь «один ко многим». По определению идентифицирующей связи в родительском отношении «Сотрудники» допускаются кортежи, которым нет соответствия в отношении-потомке «Дети». Если в качестве левого операнда используется родительское

отношение («Сотрудники»), то левое соединение даст полный список всех сотрудников и их отношение к детям.

Все сотрудники (R_1)

номер сотрудника	**фамилия**	**должность**	**номер отдела**	**имя ребенка**	**дата рождения**
1001	Иванов	инженер	101	Иван	10.04.1998
1001	Иванов	инженер	101	Татьяна	18.05.2004
1002	Петров	инженер	102	Сергей	22.01.1995
1003	Смирнов	техник	101	Null	Null
1004	Волков	инженер	101	Наталья	05.10.2005
1005	Белов	техник	102	Null	Null
1006	Кравцов	оператор	102	Null	Null

Результат операции правого соединения зависит от типа связи и роли отношений в качестве левого или правого операнда. Например, между отношениями «Сотрудники» и «Дети» имеется *идентифицирующая* связь «один ко многим». По определению идентифицирующей связи в отношении-потомке «Дети» не может быть кортежей, которым нет соответствия в родительском отношении «Сотрудники». Если в качестве правого операнда используется отношение-потомок («Дети»), то правое соединение даст результат, не отличающийся от результата естественного соединения.

Сотрудники с детьми (R)

номер сотрудника	**фамилия**	**должность**	**номер отдела**	**имя ребенка**	**дата рождения**
1001	Иванов	инженер	101	Иван	10.04.1998
1001	Иванов	инженер	101	Татьяна	18.05.2004
1002	Петров	инженер	102	Сергей	22.01.1995
1004	Волков	инженер	101	Наталья	05.10.2005

Результат операции правого соединения зависит от типа связи. Например, если между отношениями «Отдел» и «Сотрудники» имеется *неидентифицирующая* связь «один ко многим», то по определению неидентифицирующей связи в отношении-потомке «Сотрудники» допускаются кортежи, которым нет соответствия в родительском отношении «Отдел». Если в таком случае в качестве правого операн-

да используется отношение-потомок («Сотрудники»), то правое соединение даст полный список сотрудников, в том числе временно не приписанных ни к какому отделу (сотрудники, допустим, ликвидированного отдела 102).

Отдел (R_1)

номер отдела	**название отдела**
101	ОГК

Сотрудники (R_2)

номер сотрудника	**фамилия**	**должность**	**номер отдела**
1001	Иванов	инженер	101
1002	Петров	инженер	102
1003	Смирнов	техник	101
1004	Волков	инженер	101
1005	Белов	техник	102
1006	Кравцов	оператор	102

Все сотрудники (R)

номер отдела	**название отдела**	**номер сотрудника**	**фамилия**	**должность**
101	ОГК	1001	Иванов	инженер
Null	Null	1002	Петров	инженер
101	ОГК	1003	Смирнов	техник
101	ОГК	1004	Волков	инженер
Null	Null	1005	Белов	техник
Null	Null	1006	Кравцов	оператор

Деление

Делением отношений $R_1(A, B, C)$ и R_2 (C) называется отношение $R(A, B)$, тело которого содержит кортежи такие, что для *всех* кортежей отношения R_2 найдется кортеж в отношении R_1.

Операция деления является зависимой и может быть выражена через другие операции. Синтаксис операции деления:

$$R = R_1 \div R_2 = \pi_{A,B}(R_1) - \pi_{A,B}((R_2 \times \pi_{A,B}(R_1)) - R_1).$$

Результирующее отношение R состоит из кортежей отношения R_1, определенных на множестве атрибутов (A, B), которые соответствуют комбинации *всех* кортежей отношения R_2.

Деление отношений аналогично делению чисел с остатком.

Отношение R_1 выступает в роли *делимого*, отношение R_2 выступает в роли *делителя*.

Допустим отношение «Исполнитель проекта» (R_1) моделирует связь «многие ко многим» между отношениями «Сотрудники» и «Проекты» (R_2). Между родительским отношением «Проект» и отношением-потомком «Исполнитель проекта» имеется идентифицирующая зависимость. Тогда ответ на вопрос, какие сотрудники участвовали в работе над *всеми* проектами, даст отношение «Передовики» (R), полученное при помощи операции деления.

Исполнитель проекта (R_1)

номер сотрудника	**фамилия**	**номер проекта**
1001	Иванов	1
1001	Иванов	2
1001	Иванов	3
1002	Петров	1
1002	Петров	3
1003	Смирнов	1
1003	Смирнов	2
1003	Смирнов	3
1004	Волков	2

Проект (R_2)

номер проекта
1
2
3

Передовики (R)

номер сотрудника	**фамилия**
1001	Иванов
1003	Смирнов

SQL — ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ

Введение в SQL

Различают два вида языков запросов для реляционной модели данных.

Алгебраические языки позволяют выразить запросы средствами специальных операторов над отношениями — операторов реляционной алгебры.

Языки реляционного исчисления позволяют описать запросы на языке исчисления предикатов — языке математической логики.

Алгебраические языки являются процедурными языками. Языки реляционного исчисления относятся к классу декларативных языков, т. е. языков более высокого уровня. Эти языки являются абстрактными языками и в чистом виде не реализованы, но они служат эталоном для оценки реальных языков запросов. По своим выразительным возможностям эти языки эквивалентны. Они были предложены в свое время автором реляционной модели Коддом для представления минимальных возможностей любого реального языка запросов реляционной модели. Восемь алгебраических операторов: объединение, пересечение, вычитание, декартово произведение, проекция, селекция, соединение и деление, — введенные Коддом, являются мерой оценки выразительной силы любого языка баз данных. Реальный язык баз данных обладает свойством *реляционной полноты*, если по своим выразительным возможностям он не уступает реляционной алгебре или реляционному исчислению.

Реальные языки запросов обычно обеспечивают не только реляционную полноту, но и другие возможности, выходящие за пределы

реляционной алгебры или реляционного исчисления. К числу таких возможностей обычно относятся следующие операции:

- добавление данных;
- модификация данных;
- удаление данных;
- арифметические вычисления и сравнения;
- присваивание и отображение;
- вычисление функций агрегации, выполняемые над однородными элементами данных (вычисление количества, суммы, среднего, минимального или максимального значения и других).

Наиболее распространенным языком запросов реляционной модели данных в настоящее время является язык структурированных запросов SQL (Structured Query Language).

SQL нельзя отнести к конкретному виду языков. Он содержит в себе возможности и языка реляционного исчисления (исчисления кортежей), и алгебраического языка и несомненно является реляционно полным.

В силу исторических причин SQL стал стандартным реляционным языком и в настоящее время поддерживается практически всеми системами баз данных. Поэтому каждый специалист по базам данных должен быть знаком с ним.

В чистом виде как язык, поддерживающий реляционную модель, наиболее популярен стандарт языка, известный как SQL/92 (International Standard Database Language SQL).

Ни один из коммерческих продуктов не поддерживает в полной мере стандарт SQL/92. Известные версии этого языка, по образному выражению Дейта [2], можно назвать «надмножествами подмножеств» языка SQL/92. Другими словами, любая коммерческая система, не поддерживая некоторые аспекты стандарта, в других отношениях, возможно, превосходит его. Это замечание справедливо и в отношении версии SQL Microsoft Jet, реализованной в популярной СУБД MS Access.

Язык SQL является языком декларативного типа. В нем отсутствуют какие-либо команды управления ходом вычислительного процесса типа IF-THEN-ELSE, SWITCH, WHILE, DO-WHILE, FOR, GO TO и др. Управление ходом выполнения процесса обработки данных может выполняться интерактивно, в результате действий самого пользователя или при помощи процедурных языков программирования высокого уровня. В связи с этим различают две разновидности SQL — ин-

терактивный и вложенный. По большей части обе формы работают одинаково, но используются различно.

Интерактивный SQL используется для функционирования непосредственно в базе данных и реализуется непосредственно при вводе пользователем отдельных команд. При такой форме SQL при вводе команды она сейчас же выполнится и вы сможете увидеть результат немедленно.

Вложенный SQL состоит из команд SQL внедренных внутрь программ, написанных на процедурных языках программирования высокого уровня, что делает программы более мощными и эффективными. Однако при таком способе управления процессом обработки данных приходится иметь дело со структурой SQL и стилем управления, который требует некоторых расширений к интерактивному SQL. На практике существуют два основных способа использования SQL в программах:

- внедрение SQL-операторов в исходный текст программы с последующей ее компиляцией и компоновкой;
- использование интерфейса прикладного программирования, специализирующегося на работе с базами данных.

В учебном пособии, рассчитанном на неподготовленного пользователя SQL, ограничимся интерактивной формой языка. Это даст возможность обсуждать команды и результаты их выполнения, не заботясь о том, как они связаны с помощью интерфейса с другими языками. Интерактивный SQL — это наиболее полезная для непрограммистов форма. Все, что вы узнаете относительно интерактивного SQL, в основном применимо и к вложенной форме.

В соответствии со стандартом SQL выразительные возможности языка распределены по шести типам подмножеств команд:

- *язык определения данных (Data Definition Language — DDL)*, представлен инструкциями CREATE, ALTER, DROP;
- *язык манипулирования данными (Data Manipulation Language — DML)*, представлен инструкциями INSERT, UPDATE, DELETE;
- *язык запросов данных (Data Query Language — DQL)*, представлен многофункциональной командой SELECT;
- *язык управления данными (Data Control Language — DCL)*, представлен командами GRANT (предоставление привилегий) и REVOKE (отмена привилегий);
- *язык обработки транзакций (Transaction Processing Language — TPL)*, включает команды BEGIN (начать транзакцию), COMMIT (завершить транзакцию), ROLLBACK (откатить транзакцию);

- *язык управления курсором (Cursor Control Language — CCL)* предназначен для выполнения операций с отдельными строками одной или нескольких таблиц, представлен командами DECLARE CURSOR, OPEN, CLOSE, FETCH INTO, DROP CURSOR.

К сожалению, эти подмножества не используются повсеместно во всех реализациях. Они подчеркиваются ANSI и полезны на концептуальном уровне. В частности, интерактивный Microsoft Jet SQL, реализованный в популярной СУБД MS Access, поддерживает только первые три подмножества.

Определение данных в SQL

Рассмотрим терминологию SQL. В языке SQL поддерживаются такие термины, как таблица, столбец, строка, вместо реляционных терминов — отношение, атрибуты, кортежи.

Связь терминов реляционной модели данных, SQL и Access представлена в табл. 2.

Таблица 2

Соответствие терминов

Реляционная модель	SQL	Access
Отношение	Таблица (Table)	Таблица
Атрибут	Столбец (Column)	Поле
Кортеж	Строка (Row)	Запись

База данных в SQL представляет собой набор взаимосвязанных таблиц. При этом различают базовые таблицы и таблицы-представления. **Базовая таблица** (TABLE) — это основной структурный элемент базы данных. Ей соответствуют реальные хранимые данные. Концептуальная модель базы данных представляет собой совокупность взаимосвязанных базовых таблиц. *Представление* (VIEW) является виртуальной таблицей, которая выглядит как реально существующая таблица. Представление не содержит собственных данных, в нем скомбинированы данные из одной или нескольких связанных базовых таблиц. Средством наполнения таблицы-представления являются запросы, в которых реализуются информационные потребности пользователей базы данных. С помощью представления реа-

лизуются локальные представления (внешняя модель данных) базы данных, используемые в прикладных программах или запросах пользователей. Сразу необходимо отметить, что концепция представлений в версии SQL Microsoft Jet по умолчанию не поддерживается.

Средством эффективного доступа к хранимым данным являются *индексы* (INDEX). При помощи индексов осуществляется доступ к данным, упорядоченным по определенным критериям. В качестве критерия упорядочения могут использоваться один или несколько столбцов таблицы. Наглядным образом индекса является таблица, первый столбец которой содержит значения индексируемых полей, а второй — ссылки на соответствующие строки. При создании базовых таблиц автоматически создаются индексы по первичному ключу, по альтернативным ключам и по внешним ключам.

Действующим на данный момент стандартом языка SQL является принятая Американским национальным институтом стандартов (American National Standards Institute — ANSI) версия SQL-92. Фирмы-разработчики СУБД при реализации языка SQL могут вносить в него расширения, но обязаны реализовать базовый набор команд ANSI SQL.

Процессор обработки данных Microsoft Jet является составной частью Access и выполняет инструкции Access SQL (SQL Microsoft Jet), который отличается от ANSI SQL существенно [12] (как правило, настольные СУБД, совместимые со стандартом SQL, реализуют не все инструкции ANSI SQL).

Язык SQL Microsoft Jet почти соответствует стандарту ANSI SQL-89. В реализацию языка SQL для Microsoft Jet *4.x* (начиная с версии Microsoft Access 2000) внесены несколько расширений, которые приближают его к стандарту ANSI SQL-92 и Transact-SQL — диалекту языка SQL для Microsoft SQL Server. Для того чтобы обеспечить совместимость с предыдущими версиями Microsoft Jet, эти расширения можно использовать только в специальном режиме — ANSI SQL-92.

Основные различия языков SQL Microsoft Jet и ANSI SQL состоят в следующем [12]:

- они имеют разные наборы зарезервированных слов и типов данных;
- разные правила применимы к оператору Between… And, используемому для определения условий выборки записей;
- подстановочные знаки ANSI и Microsoft Jet, которые используются в операторе Like, взаимно исключают друг друга;

- язык Jet SQL обычно предоставляет пользователю большую свободу, например разрешается группировка и сортировка по выражениям;
- язык Jet SQL позволяет использовать более сложные выражения.

Типы данных языка SQL ядра базы данных Microsoft Jet включают следующие основные типы данных [12] (табл. 3).

Таблица 3

Типы данных SQL Microsoft Jet

Тип данных	Размер	Описание
BINARY	1 байт на символ	В таком поле могут быть сохранены данные любого типа. Преобразование данных (например, в текст) не производится. От способа ввода данных в бинарное поле зависит способ вывода выходных данных
BIT	1 байт	Значения «Да» и «Нет», а также поля, содержащие только одно из двух возможных значений
TINYINT	1 байт	Целое число от 0 до 255
MONEY	8 байт	Масштабируемое целое число в диапазоне от –922 337 203 685 477,5808 до 922 337 203 685 477,5807
DATETIME (см. DOUBLE)	8 байт	Даты и время, относящиеся к годам с 100 по 9999
UNIQUEIDENTIFIER	128 бит	Уникальный идентификационный номер, используемый для удаленного вызова процедур
REAL	4 байт	Значения обычной точности с плавающей запятой в диапазоне от –3,402823E38 до –1,401298E-45 для отрицательных значений, от 1,401298E-45 до 3,402823E38 для положительных значений и 0
FLOAT	8 байт	Значения двойной точности с плавающей запятой в диапазоне от –1,79769313486232E308 до –4,94065645841247E-324 для отрицательных значений и от 4,94065645841247E-324 до 1,79769313486232E308 для положительных значений и 0
SMALLINT	2 байт	Короткое целое число в диапазоне от –32 768 до 32 767
INTEGER	4 байт	Длинное целое число в диапазоне от –2 147 483 648 до 2 147 483 647

Окончание табл. 3

Тип данных	Размер	Описание
DECIMAL	17 байт	Точный числовой тип данных, включающий значения от 10 ^ 28–1 до –10 ^ 28–1. Можно определить как точность (1–28), так и масштаб (точность определена как 0). Точность и масштаб по умолчанию составляют 18 и 0 соответственно
TEXT	2 байт на символ	От 0 до 2,14 Гбайт
IMAGE	По требованию	От 0 до 2,14 Гбайт. Используется для объектов OLE
CHARACTER	2 байт на символ	От 0 до 255 символов

Типы данных ANSI SQL, эквивалентные им типы данных языка SQL Microsoft Jet и допустимые синонимы приведены ниже [12] (табл. 4).

Таблица 4

Типы данных стандартного SQL и SQL Microsoft Jet

Тип данных ANSI SQL	Тип данных SQL Microsoft Access	Синоним	Тип данных Microsoft SQL Server
BIT, BIT VARYING	BINARY	VARBINARY, BINARY VARYING BIT VARYING	BINARY, VARBINARY
Не поддерживается	BIT	BOOLEAN, LOGICAL, LOGICAL1, YESNO	BIT
Не поддерживается	TINYINT	INTEGER1, BYTE	TINYINT
Не поддерживается	COUNTER	AUTOINCREMENT	—
Не поддерживается	MONEY	CURRENCY	MONEY
DATE, TIME, TIMESTAMP	DATETIME	DATE, TIME	DATETIME
Не поддерживается	UNIQUEIDENTIFIER	GUID	UNIQUEIDENTIFIER
DECIMAL	DECIMAL	NUMERIC, DEC	DECIMAL
REAL	REAL	SINGLE, FLOAT4, IEEESINGLE	REAL
DOUBLE PRECISION, FLOAT	FLOAT	DOUBLE, FLOAT8, IEEEDOUBLE, NUMBER	FLOAT

Окончание табл. 4

Тип данных ANSI SQL	Тип данных SQL Microsoft Access	Синоним	Тип данных Microsoft SQL Server
SMALLINT	SMALLINT	SHORT, INTEGER2	SMALLINT
INTEGER	INTEGER	LONG, INT, INTEGER4	INTEGER
INTERVAL	Не поддерживается		Не поддерживается
Не поддерживается	IMAGE	LONGBINARY, GENERAL, OLEOBJECT	IMAGE
Не поддерживается	TEXT	LONGTEXT, LONGCHAR, MEMO, NOTE, NTEXT	TEXT
CHARACTER, CHARACTER VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING	CHAR	TEXT (n), ALPHANUMERIC, CHARACTER, STRING, VARCHAR, CHARACTER VARYING, NCHAR, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING	CHAR, VARCHAR, NCHAR, NVARCHAR

Ядро базы данных Microsoft Jet SQL в основном совместимо с ANSI-89 Level 1. Однако некоторые возможности языка запросов ANSI SQL не реализованы в Microsoft Jet SQL. Режим ANSI-89 является настройкой по умолчанию для новой базы данных Microsoft Access в формате файла 2002–2003, 2007, 2010.

В ANSI-92 имеются новые зарезервированные слова, правила синтаксиса и подстановочные знаки, расширяющие возможности создания запросов и использования команд SQL. Этот режим близок к спецификации ANSI-92 уровня 1, но не является совместимым с ANSI-92 уровня 1. В данном режиме запроса содержится больше синтаксиса ANSI, а подстановочные знаки удовлетворяют спецификации SQL.

Использование ANSI-92 SQL может потребоваться, если предполагается будущее развитие приложения до проекта Microsoft Access и требуется разработать запросы, которые будут выполняться с минимальными изменениями в базе данных Microsoft SQL Server.

Для настройки режима совместимости с более совершенной версией стандарта ANSI-92 SQL для текущей базы данных рекомендуется выполнить следующие действия:

1. «Запустить» MS Access.
2. Создать пустую базу данных.
3. В левом верхнем углу окна нажать кнопку <Office>.
4. В диалоговом окне нажать кнопку <Параметры Access>.
5. В диалоговом окне «Параметры Access» в списке слева выбрать «Конструкторы объектов».
6. В разделе «Конструктор запросов» в опции «Синтаксис для SQL Server (ANSI 92)» установить флажок «*эта база данных*».
7. Нажать кнопку <OK>.

Описание учебного проекта

При знакомстве с языком SQL в качестве примера предметной области рассмотрим в упрощенном виде организацию учебного процесса в вузе. Прежде всего, следует напомнить, что реализации проекта базы данных предшествует большая аналитическая работа, которая выполняется на этапах инфологического и даталогического проектирования. На основании анализа семантики данных предметной области разрабатывается ее информационная модель, которая затем отображается в компьютерную среду с учетом структурных и процедурных возможностей СУБД реляционного типа. Разработка структуры базы данных, в том числе разработка структуры каждой таблицы, определение свойств атрибутов, установление зависимостей между ними, назначение ключевых атрибутов, установление связей между таблицами, является результатом продолжительной аналитической работы [13]. Генерация схемы базы данных, независимо от того выполняется ли она с помощью SQL или используются другие инструментальные возможности СУБД, возможна лишь тогда, когда проработаны особенности структуры данных предметной области.

Допустим, что нашу предметную область можно описать в терминах понятий (сущностей): «факультет», «кафедра», «сотрудник» (в том числе «преподаватель», «инженер», «заведующий кафедрой»), «специальность» (направление), «студент», «учебная дисциплина», «экзамен».

Пусть в нашей учебной предметной области действуют следующие семантические условия или бизнес-правила.

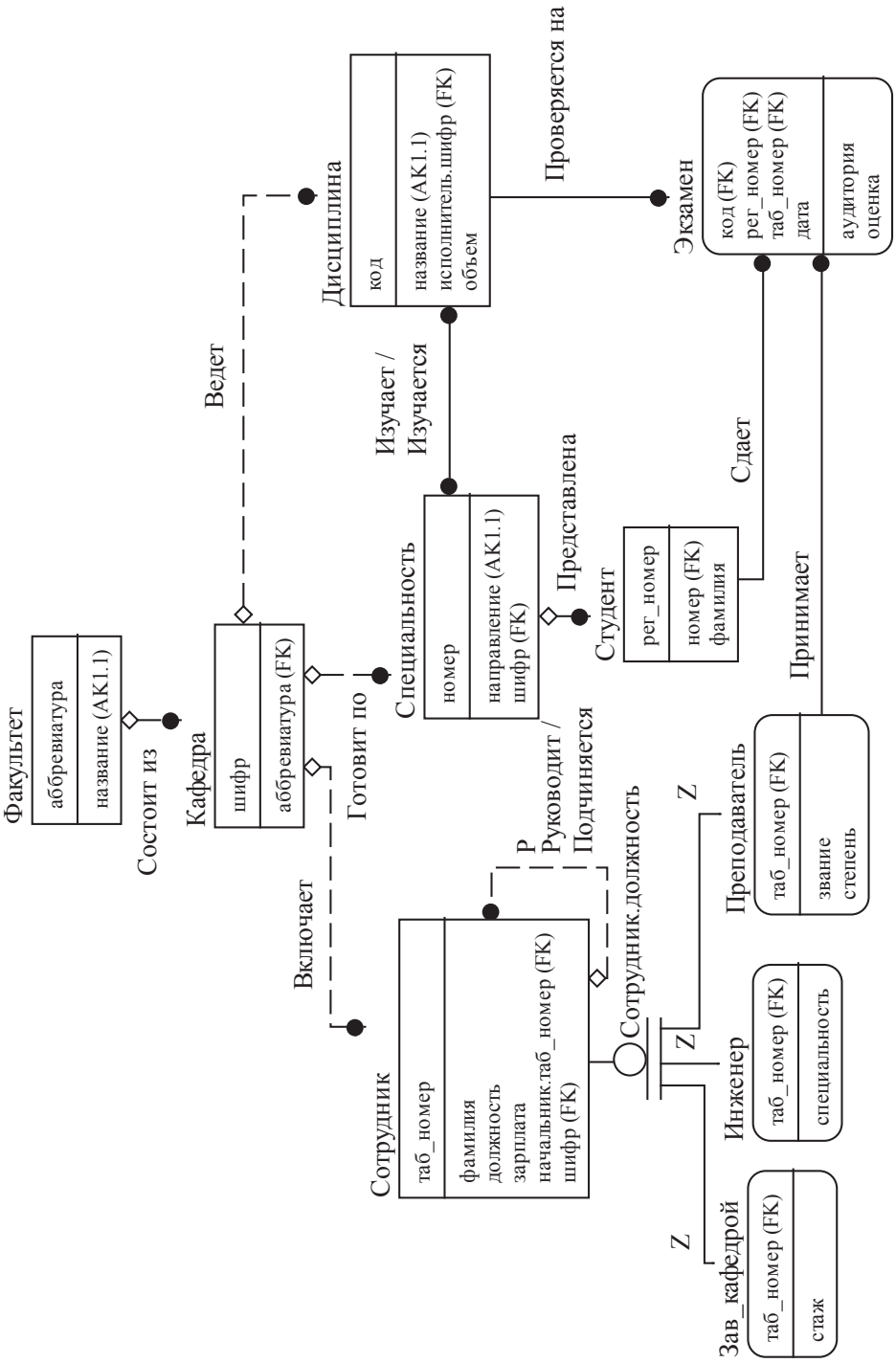

Рис. 10. Модель «сущность-связь» предметной области «Учебный процесс»

Вуз состоит из факультетов (рис. 10). Каждый факультет имеет уникальный идентификатор-аббревиатуру и название. Работа на факультете организована по кафедрам. Каждая кафедра также имеет уникальный шифр и название и может быть приписана только к одному факультету (неидентифицирующая связь «Состоит из»).

Коллектив кафедры образуют сотрудники. Уникальным идентификатором сотрудника в пределах вуза является его табельный номер. Сотрудники могут работать только на одной кафедре (неидентифицирующая связь «Включает») и выполнять обязанности в роли преподавателей, инженеров или заведующих кафедрами (категориальная связь).

Каждый сотрудник кафедры имеет одного непосредственного начальника из числа сотрудников кафедры (рекурсивная унарная связь «Руководит/Подчиняется»).

Кафедра может готовить специалистов по нескольким специальностям (направлениям), причем по каждой специальности осуществляется подготовка только на одной кафедре (неидентифицирующая связь «Готовит по»). Каждая специальность имеет уникальный номер.

По каждой специальности обучаются несколько студентов. Каждый студент данной кафедры может обучаться только по одной специальности (неидентифицирующая связь «Представлена»). Уникальной характеристикой персоны студента в пределах вуза является его регистрационный номер.

По любой специальности предполагается изучение многих дисциплин. Любая дисциплина имеет уникальный код и название. Исполнителем учебной дисциплины может быть только одна кафедра (неидентифицирующая связь «Ведет»). Причем любую дисциплину кафедра-исполнитель может вести на многих специальностях (связь «многие-ко-многим» «Изучает/Изучается»).

По результатам обучения студенты сдают экзамены по определенной дисциплине конкретному преподавателю в конкретное время и получают оценки. При этом каждый студент сдает экзамены по нескольким дисциплинам, любой преподаватель может принимать экзамены по нескольким дисциплинам и, наконец, по любой дисциплине могут принимать экзамены разные преподаватели у разных студентов (тернарная связь «Сдает–Принимает–Проверяется на»).

Структура данных предметной области в виде информационных элементов (сущностей) и связей между ними может быть систематизирована в образе ее инфологической (информационной) модели (рис. 10).

Обычно для наименования объектов базы данных SQL по традиции базы данных используется латиница. Имея в виду учебный характер пособия, воспользуемся возможностью SQL Microsoft Jet и будем именовать объекты базы данных на русском языке. Ключевые слова SQL можно набирать строчными или прописными буквами. В демонстрационных примерах для наглядности ключевые слова будем набирать прописными буквами.

Для работы в режиме интерактивного SQL в MS Access необходимо выполнить следующие действия[1]:

1. «Запустить» MS Access.
2. Настроиться на режим совместимости с ANSI-92 SQL.
3. Создать файл базы данных (или открыть уже созданную базу данных).
4. Перейти в режим «Конструктора запросов».
5. Закрыть диалоговое окно «Добавление таблицы».
6. Перейти в интерактивный режим SQL.
7. Набрать текст SQL-команды.
8. Запустить команду (запрос) на исполнение и в случае корректности результата сохранить запрос под оригинальным именем для отчета.

Для закрепления навыков конструирования SQL-инструкций предлагается:

- познакомиться с форматом SQL-инструкции;
- разобраться с демонстрационными примерами;
- выполнить предлагаемые задания.

При работе с печатным вариантом учебного пособия демонстрационные примеры являются всего лишь иллюстрациями принципов построения SQL-инструкций. При работе с электронным вариантом учебного пособия текст SQL-инструкции примера можно перенести через буфер обмена в текстовый редактор SQL-режима «Конструктора запросов» и выполнить запрос.

[1] Предполагается, что пользователь владеет навыками работы в среде MS Access.

Язык определения данных

Создание таблиц

Для описания новой таблицы, ее полей и индексов используется инструкция CREATE TABLE, которая имеет следующий обобщенный формат.

CREATE TABLE <имя таблицы>
(<список элементов базовой таблицы>);

Здесь и далее в отличие от круглых скобок, которые являются элементами синтаксиса языка, угловые скобки < > обозначают метасимволы, квадратные скобки [] обозначают необязательные элементы, фигурные скобки {} обозначают альтернативу. Элементы списка перечисляются через запятую.

Элементами списка могут быть:

- определение столбца;
- определение ограничения целостности.

Определение столбца имеет следующий формат:

<имя столбца> <тип> [(<размер>)] [NOT NULL]

Если для поля установлено свойство NOT NULL, то поле обязательно должно содержать допустимые данные. Например:

шифр Text (5) NOT NULL

Определение ограничения целостности имеет такой формат:

CONSTRAINT <имя индекса> <ограничение>

Ограничения целостности бывают следующих типов:

- первичный ключ;
- альтернативный (уникальный) ключ;
- внешний ключ.

Для каждого типа ключа создается индекс.

Первичный ключ может быть составным. Определение первичного ключа — не более одного на таблицу:

PRIMARY KEY (<список имен столбцов>)

Здесь **<список имен столбцов>** — имена одного или нескольких столбцов, которые следует назначить ключевыми. По определению они не могут принимать значение NULL.

Альтернативные ключи могут быть составными. Альтернативных ключей может быть несколько. Определение альтернативного ключа:

UNIQUE (<список имен столбцов>)

Здесь **<список имен столбцов>** — имена одного или нескольких столбцов, которые следует включить в альтернативный ключ (уникальный индекс).

Внешние ключи могут быть составными. Внешних ключей может быть несколько. Определение внешнего ключа:

FOREIGN KEY (<список имен столбцов внешнего ключа>)
REFERENCES <внешняя таблица> (<список имен столбцов первичного ключа внешней таблицы>)
[ON DELETE <опция>]
[ON UPDATE <опция>]

Здесь **<список имен столбцов>** — имена одного или нескольких столбцов, включенных во внешний ключ, которые содержат ссылки на столбцы потенциального (часто первичного) ключа другой внешней таблицы.

<опция>={CASCADE|SET NULL} — реакция на нарушение ограничения ссылочной целостности при выполнении корректирующих операций. Умолчанием является RESTRICT — запрет на выполнение операций, ведущих к нарушению ссылочной целостности. Альтернативы: CASCADE — изменение «по каскаду» значений внешних ключей во всех кортежах таблиц-потомков, ссылающихся на обновляемый кортеж родительской таблицы, SET NULL — присваивание внешнему ключу таблицы-потомка значения «не определено» (NULL).

Допустим, что на этапах, предшествующих использованию SQL, все характеристики реляционных таблиц (имена таблиц, имена столбцов, их свойства, зависимости между ними, первичные, альтернативные, внешние ключи) уже определены. Спецификации структуры реляционной базы данных (даталогическая модель) приведены в табл. 5–15[2].

Все таблицы базы данных должны быть созданы и заполнены с учетом порядка их старшинства. Первой создается таблица «Факультет».

[2] В спецификации некоторых таблиц намеренно внесены ошибки, для того чтобы впоследствии продемонстрировать возможности модификации структуры таблиц средствами языка SQL.

Таблица 5

Структура таблицы «Факультет»

Номер	Имя	Домен (тип)	NULL	Примечание
1	аббревиатура	TEXT (3)	Запрещено	Первичный ключ
2	название	TEXT (50)	Запрещено	Альтернативный ключ

Пример 1. Создать таблицу «Факультет».

```
CREATE TABLE Факультет
(аббревиатура TEXT (3) NOT NULL,
название TEXT (50) NOT NULL,
CONSTRAINT ПК_Факультет PRIMARY KEY (аббревиатура),
CONSTRAINT АК_Факультет UNIQUE (название));
```

Таблица 6

Структура таблицы «Кафедра»

Номер	Имя	Домен (тип)	NULL	Примечание
1	шифр	TEXT (20)	Запрещено	Первичный ключ
2	название	TEXT (50)	–	Альтернативный ключ
3	фак	TEXT (2)	–	Внешний ключ

Пример 2. Создать таблицу «Кафедра».

```
CREATE TABLE Кафедра
(шифр TEXT (8) NOT NULL, название TEXT (50) NOT NULL, фак TEXT (4),
CONSTRAINT ПК_Кафедра PRIMARY KEY (шифр),
CONSTRAINT АК_Кафедра UNIQUE (название),
CONSTRAINT Кафедра_Факультет FOREIGN KEY (фак)
REFERENCES Факультет (аббревиатура)
ON UPDATE CASCADE
ON DELETE SET NULL);
```

В этом примере выбрана реакция ON UPDATE CASCADE на обновление первичного ключа в родительской таблице «Факультет», что представляется естественным для нашей предметной области. Изменение аббревиатуры факультета можно распространить на дочернюю таблицу «Кафедра». Стратегия ON DELETE CASCADE вряд ли будет уместна: удаление строки в родительской таблице «Факультет» не должно повести за собой серию удалений в таблице-потомке «Кафедра». Стратегия ON DELETE SET NULL выбрана потому, что ликвидация факультета (удаление строки в таблице «Факультет») не долж-

на сопровождаться ликвидацией его кафедр. Как возможный вариант реакции на это событие — перевод соответствующих кафедр в неопределенное промежуточное состояние, в котором внешний ключ «факультет» таблицы «Кафедра» принимает значение NULL.

Анализ семантики данных, подобный тому, что был использован в спецификации ограничений ссылочной целостности таблицы «Кафедра», может быть формализован и в случае других таблиц.

Другие таблицы могут быть созданы по образцу. При создании таблиц следует учитывать порядок их старшинства. В первую очередь создаются родительские таблицы, затем таблицы-потомки, содержащие ссылки на них. В нашем случае порядок создания таблиц должен быть такой:

- «Факультет»;
- «Кафедра»;
- «Сотрудник»;
- «Специальность»;
- «Дисциплина»;
- «Заявка»;
- «Зав_кафедрой»;
- «Инженер»;
- «Преподаватель»;
- «Студент»;
- «Экзамен».

Таблица 7

Структура таблицы «Сотрудник»

Номер	Имя	Домен (тип)	NULL	Примечание
1	таб_номер	TEXT (3)	Запрещено	Первичный ключ
2	шифр	TEXT (8)	–	Внешний ключ
3	фамилия	TEXT (20)	–	–
4	должность	TEXT (20)	–	–
5	зарплата	MONEY	–	–
6	шеф	TEXT (3)	Запрещено	Внешний ключ

Пример 3. Создать таблицу «Сотрудник».

```
CREATE TABLE Сотрудник
(таб_номер TEXT (3) NOT NULL,
шифр TEXT (8),
фамилия TEXT (20),
```

должность TEXT (20),
зарплата MONEY,
шеф TEXT (3),
CONSTRAINT ПК_Сотрудник PRIMARY KEY (таб_номер),
CONSTRAINT Сотрудник_Сотрудник FOREIGN KEY (шеф)
REFERENCES Сотрудник (таб_номер)
ON UPDATE CASCADE
ON DELETE SET NULL,
CONSTRAINT Сотрудник_Кафедра FOREIGN KEY (шифр)
REFERENCES Кафедра (шифр)
ON UPDATE CASCADE
ON DELETE SET NULL);

З а д а н и е . Создайте остальные таблицы базы данных в указанной последовательности по образцу. Заданная последовательность обеспечит корректность ссылок между таблицами.

Таблица 8

Структура таблицы «Специальность»

Номер	Имя	Домен (тип)	NULL	Примечание
1	номер	TEXT (8)	Запрещено	Первичный ключ
2	направление	TEXT (50)	Запрещено	Альтернативный ключ
3	шифр	TEXT (20)	–	Внешний ключ

П р и м е р 4 . Создать таблицу «Специальность».
CREATE TABLE Специальность
(номер TEXT (8) NOT NULL,
направление TEXT (50) NOT NULL,
шифр TEXT (8),
CONSTRAINT ПК_Специальность PRIMARY KEY (номер),
CONSTRAINT АК_Специальность UNIQUE (направление),
CONSTRAINT Специальность_Кафедра FOREIGN KEY (шифр) REFERENCES Кафедра (шифр)
ON UPDATE CASCADE
ON DELETE SET NULL);

Таблица 9

Структура таблицы «Дисциплина»

Номер	Имя	Домен	NULL	Примечание
1	код	TEXT (6)	Запрещено	Первичный ключ
2	объем	SMALLINT	–	–

Пример 5. Создать таблицу «Дисциплина».

```
CREATE TABLE Дисциплина
(код TEXT (6) NOT NULL,
объем SMALLINT,
CONSTRAINT ПК_Дисциплина PRIMARY KEY (код));
```

Пример 6. Создать таблицу «Заявка». Эта таблица является ассоциативной и моделирует связь типа «многие-ко-многим» между таблицами «Специальность» и «Дисциплина». Действительно, с одной стороны, на любой специальности предполагается изучение многих дисциплин. С другой стороны, любая дисциплина может быть поставлена для многих специальностей. По общему правилу построения таких таблиц первичный ключ образуется из внешних ключей связи ассоциативной таблицы со связуемыми таблицами. Таким образом, между ассоциативной таблицей и связуемыми таблицами устанавливаются идентифицирующие связи со всеми их особенностями в отношении контроля ссылочной целостности.

Таблица 10

Структура таблицы «Заявка»

Номер	Имя	Домен (тип)	NULL	Примечание
1	номер	TEXT (8)	Запрещено	Первичный ключ Внешний ключ
2	код	TEXT (6)	Запрещено	Первичный ключ Внешний ключ

```
CREATE TABLE Заявка
(номер TEXT (8) NOT NULL,
код TEXT (6) NOT NULL,
CONSTRAINT ПК_Заявка PRIMARY KEY (номер, код),
CONSTRAINT Заявка_Дисциплина FOREIGN KEY (код)
REFERENCES Дисциплина (код),
```

CONSTRAINT Заявка_Специальность FOREIGN KEY (номер)
REFERENCES Специальность (номер));

В таблице «Заявка» по определению идентифицирующей связи действует правило запрета на обновление и удаление строк в родительских таблицах. Оно подразумевается по умолчанию, и поэтому явно ограничения ссылочной целостности задавать не нужно.

Таблица 11

Структура таблицы «Зав_кафедрой»

Номер	Имя	Домен (тип)	NULL	Примечание
1	таб_номер	TEXT (3)	Запрещено	Первичный ключ Внешний ключ
2	стаж	SMALLINT	–	–

Пример 7. Создать таблицу «Зав_кафедрой».

```
CREATE TABLE Зав_кафедрой
(таб_номер TEXT (3) NOT NULL,
стаж SMALLINT,
CONSTRAINT ПК_Зав_кафедрой PRIMARY KEY (таб_номер),
CONSTRAINT Зав_кафедрой_Сотрудник FOREIGN KEY (таб_номер) REFERENCES Сотрудник (таб_номер)
ON UPDATE CASCADE
ON DELETE CASCADE);
```

В этом примере и некоторых последующих на удаление строки в таблице «Сотрудник» выбрана стратегия CASCADE, потому что по определению категориальной связи строка в родительской таблице и в таблице-потомке представляют один и тот же экземпляр объекта.

Таблица 12

Структура таблицы «Инженер»

Номер	Имя	Домен (тип)	NULL	Примечание
1	таб_номер	TEXT (3)	Запрещено	Первичный ключ Внешний ключ
2	специальность	TEXT (20)	–	–

Пример 8. Создать таблицу «Инженер».

```
CREATE TABLE Инженер
(таб_номер TEXT (3) NOT NULL,
специальность TEXT (20),
CONSTRAINT ПК_Инженер PRIMARY KEY (таб_номер),
CONSTRAINT Инженер_Сотрудник FOREIGN KEY (таб_номер) REFERENCES Сотрудник (таб_номер)
ON UPDATE CASCADE
ON DELETE CASCADE);
```

Задание. Таблицы базы данных «Преподаватель», «Студент», «Экзамен» создайте самостоятельно.

Таблица 13

Структура таблицы «Преподаватель»

Номер	Имя	Домен (тип)	NULL	Примечание
1	таб_номер	TEXT (3)	Запрещено	Первичный ключ Внешний ключ
2	звание	TEXT (10)	–	–
3	степень	TEXT (10)	–	–

Таблица 14

Структура таблицы «Студент»

Номер	Имя	Домен (тип)	NULL	Примечание
1	рег_номер	TEXT (6)	Запрещено	Первичный ключ
2	номер	TEXT (8)	–	Внешний ключ
3	фамилия	TEXT (20)	–	–

Таблица 15

Структура таблицы «Экзамен»

Номер	Имя	Домен (тип)	NULL	Примечание
1	код	TEXT (6)	Запрещено	Первичный ключ Внешний ключ
2	рег_номер	TEXT (6)	Запрещено	Первичный ключ Внешний ключ
3	таб_номер	TEXT (6)	Запрещено	Первичный ключ Внешний ключ
4	дата	DATE	Запрещено	Первичный ключ
5	аудитория	TEXT (3)	–	–
6	оценкаl	REAL	–	–

В таблице «Экзамен» по определению идентифицирующей связи действует правило запрета на обновление и удаление строк в родительских таблицах. Оно подразумевается по умолчанию, и поэтому явно ограничения ссылочной целостности задавать не нужно.

В результате созданная в MS Access схема базы данных будет иметь вид, представленный на рис. 11.

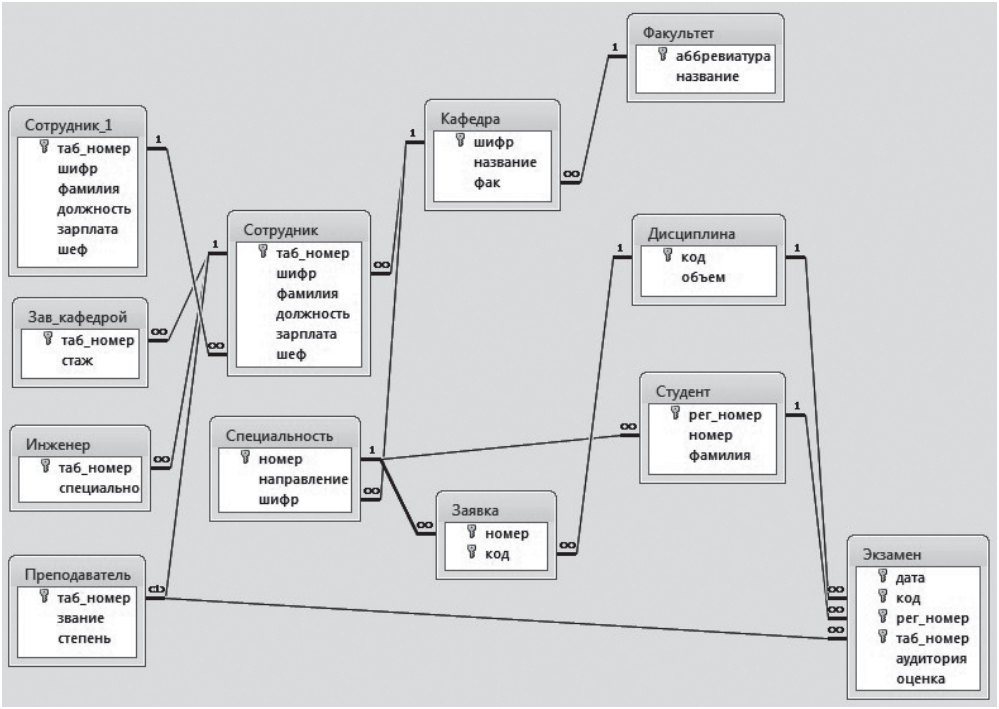

Рис. 11. Исходная схема базы данных

Модификация структуры таблицы

После того как таблица создана, можно в любое время (даже если таблица заполнена) изменить ее структуру. Для модификации структуры таблицы предназначена инструкция ALTER TABLE, которая имеет следующий формат:

```
ALTER TABLE <имя таблицы>
{ADD {COLUMN<имя столбца> <тип> [(<размер>)] [NOT NULL]
|CONSTRAINT <имя индекса> <ограничение>}
|DROP {COLUMN <имя столбца> |CONSTRAINT <имя индекса>}};
```

Инструкция ALTER TABLE дает возможность добавить (ADD) или удалить (DROP) один столбец (COLUMN) или одно ограничение целостности (CONSTRAINT) в виде индекса. Метасимволы имеют тот же смысл, что и в команде CREATE TABLE. Метасимвол <ограничение> может быть представлен спецификацией или первичного ключа, или альтернативного ключа, или внешнего ключа. Для репетиции выполните нижеследующие примеры.

Пример 9. Переименовать поле «фак» таблицы «Кафедра» на «факультет».

```
ALTER TABLE Кафедра ADD COLUMN факультет TEXT (4);
ALTER TABLE Кафедра DROP CONSTRAINT Кафедра_Факультет;
ALTER TABLE Кафедра DROP COLUMN фак;
ALTER TABLE Кафедра ADD CONSTRAINT Кафедра_Факультет FOREIGN KEY (факультет) REFERENCES Факультет (аббревиатура)
ON UPDATE CASCADE ON DELETE SET NULL);
```

В этом примере первая команда ALTER TABLE добавляет новое поле «факультет». Вторая — удаляет индекс, ассоциированный с внешним ключом «фак», который моделирует связь с таблицей «Факультет». Третья — удаляет столбец «фак». И, наконец, четвертая — присваивает столбцу «факультет» статус внешнего ключа, восстанавливая тем самым связь с таблицей «Факультет» и контроль ограничения ссылочной целостности.

Пример 10. Добавить новое поле «название» — альтернативный ключ таблицы «Дисциплина».

```
ALTER TABLE Дисциплина ADD COLUMN название TEXT (20);
ALTER TABLE Дисциплина ADD CONSTRAINT АК_Дисциплина UNIQUE (название);
```

Задание. В таблице «Дисциплина» создать поле «исполнитель», определенное на домене TEXT (8) — внешний ключ для связи с таблицей «Кафедра» по полю «шифр» («кафедра читает курс»). В качестве ограничения ссылочной целостности по связи между таблицами «Дисциплина» и «Кафедра» задайте стратегию ON UPDATE CASCADE ON DELETE SET NULL. Изменение шифра кафедры-исполнителя в таблице «Кафедра» будет передано по каскаду в таблицу-потомок «Дис-

циплина» и приведет к изменению внешнего ключа «исполнитель». Удаление строки родительской таблицы «Кафедра» не должно привести к удалению соответствующих строк.

После модификации структуры окончательный вариант схемы реляционной базы данных для предметной области «Кафедра» будет выглядеть, как на рис. 12.

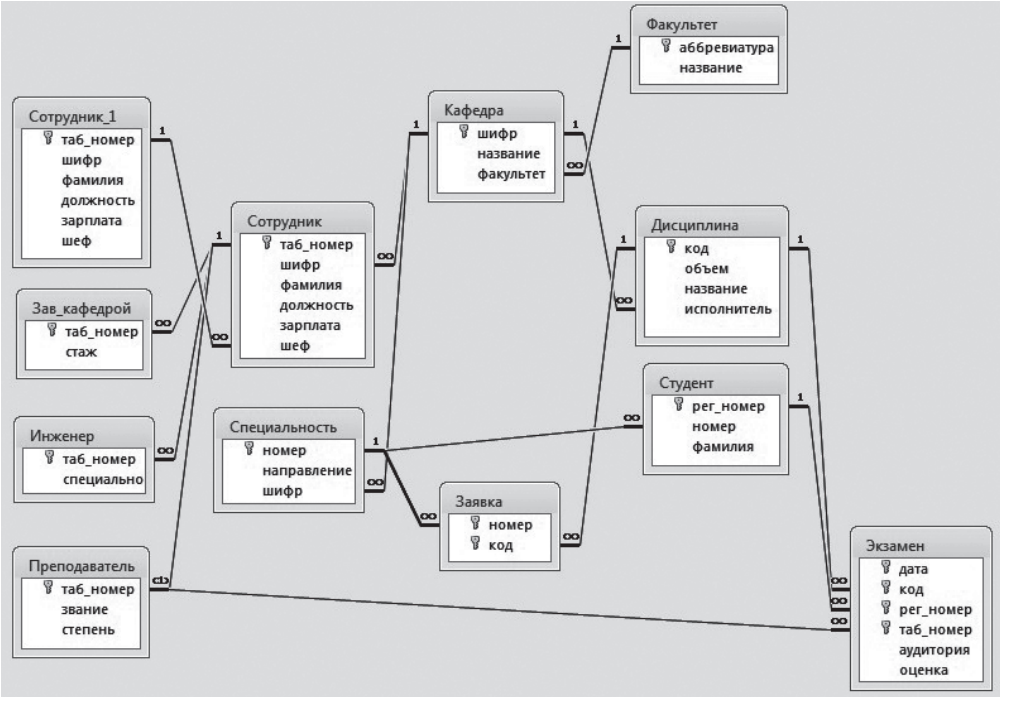

Рис. 12. Схема реляционной базы данных

Удаление таблицы

Созданную и заполненную таблицу можно удалить при помощи инструкции DROP TABLE.

DROP TABLE <имя таблицы>;

Перед удалением таблицы ее следует закрыть. При удалении таблицы, имеющей связанные с ней таблицы-потомки, в принципе возможны два исхода. Удаление родительской таблицы будет запрещено, пока не удалены все ее таблицы-потомки. Если разрешено ка-

скадирование, то удаление родительской таблицы повлечет за собой удаление и всех ее таблиц-потомков. В SQL Microsoft Jet реализуется первый случай.

Пример 11. Удалить таблицу «Экзамен». Ее удаление не должно повлечь за собой проблемы в отношении других связанных с ней таблиц.

DROP TABLE Экзамен;

После удаления таблицы «Экзамен» восстановите ее при помощи соответствующей сохраненной команды (SQL запроса) CREATE TABLE.

Язык манипулирования данными

Реальные языки манипулирования данными в реляционном представлении должны, как минимум, обладать свойством *реляционной полноты*, т. е. обеспечивать выполнение восьми реляционных операторов (объединение, пересечение, вычитание, декартово произведение, проекция, селекция, соединение и деление), введенных Коддом. Эти операторы лежат в основе всего многообразия запросов-выборок. Однако реальные языки обычно обеспечивают не только реляционную полноту, но и другие возможности, выходящие за пределы реляционной алгебры или реляционного исчисления:

- добавление данных;
- модификация данных;
- удаление данных;
- арифметические вычисления и сравнения;
- присваивание и отображение;
- вычисление функций агрегации, выполняемые над однородными элементами данных (вычисление количества, суммы, среднего, минимального или максимального значения и др.).

В связи с этим запросы, с которыми можно обращаться к базе данных, принято разделять на два класса:

- **Корректирующие запросы** — запросы на добавление (INSERT), удаление (DELETE) и изменение (UPDATE) данных, которые образуют подмножество SQL — **язык манипулирования данными**. Это весьма значительные по своим последствиям за-

просы, так как их реализация приводит к изменению хранимой в базе данных информации.

- **Запросы-выборки** (SELECT) — запросы, осуществляющие только извлечение информации из одной или нескольких связанных таблиц и представление ее в виде виртуальной (не существующей на самом деле) таблицы. Структура такого рода таблицы определяется локальным представлением о предметной области конечного пользователя базы данных в контексте запроса. Запросы на основе команды SELECT иногда выделяют в подмножество SQL — **язык запросов**. В данной главе рассмотрим подмножество языка манипулирования данными.

База данных одной и той же предметной области может быть представлена в виде одной таблицы (так называемого «универсального отношения») или в виде совокупности связанных таблиц. Выбор зависит от соотношения частот обращения к базе данных с корректирующими запросами или с запросами-выборками. Для первых предпочтительнее совокупность связанных таблиц. В теории баз данных есть на этот счет специальная методология, называемая нормализацией баз данных[3]. Для запросов-выборок предпочтительнее использование одной таблицы. В учебной базе данных используются только нормализованные таблицы.

Ввод (добавление) данных в таблицу

Новые данные можно добавлять в таблицу вручную или с помощью другой таблицы, используемой в качестве источника данных. Ввод данных в реляционную таблицу выполняется под контролем ссылочной целостности. Поэтому ввод данных в родительскую таблицу должен предшествовать вводу данных в таблицу-потомок.

Запрос на добавление одной записи выполняется при помощи инструкции INSERT INTO:

```
INSERT INTO <имя таблицы> [(<список имен столбцов приемника>)]
VALUES (<список значений>);
```

[3] Обсуждение этой проблемы выходит за рамки данного учебного пособия.

Списки столбцов и значений должны быть согласованы по типу, по количеству и по порядку следования. Список имен столбцов может отсутствовать, если он полный и согласован со списком значений.

Пример 12. Заполнить таблицу «Факультет» фактическими данными при помощи команды INSERT по образцу:

```
INSERT INTO Факультет (аббревиатура, название)
VALUES ('ит', 'Информационные технологии');
```

Команду INSERT INTO следует повторить три раза (по числу строк), каждый раз с новыми значениями вводимых данных.

Данные для наполнения таблицы «Факультет»

Аббревиатура	Название
ит	Информационные технологии
ен	Естественные науки
фм	Физико-математический

Другой формат команды INSERT формирует вводимые данные как результат подзапроса на добавление строк из другой таблицы.

INSERT INTO <имя таблицы-приемника> [(<список имен столбцов приемника>)] [IN <внешняя_база_данных>]
SELECT <список имен столбцов источника>
FROM <источник> [IN <внешняя_база_данных>];

Здесь:

- <список имен столбцов приемника> — имена столбцов таблицы-приемника для добавления данных;
- <список имен столбцов источника> — имена столбцов или одной таблицы, или нескольких таблиц, или результата соединения таблиц;
- <внешняя_база_данных> — полный путь к внешней базе данных;
- <источник> — имена таблицы или таблиц, откуда вставляются данные; это может быть имя отдельной таблицы или результат операции соединения, а также сохраненный запрос.

Пример 13. Заполнить по образцу при помощи инструкции INSERT таблицу «Кафедра» фактическими данными из таблицы «Кафедра» в базе данных (файле) Кафедра.mdb, размещенной в папке «Мои документы»:

INSERT INTO Кафедра SELECT * FROM Кафедра IN 'Кафедра.mdb';

Данные для наполнения таблицы «Кафедра»

шифр	название	факультет
вм	Высшая математика	ен
ис	Информационные системы	ит
мм	Математическое моделирование	фм
оф	Общая физика	фм
пи	Прикладная информатика	ит
эф	Экспериментальная физика	фм

Задание. Остальные таблицы заполните или «вручную»[4], или из внешних текстовых файлов, соблюдая порядок старшинства таблиц. Для импорта данных в базу данных MS Access из текстовых файлов воспользуйтесь следующим алгоритмом:

1. Закрыть целевую таблицу, если она открыта.
2. На вкладке «Внешние данные» в группе «Импорт» выбрать иконку «Импорт текстового файла».
3. В диалоге «Внешние данные — Текстовый файл» найти и выбрать по имени текстовый файл — источник данных.
4. В диалоге «Внешние данные — Текстовый файл» выбрать вариант «Добавить копию записей в конец таблицы», выбрать из списка целевую таблицу и нажать <ОК>.
5. В диалоге на первом шаге мастера «Импорт текста» выбрать «с разделителями» и нажать <Далее>.
6. На втором шаге выбрать разделитель полей «точка с запятой» и нажать <Далее>.
7. На третьем шаге убедиться, что данные импортируются в целевую таблицу, и нажать <Готово>.
8. На четвертом шаге без сохранения шагов импорта нажать <Закрыть>.

Результаты выполнения запросов на добавление контролируйте визуально.

[4] Пример наполнения таблиц фактическими данными приведен в приложении.

Задание. Добавить в таблицу «Студент» данные о новом студенте: «рег_номер» = '11111', «номер»= '11.11.11', «фамилия»= 'Одинцов О. О.' (вставить строку). Объясните причину неудачи.

Задание. Добавить в таблицу «Специальность» данные о новой специальности: «номер»= '11.11.11', «название»='Новая специальность', «шифр»='ис' (вставить строку).

Обновление данных

Инструкция UPDATE создает запрос на обновление, который изменяет значения столбцов указанной таблицы на основе заданного условия отбора:

UPDATE <имя таблицы>
SET <имя столбца1>=<выражение1>,
[<имя столбца2>=<выражение2>,...<имя столбцаN>=<выражениеN>]
[WHERE <условие>];

Все строки, удовлетворяющие условию отбора во фразе WHERE, или все строки, если WHERE опущено, обновляются в соответствии с присваиваниями <имя столбца>=<выражение> во фразе SET.

Условие отбора во фразе WHERE может быть задано не только в виде логического выражения, порождающего истинностное значение, но и так называемым *подзапросом*.

Пример 14. Увеличить зарплату инженерам на 10%:

```
UPDATE Сотрудник SET зарплата = 1.1*зарплата
WHERE должность='инженер';
```

Пример 15. Заменить в таблице «Студент» фамилию студента с данным номером на новую фамилию. Компоненты команды UPDATE могут быть заданы как параметры динамически в процессе выполнения запроса. Например, «рег_номер» = 10101, «фамилия» = Сергеева Н. Н.:

```
UPDATE Студент SET фамилия = [Введите новую фамилию]
WHERE рег_номер = [Введите номер студента];
```

Пример 16. Попробуйте изменить в таблице «Экзамен» номер студента (например, '10101') на '11111' и объясните причину неудачи:

UPDATE Экзамен SET рег_номер ='11111' WHERE рег_номер = '10101';

Невозможно обновить в одном запросе более одной таблицы.

Обновление ключевых столбцов в связанных таблицах порождает проблему сохранения целостности в переходном состоянии.

Результаты выполнения запросов на обновление контролируйте визуально.

Задание. Инженеров кафедры «Информационные системы» (шифр ='ис') перевести в подчинение сотрудника с табельным номером 201 (столбец «шеф»). Для конъюнкции двух условий использовать оператор AND.

Задание. Увеличить объем преподавания информатики на 20 часов.

Задание. Преподавателю с таб_номером = 102 присвоить звание профессора.

Задание. Перевести студента с заданным номером («регномер») на специальность с заданным номером («номер»). Компоненты команды UPDATE задать параметрически.

Удаление данных

Инструкция DELETE создает запрос на удаление строк, которые удовлетворяют условию отбора во фразе WHERE:

DELETE FROM <имя таблицы> [WHERE <условие>];

Команда DELETE удаляет из таблицы строки целиком, а не отдельные столбцы, поэтому список столбцов не указывается.

Если фраза WHERE опущена, DELETE удалит все строки таблицы. Однако сама *таблица удалена не будет*, в этом отличие команды DELETE от команды DROP. Условие отбора во фразе WHERE может быть

задано не только в виде логического выражения, порождающего истинностное значение, но и так называемого *подзапроса*.

Пример 17. Удалить из таблицы «Специальность» строку о специальности с «номером» 111111:

```
DELETE FROM Специальность
WHERE номер='11.11.11';
```

Результаты выполнения запросов на удаление контролируйте визуально.

Задание. Удалить из таблицы «Специальность» строку о специальности («номер») '09.03.03'. Объяснить причину неудачи.

Язык запросов

База данных представляет собой информационную модель предметной области и предназначена для удовлетворения информационных запросов пользователей, работающих в этой предметной области. После того как база данных будет наполнена содержательной информацией, к ней можно обращаться с запросами. **Запрос-выборка** представляет собой спецификацию условий манипулирования данными, в результате которого создается то или иное представление о хранимой в базе данных информации. База данных является объектом коллективного пользования, и каждый запрос выражает индивидуальное, локальное представление пользователей о структуре информации моделируемой предметной области.

Выборка данных по запросу осуществляется при помощи многофункциональной команды SELECT. Полный ее формат содержит множество спецификаторов ее действия. Основной результат выполнения операции можно выразить в виде фразы SELECT-FROM-WHERE. В команде задаются:

- само действие SELECT;
- источник данных FROM;
- условие отбора (факультативно) WHERE.

В общем случае инструкция SELECT имеет следующий формат:

SELECT [<спецификатор результата>] <список элементов>
FROM <ссылка на источник данных>
[WHERE.<условие отбора строк>]
[GROUP BY <список полей группировки строк>]
[HAVING <условие отбора групп>]
[ORDER BY <список полей сортировки строк>]

Порядок следования предложений команды SELECT имеет принципиальное значение и должен соответствовать указанному выше. Рассмотрим синтаксис команды по частям.

Здесь и далее в отличие от круглых скобок, которые являются элементами синтаксиса языка, угловые скобки <> обозначают метасимволы, квадратные скобки [] обозначают необязательные элементы, фигурные скобки {} обозначают альтернативу.

Инструкции SELECT не изменяют данные в базе данных. При ее выполнении ядро базы данных Microsoft Jet находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке.

Фраза SELECT в обобщенном виде содержит:

SELECT [<спецификатор результата>] <список элементов>;
Здесь
<спецификатор результата>={ALL|DISTINCT|DISTINCTROW|TOP n}

ALL (умолчание) — вывод всех строк;

DISTINCT — исключаются записи, которые содержат повторяющиеся комбинации значений в выбранных полях;

DISTINCTROW — опускает данные, основанные на целиком повторяющихся записях, а не отдельных повторяющихся полях. DISTINCTROW влияет на результат только в том случае, если в запрос включены *не все поля* из анализируемых таблиц. Предикат DISTINCTROW *игнорируется, если запрос содержит только одну таблицу или все поля всех таблиц запроса*.

TOP n — возвращает *n* записей, находящихся в начале результирующего списка строк.

<список элементов> не может быть пустым.

Элементы списка перечисляются через запятую. Элементами списка могут быть:

- [<имя таблицы>.]* — все поля таблицы;
- [<имя таблицы>.]<имя столбца> [AS <псевдоним>];
- скалярное выражение — вычисляемое значение;
- константа;
- ссылка на функцию агрегации (статистическую функцию);
- подзапрос.

Фраза FROM в общем виде содержит

FROM <ссылка на источник данных>

В качестве ссылки на источник данных могут использоваться:

- имя таблицы;
- список имен таблиц;
- соединение двух и более таблиц;
- подзапрос.

Предложение FROM должно присутствовать в каждой инструкции SELECT.

Порядок следования имен таблиц, если их несколько, значения не имеет.

В некоторых случаях в качестве разных источников данных есть необходимость использовать одну и ту же таблицу. В этом случае используются так называемые *псевдонимы* таблицы по следующей схеме:

```
SELECT <псевдоним1>.*, <псевдоним2>.*
FROM <имя таблицы1> AS <псевдоним1>,
     <имя таблицы2> AS <псевдоним2>;
```

Простые запросы

Выборка столбцов (проекция)

Минимальный формат инструкции SELECT включает список имен столбцов и имя таблицы-источника данных. Для отбора всех полей таблицы можно использовать символ звездочка (*). В этом случае в выборке будут представлены все столбцы, причем в таком порядке следования, в котором они были заданы в структуре таблицы. Если нужно вывести не все столбцы, а только некоторые, или изменить их порядок следования, то столбцы должны быть заданы в списке SELECT явно.

Пример 18. Вывести все поля таблицы «Сотрудник»:

```
SELECT *
FROM Сотрудник;
```

Результат выполнения запроса из примера 18

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	15 000,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	20 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	15 000,00 р.	202
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	20 000,00 р.	402
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	20 000,00 р.	602

Задание. Вывести некоторые поля таблицы «Сотрудник» («шифр», «должность» и «зарплата»).

Пример 19. Вывести список ставок кафедр как проекцию (выборку некоторых столбцов без дублей строк) таблицы «Сотрудник» на атрибуты «шифр», «должность» и «зарплата». Для выполнения проекции используется спецификатор DISTINCT.

```
SELECT DISTINCT шифр, должность, зарплата
FROM Сотрудник;
```

Результат выполнения запроса из примера 19

шифр	должность	зарплата
вм	зав.кафедрой	35 000,00 р.
вм	преподаватель	25 000,00 р.
ис	зав.кафедрой	35 000,00 р.
ис	инженер	16 500,00 р.
ис	инженер	22 000,00 р.
ис	преподаватель	25 000,00 р.
мм	зав.кафедрой	35 000,00 р.
мм	преподаватель	25 000,00 р.
оф	зав.кафедрой	35 000,00 р.
оф	инженер	22 000,00 р.
оф	преподаватель	15 000,00 р.
оф	преподаватель	25 000,00 р.
пи	зав.кафедрой	35 000,00 р.
пи	инженер	16 500,00 р.
пи	преподаватель	25 000,00 р.
эф	зав.кафедрой	35 000,00 р.
эф	инженер	22 000,00 р.
эф	преподаватель	25 000,00 р.

Задание. Выполнить проекцию таблицы «Экзамен» на атрибуты «код» и «таб_номер» (по какой дисциплине кто экзаменует).

Выборка строк по условию (селекция)

Предложение **WHERE** определяет, какие записи из таблиц, перечисленных в предложении FROM, следует включить в результат выполнения инструкции SELECT, UPDATE или DELETE.

Фраза WHERE в общем виде:

WHERE <условие отбора строк>

В спецификации условия отбора могут использоваться любые комбинации констант, литералов, функций, имен полей, а также операторов, порождающие истинностное значение:

- операции сравнения;
- оператор **Between <значение1> And <значение2>**;
- оператор **Like <шаблон>**;
- оператор **In (<список значений>)**;
- оператор инверсии **NOT <выражение>**;
- оператор конъюнкции **<выражение1> AND <выражение2>**;
- оператор дизъюнкции **<выражение1> OR <выражение2>**;
- подзапрос.

Ядро базы данных Microsoft Jet отбирает записи, соответствующие условиям, перечисленным в предложении WHERE. Если не задавать предложение WHERE, запрос возвращает все строки таблицы.

Предложение WHERE не является обязательным. Однако если оно присутствует, то должно следовать после предложения FROM.

Имена полей, которые содержат пробелы или знаки препинания, необходимо заключать в квадратные скобки ([]).

Аргументы условия отбора задаются как литералы. Литералы представляют собой константы, которые используются в SQL-командах. Существуют различные формы литералов в зависимости от типа данных, которые поддерживает конкретная версия SQL. Литералы числовых типов данных задаются как обычно в виде чисел, литералы строк символов заключаются в апострофы (например, должность = 'преподаватель'). Литералы даты следует заключать в знаки номера (#) и представлять в американском формате, даже если американская версия ядра базы данных Microsoft Jet не используется. Например, дата «10 мая 1996» записывается как #5/10/96#.

Предложение WHERE может содержать до 40 выражений, связанных логическими операторами, такими как And и Or.

При построении условий отбора могут использоваться арифметические, логические операторы и операторы сравнения.

Если выражение содержит несколько операторов, то значения компонентов выражения рассчитываются в определенном порядке. Такой порядок называют порядком старшинства или приоритетом операторов.

Если выражение содержит операторы разных типов, то первыми выполняются арифметические операции, следом за ними — опера-

ции сравнения, а последними — логические операции. Все операторы сравнения имеют равный приоритет, т. е. выполняются в порядке их расположения в выражении слева направо. Арифметические и логические операторы выполняются в порядке их расположения в табл. 16.

Таблица 16

Операторы, определенные над данными в SQL

Арифметические	Сравнения	Логические
Возведение в степень (^)	Равняется (=)	Not
Изменение знака (–)	Не равняется (<>)	And
Умножение и деление (*,/)	Меньше (<)	Or
Целое деление (\)	Больше (>)	Xor
Деление по модулю (Mod)	Меньше или равняется (<=)	Eqv
Сложение и вычитание (+,–)	Больше или равняется (>=)	Imp
Слияние строк (&)	Like	
	Is	

Стоящие рядом в выражении операторы умножения и деления выполняются слева направо. В таком же порядке выполняются стоящие рядом операторы сложения и вычитания. Операторы внутри круглых скобок всегда выполняются раньше, чем операторы вне скобок. Порядок выполнения операторов, стоящих внутри скобок, определяется старшинством операторов.

Оператор конкатенации (слияния строк) (&) не является арифметическим оператором, однако по порядку старшинства он следует сразу за арифметическими операторами и перед операторами сравнения.

Оператор Like, равный по старшинству остальным операторам сравнения, выделяется в самостоятельный тип оператора сравнения с образцом.

Оператор Is является оператором сравнения ссылок на объект. Этот оператор не выполняет сравнение объектов или их значений; он проверяет только, указывают ли две разные ссылки на один объект.

Спецификация условия отбора строк используется в самой распространенной реляционной операции — *операции селекции*.

Пример 20. Вывести все данные о сотрудниках-инженерах. Выполнить селекцию таблицы «Сотрудник» по условию «должность»='инженер'.

```
SELECT *
FROM Сотрудник
WHERE должность=’инженер’;
```

Результат выполнения запроса из примера 20

таб_номер	шифр	фамилия	должность	зарплата	шеф
153	пи	Сидорова С. С.	инженер	15 000,00 р.	102
241	ис	Глухов Г. Г.	инженер	20 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	15 000,00 р.	202
435	оф	Лисин Л. Л.	инженер	20 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	20 000,00 р.	602

Пример 21. Вывести все данные, а также как вычисляемое данное годовую зарплату («зарплата»*12) тех сотрудников (таблица «Сотрудник»), чья годовая зарплата не меньше 300 000 руб.

```
SELECT таб_номер, шифр AS кафедра, фамилия, должность, зарплата, шеф, зарплата*12 AS [годовая зарплата]
FROM Сотрудник
WHERE зарплата*12>=300000;
```

Результат выполнения запроса из примера 21

таб_номер	кафедра	фамилия	должность	зарплата	шеф	годовая зарплата
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101	420 000,00 р.
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101	300 000,00 р.
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101	300 000,00 р.
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201	420 000,00 р.
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201	300 000,00 р.
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301	420 000,00 р.
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301	300 000,00 р.
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401	420 000,00 р.
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401	300 000,00 р.
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501	420 000,00 р.
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501	300 000,00 р.

таб_ номер	ка-фе-дра	фамилия	должность	зарплата	шеф	годовая зарплата
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501	300 000,00 р.
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601	420 000,00 р.
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601	300 000,00 р.

Пример 22. Вывести данные об инженерах данной кафедры. Шифр кафедры задать параметрически (например, «шифр»='ис'). Выполнить селекцию таблицы «Сотрудник» по условию «должность»='инженер' и «шифр»= [Введите шифр кафедры] (использовать оператор AND).

```
SELECT *
FROM Сотрудник
WHERE должность='инженер' AND шифр= [Введите шифр кафедры];
```

Результат выполнения запроса из примера 22

таб_номер	шифр	фамилия	должность	зарплата	шеф
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202

Задание. Вывести список сотрудников данной кафедры. Шифр кафедры задать параметрически.

Задание. Вывести список дисциплин, которые ведет данная кафедра. Шифр кафедры (столбец «исполнитель») задать параметрически.

Задание. Вывести список сотрудников данной кафедры, чья зарплата менее заданной величины. Шифр кафедры и граничную величину зарплаты задать параметрически. Использовать оператор AND.

В условии выборки могут использоваться операторы Between, Like, In.

Селекция на основе BETWEEN… AND

Оператор BETWEEN… AND определяет принадлежность значения выражения (в частности, столбца реляционной таблицы) указанному диапазону и имеет следующий формат:

<выражение> [NOT] BETWEEN <значение1> AND <значение2>

Если значение выражения или анализируемого столбца попадает в диапазон, задаваемый аргументами «значение1» и «значение2» (включительно), оператор BETWEEN… AND возвращает значение True; в противном случае возвращается значение False. Логический оператор NOT позволяет проверить противоположное условие (что выражение находится за пределами диапазона, заданного с помощью аргументов «значение1» и «значение2»). Порядок следования границ диапазона значения не имеет. Более того, границы могут совпадать. Оператор BETWEEN… AND можно применять к данным тех типов, для которых понятие диапазона имеет смысл. Границы диапазона могут быть заданы параметрически.

Пример 23. Вывести из таблицы «Сотрудник» данные о сотрудниках, имеющих зарплату в диапазоне, заданном параметрически (например, от 15 000 до 20 000). Запрос оформить как выборку с BETWEEN… AND.

```
SELECT *
FROM Сотрудник
WHERE зарплата BETWEEN [от] AND [до];
```

Результат выполнения запроса из примера 23

таб_номер	шифр	фамилия	должность	зарплата	шеф
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401

Задание. Вывести из таблицы «Экзамен» все данные о результатах сдачи экзаменов в диапазоне заданных дат. Запрос оформить как параметрический с BETWEEN… AND.

Селекция на основе LIKE

Оператор LIKE используется для сравнения строкового выражения с образцом в выражении SQL:

<выражение> LIKE <шаблон>

Оператор LIKE исследует анализируемое выражение, в том числе значение столбца, на совпадение с заданным шаблоном (образцом сравнения) с точностью до символов шаблона. В качестве символов шаблона стандарта ANSI SQL могут использоваться, например:

- % — любая последовательность символов;
- _ (подчеркивание) — один любой символ.

Оператор LKE применим только к данным текстового типа.

Для образца можно задавать полное значение (например, LIKE 'Иванов') или использовать символы шаблона для поиска диапазона значений (например, LIKE 'Ив%', здесь «Ив» — первые буквы, % — любые комбинации любых символов).

Пример 24. Вывести данные о сотрудниках (таблица «Сотрудник»), чья фамилия начинается на «С».

```
SELECT *
FROM Сотрудник
WHERE фамилия Like 'С%';
```

Результат выполнения запроса из примера 24

таб_номер	шифр	фамилия	должность	зарплата	шеф
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601

Шаблон оператора LIKE может быть задан параметрически, как в следующем примере.

Пример 25. Вывести данные о сотрудниках (таблица «Сотрудник») по первым буквам фамилии (например, по одной букве «Б»).

```
SELECT *
FROM Сотрудник
WHERE фамилия LIKE [Введите первые буквы фамилии]&'%';
```

Здесь первые буквы фамилии задаются параметрически, а остальные могут быть любыми.

Результат выполнения запроса из примера 25

таб_номер	шифр	фамилия	должность	зарплата	шеф
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301

Задание. Вывести данные о сотрудниках (таблица «Сотрудник») по первым буквам названия должности. Запрос оформить как параметрический запрос.

Селекция на основе IN

Оператор IN проверяет, попадает ли значение выражения или анализируемого столбца в список «разрешенных» значений:

<выражение> [NOT] IN (<список значений>)

Если «выражение» содержится в списке значений, оператор IN возвращает значение True; в противном случае возвращается значение False. С помощью логического оператора NOT можно проверить обратное условие (что «выражение» не принадлежит списку значений).

Пример 26. Вывести из таблицы «Дисциплина» данные о дисциплинах базового цикла {математика, физика, информатика}. Запрос оформить как выборку с IN.

```
SELECT *
FROM Дисциплина
WHERE название IN ('математика', 'физика', 'информатика');
```

Результат выполнения запроса из примера 26

код	объем	название	заявитель	исполнитель
101	320	математика	пи	вм
102	160	информатика	пи	пи
103	160	физика	ис	оф

Пример 27. Вывести из таблицы «Студент» данные о студентах, обучающихся по специальностям факультета «Информационные технологии», т. е. с «номерами» из списка {09.03.02, 09.03.03, 38.03.05}. Запрос оформить как выборку с IN.

```
SELECT *
FROM Студент
WHERE номер IN ('09.03.02', '09.03.03', '38.03.05');
```

Результат выполнения запроса из примера 27

рег_номер	номер	фамилия
10101	09.03.03	Николаева Н. Н.
10102	09.03.03	Иванов И. И.
20101	09.03.02	Андреев А. А.
80101	38.03.05	Макаров М. М.
80102	38.03.05	Яковлев Я. Я.
20102	09.03.02	Федоров Ф. Ф.
10103	09.03.03	Крюков К. К.

Задание. Вывести из таблицы «Сотрудник» данные о сотрудниках, работающих на кафедрах факультета «Естественные науки», т. е. с «шифрами» из списка {оф, вм}. Запрос оформить как выборку с IN.

Сортировка строк

Предложение ORDER BY сортирует записи, полученные в результате запроса, в порядке возрастания или убывания на основе значений указанных имен столбцов.

Фраза ORDER BY в общем случае имеет вид

ORDER BY <имя столбца1> [{ASC|DESC}]
[,<имя столбца2> [{ASC|DESC}]] [, …]];

В запросе на сортировку записей должны быть выполнены следующие условия:

- сортировка может выполняться по одному или нескольким столбцам;
- порядок сортировки определяется порядком следования столбцов в списке ODER BY: сначала по первому столбцу, затем по второму и т. д.;
- столбцы в списке ODER BY должны быть представлены своими именами с указанием спецификатора способа сортировки;
- столбцы в списке сортировки должны быть явно или не явно (в виде *) присутствовать в списке фразы SELECT;
- сортировка может выполняться по возрастанию или по убыванию значений столбцов сортировки;
- для сортировки по возрастанию используется спецификатор ASC (умолчание); для сортировки по убыванию — спецификатор DESC.

Предложение ORDER BY является необязательным. Однако оно необходимо для отображения данных в порядке сортировки.

При включении поля Memo или объекта ActiveX в предложение ORDER BY возникает ошибка. Ядро базы данных Microsoft Jet не поддерживает сортировку по полям этих типов.

Предложение ORDER BY является последним элементом инструкции SQL.

Пример 28. Вывести из таблицы «Сотрудник» данные о сотрудниках в алфавитном порядке шифров кафедр («шифр» ASC) и по убыванию зарплаты («зарплата» DESC).

```
SELECT *
FROM Сотрудник
ORDER BY шифр, зарплата DESC;
```

Результат выполнения запроса из примера 28

таб_номер	шифр	фамилия	должность	зарплата	шеф
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201

таб_номер	шифр	фамилия	должность	зарплата	шеф
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Задание. Вывести из таблицы «Студент» данные о студентах, упорядоченные по возрастанию номеров специальностей («номер») и в алфавитном порядке фамилий («фамилия»). Запрос оформить как выборку с ORDER BY.

Группировка строк

Однородные в определенном отношении строки реляционной таблицы могут быть объединены в группы. Предложение GROUP BY объединяет записи с одинаковыми значениями в указанном списке столбцов в одну группу.

Фраза GROUP BY в общем случае имеет следующий вид:

GROUP BY <список столбцов группировки строк>

Если в списке столбцов указано не одно поле, то создаются вложенные группы — подгруппы, подподгруппы и т. д. по числу элементов списка.

В запросе на группировку записей должны быть выполнены следующие условия:

- группировка может выполняться по одному или нескольким столбцам;
- порядок группировки определяется порядком следования столбцов в списке GROUP BY (сначала по первому столбцу, затем по второму и т. д.);
- список полей группировки в предложении GROUP BY и список элементов предложения SELECT должны быть согласованы;
- в списке SELECT можно использовать только те поля, по которым выполняется группировка, а также вычисляемые поля, например на основе статистических функций: COUNT, SUM, AVG, MAX, MIN и др.

Предложение GROUP BY является необязательным.

Итоговые значения не рассчитываются, если инструкция SELECT не содержит статистической функции SQL.

Значения Null, которые находятся в полях, заданных в предложении GROUP BY, группируются и не опускаются. Однако статистические функции SQL не обрабатывают значения Null.

Если столбец, включенный в предложение GROUP BY, не является данным типа Memo или объекта ActiveX, то это предложение может содержать ссылку на любой столбец, перечисленный в предложении FROM, даже если этот столбец не включен в инструкцию SELECT при условии, что инструкция SELECT содержит, по крайней мере, одну статистическую функцию SQL. Ядро базы данных Jet не поддерживает группировку полей MEMO или объекта ActiveX.

Пример 29. Сформировать список кафедр и их должностей путем вывода из таблицы «Сотрудник» шифров кафедр («шифр») и должности сотрудников («должность»), сгруппировав их по этим атрибутам.

```
SELECT шифр, должность
FROM Сотрудник
GROUP BY шифр, должность;
```

Результат выполнения запроса из примера 29

шифр	должность
вм	зав.кафедрой
вм	преподаватель

шифр	должность
ис	зав.кафедрой
ис	инженер
ис	преподаватель
мм	зав.кафедрой
мм	преподаватель
оф	зав.кафедрой
оф	инженер
оф	преподаватель
пи	зав.кафедрой
пи	инженер
пи	преподаватель
эф	зав.кафедрой
эф	инженер
эф	преподаватель

Задание. Вывести из таблицы «Студент» под заголовком «номера специальностей» номера специальностей, сгруппировав строки по этому атрибуту. Запрос оформить как выборку с GROUP BY.

Вычисляемые выражения и статистические функции SQL

В списке элементов предложения SELECT могут использоваться данные, которые отсутствуют в реляционных таблицах, но которые могут быть вычислены по формулам в виде выражений.

Статистические функции SQL (AVG, Count, MIN, MAX, StDev, StDevP, SUM, Var, VarP) представляют собой частный случай вычисляемых данных. Вызов статистической функции имеет следующий формат:

Функция (выражение)

Аргумент «выражение» является строковым выражением, которое определяет столбец, содержащий числовые данные для вычисления среднего значения, или выражение, выполняющее вычисления с данными из этого поля. Операнды аргумента «выражение» могут включать имя столбца таблицы, константу или функцию (последняя может быть внутренней или определяться пользователем, но не может быть другой статистической функцией SQL).

Результат вычисления статистической функции может быть ограничен условием, задаваемым в предложении WHERE.

Функция Count

Функция Count вычисляет количество записей, возвращаемых запросом.

Count (выражение)

Подсчитывать можно любые данные, включая текстовые.

Функцию Count используют для подсчета количества записей в базовом запросе. Хотя аргумент-выражение допускает выполнение вычислений над столбцом, функция Count возвращает только количество записей независимо от того, какие данные содержатся в этих записях.

Функция Count не подсчитывает записи со значениями Null, если только аргумент-выражение не содержит подстановочный знак звездочки (*). Если используются знаки звездочки, то функция Count вычисляет общее количество записей, включая те, которые содержат пустые столбцы. Функция Count (*) работает значительно быстрее функции Count (<имя столбца>).

Пример 30. Вывести из таблицы «Студент» количество студентов (Count (*)) как поле с заголовком «количество студентов».

```
SELECT Count (*) AS [количество студентов]
FROM Студент;
```

Результат выполнения запроса из примера 30

количество студентов
12

Если в аргументе-выражении задано несколько столбцов, функция Count подсчитывает запись только в том случае, если хотя бы одно из полей не содержит значение Null. Если все указанные поля содержат значения Null, то запись не подсчитывается. Для разделения имен полей используется символ (&).

Функция AVG

Функция AVG применяется только к числовым данным и вычисляет арифметическое среднее набора чисел, содержащихся в указанном столбце выражения-аргумента.

AVG (выражение)

Среднее значение, вычисленное функцией AVG, является числовым значением (сумма значений, деленная на их количество).

Функция AVG не включает в вычисления поля со значениями Null.

Функции MIN, MAX

Функции MIN и MAX применяются как к числовым, так и нечисловым данным и возвращают минимальное и максимальное значения из набора значений, содержащихся в указанном столбце выражения-аргумента:

Min (выражение)
Max (выражение)

Функции Min и Max используются для определения наименьшего и наибольшего значений из столбца на основе выборки или группировки. Например, можно применить эти функции для возврата наименьшей и наибольшей стоимости доставки. Если не указан способ группировки, то используется вся таблица.

Пример 31. Вычислить значение минимальной, максимальной и средней зарплаты сотрудников.

```
SELECT MIN (зарплата) AS [минимальная зарплата], MAX (зарплата) AS [максимальная зарплата], AVG (зарплата) AS [средняя зарплата]
FROM Сотрудник;
```

Результат выполнения запроса из примера 31

минимальная зарплата	максимальная зарплата	средняя зарплата
15 000,00 р.	35 000,00 р.	26 200,00 р.

Функция SUM

Функция SUM применяется только к числовым данным и возвращает сумму набора значений, содержащихся в заданном выражении-аргументе.

SUM (выражение)

Функция SUM выполняет суммирование значений в столбце. Функция SUM пропускает записи со столбцами, содержащими значения Null.

Пример 32. Вычислить сумму ежемесячных зарплат сотрудников-инженеров.

```
SELECT SUM (зарплата) AS [сумма зарплат инженеров]
FROM Сотрудник
WHERE должность='инженер';
```

Результат выполнения запроса из примера 32

сумма зарплат инженеров
99 000,00 р.

Функции StDev, StDevP

Функции StDev, StDevP возвращают смещенное и несмещенное значения среднего квадратического отклонения, вычисляемого по набору значений, содержащихся в указанном столбце запроса.

StDev (выражение)
StDevP (выражение)

Функции StDevP и StDev вычисляют величину смещенного и несмещенного среднего квадратического отклонения.

Если базовый запрос содержит меньше двух записей (или не содержит записей для функции StDevP), то эти функции возвращают значение Null (что означает невозможность вычисления среднеквадратичного отклонения).

Пример 33. Вывести из таблицы «Сотрудник» среднюю зарплату (AVG (зарплата)) и ее среднее квадратическое отклонение (StDev (зарплата)) как поля с заголовками «Средняя зарплата» и «Среднее квадратическое отклонение».

SELECT AVG (зарплата) AS [средняя зарплата], StDev (зарплата) AS [среднее квадратическое отклонение]
FROM Сотрудник;

Результат выполнения запроса из примера 33

средняя зарплата	среднее квадратическое отклонение
26 200,00 р.	6659,77714183796

Функции Var, VarP

Функции Var, VarP возвращают значения смещенной и несмещенной дисперсии, вычисляемой по набору значений, содержащихся в указанном столбце запроса:

Var (выражение)
VarP (выражение)

Функции VarP и Var вычисляют значение смещенной и несмещенной дисперсии.

Если базовый запрос содержит меньше двух записей, то функции Var и VarP возвращают значение Null (что означает невозможность вычисления дисперсии).

Статистические функции (функции группирования) часто используются в запросах группирования для вычисления интегральных характеристик групп.

Пример 34. Вывести из таблицы «Сотрудник» шифр кафедры («шифр»), среднюю зарплату (AVG (зарплата)) и количество сотрудников (Count (зарплата)) на каждой кафедре как данные с соответствующими заголовками.

SELECT шифр AS кафедра, AVG (зарплата) AS [средняя зарплата], Count (зарплата) AS [количество сотрудников]
FROM Сотрудник
GROUP BY шифр;

Результат выполнения запроса из примера 34

кафедра	средняя зарплата	количество сотрудников
вм	28 333,33 р.	3
ис	24 625,00 р.	4

кафедра	средняя зарплата	количество сотрудников
мм	30 000,00 р.	2
оф	24 250,00 р.	4
пи	25 375,00 р.	4
эф	27 333,33 р.	3

Задание. Вывести из таблицы «Сотрудник» штатное расписание кафедр, т. е. «шифр», «должность», «зарплату» и «количество ставок» (Count (зарплата)), сгруппировав строки по этим полям.

Задание. Вывести из таблицы «Экзамен» номера студентов («рег_номер»), количество сданных экзаменов (Count (оценка)) и средний балл (AVG (оценка)) для каждого студента как поля с соответствующими заголовками.

Выборка групп

Если для отбора строк, например подлежащих группировке, используется предложение WHERE, то для отбора групп применяется предложение HAVING.

Предложение HAVING определяет, какие сгруппированные записи отображаются при использовании инструкции SELECT с предложением GROUP BY. Фраза HAVING в общем случае имеет следующий вид:

HAVING <условие отбора групп>

В качестве условия отбора групп могут использоваться:

- операции сравнения;
- оператор **Between <значение1> And <значение2>**;
- оператор **Like <шаблон>**;
- оператор **In (<список значений>)**;
- оператор инверсии **NOT <выражение>**;
- оператор конъюнкции **<выражение1> AND <выражение2>**;
- оператор дизъюнкции **<выражение1> OR <выражение2>**;
- подзапрос.

Предложение HAVING является необязательным. Без предложения GROUP BY не используется.

Предложение HAVING похоже на предложение WHERE, которое определяет, какие записи должны быть отобраны.

После того как записи будут сгруппированы с помощью предложения GROUP BY, предложение HAVING отберет те из полученных групп записей, которые удовлетворяют условиям отбора, указанным в предложении HAVING.

Предложение HAVING может содержать до 40 выражений, связанных логическими операторами, такими как And и Or.

Пример 35. Вывести из таблицы «Сотрудник» название кафедры («шифр»), среднюю зарплату (AVG (зарплата)) и количество сотрудников (Count (зарплата)) только на тех кафедрах, где средняя зарплата превышает заданную границу (например, 25000).

```
SELECT шифр AS кафедра, AVG (зарплата) AS [средняя зарплата],
Count (зарплата) AS [количество сотрудников]
FROM Сотрудник
GROUP BY шифр
HAVING AVG (зарплата)> [Введите границу];
```

Результат выполнения запроса из примера 35

кафедра	средняя зарплата	количество сотрудников
вм	28 333,33 р.	3
мм	30 000,00 р.	2
пи	25 375,00 р.	4
эф	27 333,33 р.	3

Задание. Вывести из таблицы «Экзамен» номера студентов («рег_номер»), количество сданных экзаменов (Count (оценка)) и средний балл (AVG (оценка)) как поля с соответствующими заголовками для тех студентов, у которых средний балл не меньше заданного. Запрос оформить со спецификатором условий отбора групп HAVING.

Запросы к связанным таблицам

Во всех ранее рассмотренных примерах демонстрировались разнообразные средства спецификации простых запросов. Все рассмотренные приемы построения простых запросов могут быть использованы не только в чистом виде, но и во всех возможных сочетаниях

друг с другом. Рассмотренные типы запросов можно назвать простыми, поскольку все они использовали в качестве источника данных одну таблицу. В реальных информационных задачах гораздо чаще запросы адресуются не к одной, а к нескольким связанным таблицам, которые представляют все разнообразие данных конкретной предметной области.

В реальных задачах выборка данных осуществляется из нескольких связанных между собой таблиц.

В SELECT предложении выборки из связанных таблиц могут присутствовать одноименные столбцы. Во избежание неопределенности одноименные столбцы в такого рода запросах должны упоминаться с префиксом (через точку) в виде имени соответствующей таблицы: **<имя таблицы>.<имя столбца>.** Если нет проблемы относительно происхождения столбца, имя таблицы можно не указывать.

Декартово произведение

В *декартовом произведении* выводятся все возможные комбинации строк-кортежей перемножаемых таблиц, даже такие, которые не имеют смысла в моделируемой предметной области.

Пример 36. Вывести декартово произведение таблиц «Факультет» и «Кафедра».

```
SELECT Факультет.*, Кафедра.*
FROM Факультет, Кафедра;
```

Здесь спецификация типа «Кафедра.*» означает «все поля таблицы Кафедра».

Результат выполнения запроса из примера 36

аббревиатура	Факультет.название	шифр	Кафедра.название	факультет
ит	Информационные технологии	пи	Прикладная информатика	ит
ен	Естественные науки	пи	Прикладная информатика	ит

аббревиатура	Факультет.название	шифр	Кафедра.название	факультет
фм	Физико-математический	пи	Прикладная информатика	ит
ит	Информационные технологии	ис	Информационные системы	ит
ен	Естественные науки	ис	Информационные системы	ит
фм	Физико-математический	ис	Информационные системы	ит
ит	Информационные технологии	оф	Общая физика	ен
ен	Естественные науки	оф	Общая физика	ен
фм	Физико-математический	оф	Общая физика	ен
ит	Информационные технологии	вм	Высшая математика	ен
ен	Естественные науки	вм	Высшая математика	ен
фм	Физико-математический	вм	Высшая математика	ен
ит	Информационные технологии	мм	Математическое моделирование	фм
ен	Естественные науки	мм	Математическое моделирование	фм
фм	Физико-математический	мм	Математическое моделирование	фм
ит	Информационные технологии	эф	Экспериментальная физика	фм
ен	Естественные науки	эф	Экспериментальная физика	фм
фм	Физико-математический	эф	Экспериментальная физика	фм

Задание. Вывести декартово произведение трех таблиц «Кафедра», «Специальность» и «Студент».

Естественное соединение

В запросах выборки из связанных таблиц предложение SELECT задает структуру выборки; предложение FROM — источник данных, предложение WHERE — как минимум, условие соединения.

Условие соединения может иметь следующий формат:
<имя таблицы1>.<имя столбца1>
<оператор>
<имя таблицы2>.<имя столбца2>

Здесь оператор — это один из операторов сравнения.

Важным частным случаем соединения является так называемое **эквисоединение** — соединение по условию равенства столбцов, по которым устанавливается связь между таблицами.

Пример 37. Вывести эквисоединение таблиц «Кафедра» и «Специальность».

```
SELECT Кафедра.*, Специальность.*
FROM Кафедра, Специальность
WHERE Кафедра.шифр=Специальность.шифр;
```

Результат выполнения запроса из примера 37

Кафедра. шифр	название	факультет	номер	направление	Специальность. шифр
ис	Информационные системы	ит	09.03.02	Информационные системы и технологии	ис
ис	Информационные системы	ит	38.03.05	Бизнес-информатика	ис
мм	Математическое моделирование	фм	01.03.04	Прикладная математика	мм
пи	Прикладная информатика	ит	09.03.03	Прикладная информатика	пи
эф	Экспериментальная физика	фм	14.03.02	Ядерные физика и технологии	эф

Важной разновидностью соединения является так называемое естественное соединение, или эквисоединение без дублей одноименных столбцов (полей).

Операция *естественного (внутреннего) соединения* лежит в основе большинства выборок из связанных таблиц. Если таблицы

<имя таблицы1> и **<имя таблицы2>** связаны по столбцам **<имя столбца1>** и **<имя столбца2>**, то условие соединения может быть задано во фразе WHERE в запросе с обобщенным форматом:

SELECT <список элементов>
FROM <имя таблицы1>, <имя таблицы2
WHERE <имя таблицы1>.<имя поля1>=<имя таблицы2>.<имя поля2>;

Пример 38. Вывести данные о том, на каких кафедрах готовят специалистов по тем или иным направлениям (специальностям). Запрос выполнить на основе естественного соединения таблиц «Кафедра» и «Специальность».

SELECT факультет, Кафедра.шифр, название, номер, направление
FROM Кафедра, Специальность
WHERE Кафедра.шифр=Специальность.шифр;

Реализация этого запроса дает ответ на вопрос о том, по каким специальностям, на каких кафедрах и на каких факультетах обучают.

Результат выполнения запроса из примера 38

факультет	шифр	название	номер	направление
ит	ис	Информационные системы	09.03.02	Информационные системы и технологии
ит	ис	Информационные системы	38.03.05	Бизнес-информатика
фм	мм	Математическое моделирование	01.03.04	Прикладная математика
ит	пи	Прикладная информатика	09.03.03	Прикладная информатика
фм	эф	Экспериментальная физика	14.03.02	Ядерные физика и технологии

Задание. Вывести данные о том, на каких факультетах и кафедрах, по каким специальностям обучаются студенты. Запрос выполнить как естественное соединение таблиц «Кафедра», «Специальность» и «Студент».

Запрос на соединение можно выполнить также при помощи специальной инструкции SQL INNER JOIN.

Условие соединения содержится в предложении FROM:

SELECT <список элементов>
FROM <имя таблицы1> INNER JOIN <имя таблицы2
ON <имя таблицы1>.<имя столбца1>=<имя таблицы2>.<имя столбца2>;

Операцию INNER JOIN можно использовать в любом предложении FROM. Это самые обычные типы связывания. Они объединяют записи двух таблиц, если связующие столбцы обеих таблиц содержат одинаковые значения.

Попытка объединить данные Memo или объекта ActiveX приведет к возникновению ошибки.

Обязательным условием корректного соединения является совместимость типов столбцов, по которым осуществляется связь. Допускается объединение двух числовых полей подобных типов, например поле счетчика с полем типа «Длинное целое». Однако нельзя объединить типы полей «С плавающей точкой (4 байт)» и «С плавающей точкой (8 байт)».

Операции INNER JOIN могут быть вложенными; в таком случае используйте следующий синтаксис:

SELECT <список элементов>
FROM <имя таблицы1> INNER JOIN
(<имя таблицы2> INNER JOIN [(]<имя таблицы3>
[INNER JOIN [(]<имя таблицаN> [INNER JOIN …)]
ON <имя таблицы3>.<имя столбца3> = <имя таблицыN>.<имя столбцаN>)]
ON <имя таблицы2>.<имя столбца2> = <имя таблицы3>.<имя столбца3>)
ON <имя таблицы1>.<имя столбца1> = <имя таблицы2>.<имя столбца2>;

Пример 39. Вывести данные о том, на каких кафедрах готовят специалистов по тем или иным направлениям (специальностям). Запрос выполнить на основе естественного соединения таблиц «Кафедра» и «Специальность» при помощи INNER JOIN.

```
SELECT факультет, Кафедра.шифр, название, номер, направление
FROM Кафедра INNER JOIN Специальность
ON Кафедра.шифр=Специальность.шифр;
```

Результат выполнения запроса из примера 39

факультет	шифр	название	номер	направление
ит	ис	Информационные системы	09.03.02	Информационные системы и технологии
ит	ис	Информационные системы	38.03.05	Бизнес-информатика
фм	мм	Математическое моделирование	01.03.04	Прикладная математика
ит	пи	Прикладная информатика	09.03.03	Прикладная информатика
фм	эф	Экспериментальная физика	14.03.02	Ядерные физика и технологии

Задание. Вывести данные о том, на каких факультетах и кафедрах, по каким специальностям обучаются студенты. Запрос выполнить как естественное соединение таблиц «Кафедра», «Специальность» и «Студент» на основе команды INNER JOIN.

Естественное соединение является средством интеграции семантически связанных данных из разных таблиц и в сочетании с условиями проекции и селекции широко используется в запросах-выборках.

Пример 40. Вывести из таблиц «Кафедра» и «Сотрудник» данные о факультетах, кафедрах, должностях и фамилиях заведующих кафедрами

```
SELECT факультет, Кафедра.шифр, должность, фамилия
FROM Кафедра, Сотрудник
WHERE Кафедра.шифр=Сотрудник.шифр AND должность= ‘зав. кафедрой’;
```

Результат выполнения запроса из примера 40

факультет	шифр	должность	фамилия
ит	пи	зав.кафедрой	Прохоров П. П.
ит	ис	зав.кафедрой	Андреев А. А.
фм	мм	зав.кафедрой	Басов Б. Б.

факультет	шифр	должность	фамилия
ен	оф	зав.кафедрой	Волков В. В.
ен	вм	зав.кафедрой	Кузнецов К. К.
фм	эф	зав.кафедрой	Зверев З. З.

Пример 41. Вывести из таблиц «Кафедра», «Специальность» и «Студент» данные о студентах, которые обучаются на данном факультете (например, «ит»).

```
SELECT факультет, Кафедра.шифр, Специальность.номер, рег_номер, фамилия
FROM Кафедра, Специальность, Студент
WHERE Кафедра.шифр=Специальность.шифр AND
Специальность.номер=Студент.номер AND
факультет= [Введите аббревиатуру факультета];
```

Результат выполнения запроса из примера 41

факультет	шифр	номер	рег_номер	фамилия
ит	пи	09.03.03	10101	Николаева Н. Н.
ит	пи	09.03.03	10102	Иванов И. И.
ит	пи	09.03.03	10103	Крюков К. К.
ит	ис	09.03.02	20101	Андреев А. А.
ит	ис	09.03.02	20102	Федоров Ф. Ф.
ит	ис	38.03.05	80101	Макаров М. М.
ит	ис	38.03.05	80102	Яковлев Я. Я.

Пример 42. Вывести из таблиц «Кафедра», «Специальность» и «Сотрудник» данные о выпускающих кафедрах (факультет, шифр, название, фамилию заведующего). Выпускающей считается та кафедра, на которую есть ссылки в таблице «Специальность».

```
SELECT DISTINCTROW факультет, Кафедра.шифр AS [выпускающая кафедра], название, фамилия AS [заведующий кафедрой]
FROM Кафедра, Специальность, Сотрудник
WHERE Кафедра.шифр=Сотрудник.шифр AND
Кафедра.шифр=Специальность.шифр AND
должность=’зав.кафедрой’;
```

Результат выполнения запроса из примера 42

факультет	выпускающая кафедра	название	заведующий кафедрой
ит	пи	Прикладная информатика	Прохоров П. П.
ит	ис	Информационные системы	Андреев А. А.
фм	мм	Математическое моделирование	Басов Б. Б.
фм	эф	Экспериментальная физика	Зверев З. З.

В данном примере используется спецификатор результата DISTINCTROW, который позволяет избавиться от дублей строк, формируемых из связанных таблиц.

Задание. Вывести все данные о преподавателях: «Сотрудник. таб_номер», «шифр», «фамилия», «должность», «зарплата», «звание», «степень» — как запрос на естественное соединение таблиц «Сотрудник» и «Преподаватель», связанных *категориальной связью*.

Задание. Вывести данные о том, какие дисциплины преподают разным специальностям. Вывести «Специальность.номер», «Специальность.шифр», «Дисциплина.код», «Дисциплина.название». Запрос оформить как естественное соединение таблиц «Специальность» и «Дисциплина», связанных *ассоциативной таблицей* «Заявка».

Рассмотрим важный частный случай соединения — соединение таблицы самой с собой. Такой случай может встретиться при реализации *рекурсивной связи*.

Допустим, что для каждого сотрудника нужно вывести номер и фамилию его непосредственного начальника. И рядовой сотрудник, и его непосредственный начальник представлены в одной и той же таблице «Сотрудник». Соединение по рекурсивной связи позволяет коррелировать информацию из разных строк одной и той же таблицы и комбинировать ее для вывода. Для реализации такого соединения одна и та же таблица должна дважды упоминаться во фразе FROM. Для того чтобы различать эти два разные ее вхождения во фразе FROM одна и та же таблица «Сотрудник» должна быть представлена своими разными псевдонимами. Введение псевдонимов позволяет оперировать одной и той же таблицей как разными таблицами. Для того чтобы различать эти разные ее вхождения, таблица должна быть представлена (хотя бы в одном случае) своими **псевдонимами**

(«Сотрудник1» в данном примере). Введение псевдонимов позволяет оперировать одной таблицей как разными таблицами.

Пример 43. Вывести в запросе для каждого сотрудника номер и фамилию его непосредственного руководителя.

```
SELECT Сотрудник.шифр, Сотрудник.таб_номер, Сотрудник.фамилия, Сотрудник1.таб_номер, Сотрудник1.фамилия
FROM Сотрудник, Сотрудник Сотрудник1
WHERE Сотрудник1.таб_номер=Сотрудник.шеф;
```

В данном примере «шеф» — это имя роли атрибута «таб_номер», который в одной и той же таблице играет роль первичного ключа и внешнего ключа по рекурсивной связи.

Задание. Вывести данные («факультет», «Кафедра.шифр», «Кафедра.название», «код», «Дисциплина.название», «объем») о кафедрах и читаемых ими курсах. Запрос оформить на основе естественного соединения таблиц «Кафедра» и «Дисциплина» по полям «Кафедра.шифр» и «Дисциплина.исполнитель». Здесь «исполнитель» — имя роли, которую играет атрибут «шифр» в таблице «Дисциплина».

Результат выполнения запроса из примера 43

шифр	Сотрудник. таб_номер	Сотрудник. фамилия	Сотрудник1. таб_номер	Сотрудник1. фамилия
пи	101	Прохоров П. П.	101	Прохоров П. П.
пи	102	Семенов С. С.	101	Прохоров П. П.
пи	105	Петров П. П.	101	Прохоров П. П.
пи	153	Сидорова С. С.	102	Семенов С. С.
ис	201	Андреев А. А.	201	Андреев А. А.
ис	202	Борисов Б. Б.	201	Андреев А. А.
ис	241	Глухов Г. Г.	201	Андреев А. А.
ис	242	Чернов Ч. Ч.	202	Борисов Б. Б.
мм	301	Басов Б. Б.	301	Басов Б. Б.
мм	302	Сергеева С. С.	301	Басов Б. Б.
оф	401	Волков В. В.	401	Волков В. В.
оф	403	Смирнов С. С.	401	Волков В. В.
оф	402	Зайцев З. З.	401	Волков В. В.

шифр	Сотрудник. таб_номер	Сотрудник. фамилия	Сотрудник1. таб_номер	Сотрудник1. фамилия
оф	435	Лисин Л. Л.	402	Зайцев З. З.
вм	501	Кузнецов К. К.	501	Кузнецов К. К.
вм	502	Романцев Р. Р.	501	Кузнецов К. К.
вм	503	Соловьев С. С.	501	Кузнецов К. К.
эф	601	Зверев З. З.	601	Зверев З. З.
эф	602	Сорокина С. С.	601	Зверев З. З.
эф	614	Григорьев Г. Г.	602	Сорокина С. С.

Рассмотрим применение DISTINCTROW. DISTINCTROW опускает данные, основанные на целиком повторяющихся записях, а не отдельных повторяющихся столбцах. DISTINCTROW влияет на результат только в том случае, если в запрос включены *не все столбцы* из анализируемых таблиц. Предикат DISTINCTROW *игнорируется, если запрос содержит только одну таблицу или все столбцы всех таблиц* запроса.

Пример 44. Вывести список студентов, сдавших хотя бы один экзамен. По правилам соединения студенты, не сдававшие экзамены, в выборке представлены не будут.

```
SELECT Студент.рег_номер, Студент.фамилия
FROM Студент INNER JOIN Экзамен
ON Студент.рег_номер = Экзамен.рег_номер;
```

Результат выполнения запроса из примера 44

рег_номер	фамилия
10101	Николаева Н. Н.
10101	Николаева Н. Н.
10101	Николаева Н. Н.
10101	Николаева Н. Н.
10102	Иванов И. И.
10102	Иванов И. И.
20101	Андреев А. А.
20101	Андреев А. А.
20102	Федоров Ф. Ф.
30101	Бондаренко Б. Б.
30101	Бондаренко Б. Б.

рег_номер	фамилия
30102	Цветков К. К.
30102	Цветков К. К.
50101	Сергеев С. С.
50102	Кудрявцев К. К.
80101	Макаров М. М.
80101	Макаров М. М.
80101	Макаров М. М.
80102	Яковлев Я. Я.
80102	Яковлев Я. Я.

В результате будут представлены дублируемые данные о тех студентах, которые сдали более одного экзамена.

Пример 45. Чтобы избавиться от дублей строк, комбинируемых из связанных таблиц, следует применить DISTINCTROW.

```
SELECT DISTINCTROW Студент.рег_номер, Студент.фамилия
FROM Студент INNER JOIN Экзамен
ON Студент.рег_номер = Экзамен.рег_номер;
```

Результат выполнения запроса из примера 45

рег_номер	фамилия
10101	Николаева Н. Н.
10102	Иванов И. И.
20101	Андреев А. А.
30101	Бондаренко Б. Б.
30102	Цветков К. К.
80101	Макаров М. М.
80102	Яковлев Я. Я.
20102	Федоров Ф. Ф.
50101	Сергеев С. С.
50102	Кудрявцев К. К.

Рассмотрим комбинированный пример группировки строк, образованных из связанных таблиц.

Пример 46. Вывести (из таблиц «Студент» и «Экзамен») учетные номера («рег_номер»), фамилии («фамилия») студентов, а также количество сданных экзаменов (поле с именем «количество экзаменов» на основе функции COUNT (оценка)) и средний балл для каждого студента (поле с именем «средний балл» на основе функции AVG (оценка)) только для тех студентов, у которых средний балл не меньше заданного (например, 4).

```
SELECT Студент.рег_номер, фамилия, COUNT (оценка) AS [количество экзаменов], ROUND (AVG (оценка),2) AS [средний балл]
FROM Студент INNER JOIN Экзамен ON Студент.рег_номер=Экзамен.рег_номер
GROUP BY Студент.рег_номер, фамилия
HAVING AVG (оценка)>= [Введите граничную оценку];
```

Функция ROUND округляет значение своего первого аргумента до числа десятичных знаков, задаваемого вторым аргументом.

Результат выполнения запроса из примера 46

рег_номер	фамилия	количество экзаменов	средний балл
10101	Николаева Н. Н.	4	4,25
10102	Иванов И. И.	2	4
30102	Цветков К. К.	2	4,5
50101	Сергеев С. С.	1	5
80101	Макаров М. М.	3	4,67
80102	Яковлев Я. Я.	2	4

Задание. Вывести из таблиц «Кафедра» и «Сотрудник» шифр («шифр»), название («название») кафедр, а также среднюю зарплату и количество сотрудников с соответствующими заголовками только для тех кафедр, у которых средняя зарплата не меньше заданного значения.

В некоторых случаях может потребоваться исключение отдельных строк из групп (с помощью предложения WHERE) перед применением условия к группам в целом (с помощью предложения HAVING).

Предложение HAVING аналогично предложению WHERE, но применяется только к группам в целом (т. е. к строкам в результирующем наборе записей, представляющим группы), тогда как предложение WHERE применяется к отдельным строкам. Запрос может одновременно содержать и предложение WHERE, и предложение HAVING.

В этом случае запрос выполняется следующим образом. Сначала предложение WHERE применяется ко всем отдельным строкам в таблицах, представлениях или функциях, включенным в область схемы. *В группировке участвуют только строки, удовлетворяющие условию в предложении WHERE*. Затем предложение HAVING применяется к строкам в результирующем наборе записей, которые создаются после группировки. В запросе выводятся только те группы, которые удовлетворяют условиям в предложении HAVING. Допускается применение предложения HAVING только к столбцам, которые входят в предложение GROUP BY или в статистическую функцию.

Пример 47. Предположим, что таблицы «Студент» и «Экзамен» объединяются для создания запроса, в котором определяется количество экзаменов и средний балл для каждого студента. Требуется получить результат *только для определенных студентов данной специальности* (например, 09.03.03). Информация о специальностях размещается в таблице «Специальность». Кроме того, требуется отображать результаты только в том случае, когда *средний балл не меньше заданной величины* (например, 4). Первое условие задается с помощью предложения WHERE, в котором будут отброшены все студенты не той специальности еще до расчета групповых функций. Для второго условия требуется предложение HAVING, поскольку это условие базируется на результатах группировки и обработки данных. Инструкция SQL может выглядеть так:

```
SELECT Специальность.номер, Студент.рег_номер, фамилия, COUNT (оценка) AS [количество экзаменов], ROUND (AVG (оценка),2) AS [средний балл]
FROM Специальность INNER JOIN
(Студент INNER JOIN Экзамен ON Студент.рег_номер=Экзамен.рег_номер) ON Специальность.номер = Студент.номер
WHERE Специальность.номер = [Введите номер специальности]
GROUP BY Специальность.номер, Студент.рег_номер, фамилия
HAVING AVG (оценка)>= [Введите граничную оценку];
```

Результат выполнения запроса из примера 47

номер	рег_номер	фамилия	количество экзаменов	средний балл
09.03.03	10101	Николаева Н. Н.	4	4,25
09.03.03	10102	Иванов И. И.	2	4

В этом примере используется соединение трех таблиц: «Специальность», «Студент» и «Экзамен». Первые две предназначены для формирования комбинированных данных, подлежащих группировке. Таблица «Экзамен» используется для вычисления групповых функций.

В соответствии с порядком старшинства в первую очередь отсеиваются данные, не удовлетворяющие условию отбора строк во фразе WHERE. Затем производится группировка оставшихся строк в группы по условию, заданному во фразе GROUP BY, и фильтрация групп по условию их отбора во фразе HAVING.

Пример 48. Условие предыдущего примера дополнить требованием сортировки по алфавиту фамилий студентов. В результате получится многофункциональный запрос-выборка, в котором представлены все фразы команды SELECT.

```
SELECT Специальность.номер, Студент.рег_номер, фамилия, COUNT (оценка) AS [количество экзаменов], ROUND (AVG (оценка),2) AS [средний балл]
FROM Специальность INNER JOIN
(Студент INNER JOIN Экзамен ON Студент.рег_номер=Экзамен.рег_номер) ON Специальность.номер = Студент.номер
WHERE Специальность.номер = [Введите номер специальности]
GROUP BY Специальность.номер, Студент.рег_номер, фамилия
HAVING AVG (оценка)>= [Введите граничную оценку]
ORDER BY фамилия;
```

Результат выполнения запроса из примера 48

номер	рег_номер	фамилия	количество экзаменов	средний балл
09.03.03	10102	Иванов И. И.	2	4
09.03.03	10101	Николаева Н. Н.	4	4,25

Фраза ORDER BY применяется в последнюю очередь. Результат предыдущего примера упорядочивается в соответствии со спецификацией условия сортировки.

Запросы с подзапросами

Подзапрос представляет собой вложенный запрос или SELECT-FROM-WHERE предложение, которое вложено в такое же SELECT-FROM-WHERE предложение. Допускается произвольная глубина вложений. В выборке с подзапросом различают объемлющий запрос и вложенный запрос — подзапрос. Существуют следующие типы подзапросов [3].

По способу получения результата различают подзапросы:

- простой;
- коррелированный.

По виду результата различают подзапросы:

- скалярный (возвращает одно единственное значение, выбираемое на пересечении одного столбца с одной строкой);
- строковый (возвращает множество значений нескольких столбцов таблицы, но в виде одной строки);
- табличный (возвращает множество значений из одного или нескольких столбцов таблицы, размещенные более чем в одной строке).

К подзапросам применяются следующие правила [3].

1. В подзапросе нельзя использовать фразу ORDER BY.
2. Список в предложении SELECT подзапроса должен состоять из имен столбцов или из составленных из них выражений.
3. По умолчанию имена столбцов в подзапросе относятся к таблице, имя которой указано в его предложении FROM. Однако допускается ссылаться и на столбцы таблицы объемлющего запроса, используя составные имена.
4. Если подзапрос является операндом операции сравнения, то запрос должен указываться в правой части этой операции.

Простые подзапросы

Простые подзапросы характеризуются тем, что они формально никак не связаны с содержащими их внешними запросами. В случае запроса с простым подзапросом сначала выполняется полностью подзапрос, а затем его результат используется в объемлющем запросе.

Подчиненный запрос можно использовать вместо выражения в списке полей инструкции SELECT или в предложениях WHERE и HAVING, а также FROM.

Скалярный подзапрос

Скалярный подзапрос возвращает одно единственное значение, выбираемое на пересечении одного столбца с одной строкой.

Пример 49. Вывести данные из таблицы «Сотрудник» о коллегах, работающих на той же кафедре, что и сотрудник с заданным номером (например, «таб_номер»=101).

```
SELECT Сотрудник.* FROM Сотрудник
WHERE шифр = (SELECT шифр FROM Сотрудник
WHERE таб_номер= [Введите табельный номер]);
```

Результат выполнения запроса из примера 49

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	15 000,00 р.	102

В этом примере сначала в подзапросе определяется шифр кафедры, на которой работает конкретный сотрудник, а затем в объемлющем запросе производится выборка сотрудников, работающих на той же кафедре.

Отметим, что ссылка на таблицу «Сотрудник» во вложенном подзапросе означает не то же самое, что ссылка на таблицу «Сотрудник» во внешнем запросе. В действительности, два имени «Сотрудник» обозначают различные состояния одной и той же таблицы. Чтобы этот факт стал явным, полезно использовать псевдонимы, например «Сотрудник1» и «Сотрудник2».

Пример 50. В условии предыдущего примера использовать псевдонимы одной и той же таблицы.

```
SELECT * FROM Сотрудник Сотрудник1
WHERE Сотрудник1.шифр =
   (SELECT Сотрудник2.шифр FROM Сотрудник Сотрудник2
   WHERE Сотрудник2.таб_номер= [Введите табельный номер]);
```

Здесь «Сотрудник1» и «Сотрудник2» — произвольные псевдонимы таблицы «Сотрудник», определяемые во фразе FROM и исполь-

зуемые как явные уточнители во фразах SELECT и WHERE. Напомним, что псевдонимы определены лишь в пределах одного запроса. При том же значении параметра («таб_номер») результат должен совпасть с результатом предыдущего примера.

Задание. Вывести данные из таблицы «Студент» о студентах, которые обучаются по той же специальности, что и студент с заданным номером (столбец «рег_номер»).

Пример 51. Из таблицы «Специальность» выполнить как запрос с подзапросом выборку шифра кафедры и названия специальности, по которой обучается студент с данным номером (например, 10101 из таблицы «Студент»).

```
SELECT шифр, направление FROM Специальность
WHERE номер= (SELECT номер FROM Студент
WHERE рег_номер = [Введите номер студента]);
```

Результат выполнения запроса из примера 51

шифр	направление
пи	Прикладная информатика

Замечание. SQL избыточен. Этот же результат может быть получен при помощи операции соединения. Это общее правило. Выбор способа формулировки запроса — дело вкуса автора.

Пример 52. Из таблицы «Специальность» выполнить с помощью операции соединения запрос-выборку шифра кафедры и названия специальности (направления), по которой обучается студент с данным номером (например, 10101 из таблицы «Студент»).

```
SELECT шифр, направление FROM Специальность, Студент
WHERE Студент.номер = Специальность.номер
AND рег_номер = [Введите номер студента];
```

Подзапрос может содержать другой, вложенный подзапрос. Причем допускается любая глубина вложенности.

Пример 53. Из таблицы «Кафедра» выполнить как множественный (трехуровневый) запрос с подзапросом выборку аббреви-

атуры факультета и названия кафедры, где обучается студент с заданным номером (например, 10101 из таблицы «Студент»).

```
SELECT факультет, название
FROM Кафедра
WHERE шифр=
      (SELECT шифр
      FROM Специальность
      WHERE номер=
         (SELECT номер
         FROM Студент
         WHERE рег_номер = [Введите номер студента]));
```

Результат выполнения запроса из примера 53

факультет	название
ит	Прикладная информатика

В этом примере самый глубокий подзапрос формирует номер специальности, по которой обучается конкретный студент. Полученный результат передается в подзапрос следующего уровня, где определяется шифр кафедры, на которой обучают по данной специальности. Шифр кафедры передается внешнему запросу и там используется для определения факультета и названия кафедры, где учится студент.

Задание. Из таблицы «Факультет» выполнить как множественный (трехуровневый) запрос с подзапросом выборку аббревиатуры и названия факультета, где работает сотрудник с заданным табельным номером. Задействовать три таблицы: «Факультет», «Кафедра» и «Сотрудник».

Условие поиска единственного значения может быть задано не так просто, как в предыдущих примерах, а весьма сложным логическим выражением, включающим функции агрегации, логические операторы и т. д.

Пример 54. Вывести данные из таблицы «Сотрудник» о сотрудниках, чья зарплата больше, чем средняя зарплата всех сотрудников.

```
SELECT * FROM Сотрудник
WHERE зарплата> (SELECT AVG (зарплата) FROM Сотрудник);
```

Результат выполнения запроса из примера 54

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601

Табличный подзапрос с IN

Если подзапрос порождает множество значений, то в предложении WHERE объемлющего запроса могут быть использованы спецификаторы IN, ANY и ALL.

Оператор IN проверяет анализируемый столбец в данной строке на предмет вхождения его значения в список «разрешенных» значений. Список «разрешенных» значений может быть сформирован в подзапросе.

Пример 55. Вывести данные из таблицы «Сотрудник» о сотрудниках, работающих на кафедрах заданного факультета (например, «ит»).

```
SELECT *
FROM Сотрудник
WHERE шифр IN (SELECT шифр FROM Кафедра
                    WHERE факультет= [Введите аббревиатуру
                    факультета]);
```

Результат выполнения запроса из примера 55

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	15 000,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	20 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	15 000,00 р.	202

В этом примере подзапрос формирует множество значений для столбца «шифр» — множество шифров кафедр заданного факультета — и передает его объемлющему запросу. Внешний запрос формирует список сотрудников, которые работают на кафедрах, чьи шифры входят в список, сформированный подзапросом.

Замечание. Запрос с подзапросом IN в общем случае может быть заменен равносильным ему запросом на основе операции соединения. Это общее правило.

Пример 56. Предыдущий запрос «Вывести данные о сотрудниках, работающих на кафедрах заданного факультета» реализовать при помощи операции соединения.

```
SELECT Сотрудник.*
FROM Сотрудник, Кафедра
WHERE Кафедра.шифр = Сотрудник.шифр
AND факультет= [Введите аббревиатуру факультета];
```

Этот запрос при тех же исходных данных даст результат, аналогичный результату предыдущего примера.

Пример 57. Вывести данные из таблицы «Студент» о студентах, обучающихся по специальностям заданного факультета (например, «ит»). Запрос оформить как запрос с трехуровневым запросом с IN к таблицам «Кафедра» для выборки «шифров» кафедр заданного факультета и «Специальность» для выборки «номеров» специальностей.

```
SELECT * FROM Студент
WHERE номер IN
(SELECT номер FROM Специальность
WHERE шифр IN
(SELECT шифр FROM Кафедра
WHERE факультет = [Введите аббревиатуру факультета]));
```

Результат выполнения запроса из примера 57

рег_номер	номер	фамилия
10101	09.03.03	Николаева Н. Н.
10102	09.03.03	Иванов И. И.
20101	09.03.02	Андреев А. А.

рег_номер	номер	фамилия
80101	38.03.05	Макаров М. М.
80102	38.03.05	Яковлев Я. Я.
20102	09.03.02	Федоров Ф. Ф.
10103	09.03.03	Крюков К. К.

В этом примере самый глубокий подзапрос формирует список шифров кафедр заданного факультета и передает его подзапросу следующего уровня. Подзапрос второго уровня формирует список номеров специальностей, по которым обучаются студенты кафедр данного факультета. Список номеров специальностей передается внешнему запросу и там используется для выборки студентов, обучающихся по этим специальностям.

Задание. Вывести все данные из таблицы «Специальность» о специальностях, студенты которых изучают и сдают экзамены по заданной дисциплине. Запрос оформить как запрос с трехуровневым запросом с IN к таблицам «Экзамен» для выборки «рег_номеров» студентов, которые сдавали экзамены по дисциплине, заданной ее «кодом», и к таблице «Студент» для выборки «номеров» специальностей, по которым обучаются студенты, сдававшие экзамены по данной дисциплине.

Подзапрос с ALL

Если в условии отбора сравнение должно иметь значение «истина» для *каждого* (всех) значения, вырабатываемого подзапросом, то в предложении WHERE нужно использовать предикат ALL.

Подзапрос с ALL в общем случае имеет следующую структуру:

```
SELECT A FROM T1
WHERE A <оператор сравнения>
ALL (SELECT B FROM T2);
```

Запрос возвращает список тех значений столбца A, для которых оператор сравнения истинен для *всех* значений столбца B.

Пример 58. Вывести данные из таблицы «Сотрудник» о сотрудниках, чья зарплата меньше, чем у всех сотрудников кафедры, заданной ее шифром (например, «вм»).

```
SELECT *
FROM Сотрудник
WHERE зарплата< ALL
(SELECT зарплата FROM Сотрудник
WHERE шифр= [Введите шифр кафедры]);
```

Результат выполнения запроса из примера 58

таб_номер	шифр	фамилия	должность	зарплата	шеф
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

В этом примере подзапрос формирует список зарплат сотрудников данной кафедры. Объемлющий запрос формирует список сотрудников, чья зарплата меньше зарплаты *всех* сотрудников, в том числе самого низкооплачиваемого, данной кафедры. Действует принцип «меньше меньшего».

В этом варианте запроса наименьшая зарплата сотрудников кафедры «Высшая математика» («вм») составляет 25 000 р. Поэтому в результирующей таблице мы увидим список сотрудников, чья зарплата еще меньше. Если на других кафедрах есть самые низкооплачиваемые сотрудники, результирующая таблица будет пустой.

Задание. Вывести список дисциплин из таблицы «Дисциплина», чьи «объемы» больше, чем у всех дисциплин, читаемых кафедрой, заданной ее шифром.

Подзапрос с ANY

Если в условии отбора сравнение должно иметь значение «истина» хотя бы для одного *любого* из значений, вырабатываемых подзапросом, то в предложении WHERE нужно использовать предикат ANY.

Подзапрос с ANY в общем случае имеет следующую структуру:

```
SELECT A FROM T1
WHERE A <оператор сравнения>
ANY (SELECT B FROM T2);
```

Запрос возвращает список тех значений столбца A, для которых оператор сравнения истинен для *любого* значения столбца B.

Пример 59. Вывести данные из таблицы «Сотрудник» о сотрудниках, чья зарплата меньше, чем у *любого* сотрудника кафедры, заданной ее шифром (например, «вм»).

```
SELECT *
FROM Сотрудник
WHERE зарплата< ANY
(SELECT зарплата FROM Сотрудник
WHERE шифр= [Введите шифр кафедры]);
```

Результат выполнения запроса из примера 59

таб_номер	шифр	фамилия	должность	зарплата	шеф
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

В этом примере подзапрос формирует список зарплат сотрудников данной кафедры. Объемлющий запрос формирует список сотрудников, чья зарплата меньше зарплаты *любого*, в том числе самого высокооплачиваемого, сотрудника данной кафедры. Действует принцип «меньше большего».

В этом варианте запроса самая высокая зарплата у заведующего кафедрой «Высшая математика» («вм») и составляет 35 000 р. Поэтому в результирующей таблице мы видим список сотрудников, чья зарплата меньше. На других кафедрах есть сотрудники, чья зарпла-

та достаточно высока (например, заведующие кафедрами), так что результирующая таблица не будет пустой.

Задание. Вывести список дисциплин из таблицы «Дисциплина», чьи «объемы» больше, чем у любой дисциплины, читаемой кафедрой, заданной ее шифром.

Подзапрос в HAVING

Подзапрос может использоваться не только во фразе WHERE, но и во фразе HAVING как условие отбора групп.

Пример 60. Вывести список студентов, чей средний балл ниже или равен среднему баллу, полученному студентом, заданным его регистрационным номером (например, «рег_номер»=10102).

```
SELECT Студент.рег_номер, фамилия,
COUNT (оценка) AS [количество экзаменов],
ROUND (AVG (оценка),2) AS [средний балл]
FROM Студент INNER JOIN Экзамен
ON Студент.рег_номер=Экзамен.рег_номер
GROUP BY Студент.рег_номер, фамилия
HAVING AVG (оценка)<=
(SELECT AVG (оценка)
FROM Студент INNER JOIN Экзамен
ON Студент.рег_номер=Экзамен.рег_номер
WHERE Студент.рег_номер= [Введите номер студента]);
```

Результат выполнения запроса из примера 60

рег_номер	фамилия	количество экзаменов	средний балл
10102	Иванов И. И.	2	4
20101	Андреев А. А.	2	3,5
20102	Федоров Ф. Ф.	1	3
30101	Бондаренко Б. Б.	2	3,5
50102	Кудрявцев К. К.	1	3
80102	Яковлев Я. Я.	2	4

Коррелированные подзапросы

Рассмотренные в предыдущем разделе примеры представляют собой простые подзапросы. В случае выборки с простым подзапросом вложенный подзапрос выполняется *в первую очередь* и *независимо* от объемлющего запроса. Значение, которое вырабатывает простой подзапрос, *является общим для всех переменных-кортежей* во внешнем запросе. Результат, который вырабатывается простым подзапросом, используется просто как операнд условия отбора предложения WHERE внешнего запроса.

Особую разновидность подзапросов составляют *коррелированные подзапросы*. В случае коррелированного подзапроса сначала внешний запрос вырабатывает какое-то значение переменной-кортежа, а затем внутренний подзапрос для этого отдельного значения вырабатывает свой результат. Таким образом, в коррелированном подзапросе внутренний подзапрос вырабатывает результат (каждый раз свой) в зависимости от значения переменной-кортежа, передаваемого ему внешним запросом.

Пример 61. Вывести все данные о сотрудниках с максимальной зарплатой на каждой кафедре.

```
SELECT Сотрудник.*
FROM Сотрудник
WHERE зарплата=
(SELECT Max (зарплата)
FROM Сотрудник Сотрудник1
WHERE Сотрудник.шифр=Сотрудник1.шифр);
```

Результат выполнения запроса из примера 61

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601

В данном примере в запросе и подзапросе используется одна и та же таблица «Сотрудник». Для того чтобы сделать более ясной связь коррелированного подзапроса с внешним запросом, в данном примере наряду с самой таблицей «Сотрудник» во внешнем запросе используется ее псевдоним «Сотрудник1» в подзапросе.

Внешний запрос передает в подзапрос каждый раз одно значение атрибута «шифр» (название кафедры) по связи Сотрудник. шифр=Сотрудник1.шифр.

Внутренний подзапрос определяет максимальную зарплату MAX (зарплата) для группы сотрудников данной кафедры.

Внешний запрос сравнивает значение «зарплата» текущей строки таблицы с максимальным значением и в случае совпадения выводит строку в выборку.

Весьма распространенным случаем использования коррелированных подзапросов является запрос данных из таблиц, связанных ассоциативной связью (в том числе бинарной связью «многие ко многим»). В нашем примере — это таблицы «Специальность» и «Дисциплина», связанные ассоциативной таблицей «Заявка».

Пример 62. Вывести данные о специальностях (таблица «Специальность»), для которых читается дисциплина, заданная ее кодом (например, информатика с «кодом»=102). Запрос оформить как запрос с коррелированным подзапросом по таблицам «Специальность» и «Заявка».

```
SELECT *
FROM Специальность
WHERE [Введите код дисциплины] IN
(SELECT код
FROM Заявка
WHERE Специальность.номер=Заявка.номер);
```

Результат выполнения запроса из примера 62

номер	направление	шифр
09.03.03	Прикладная информатика	пи
09.03.02	Информационные системы и технологии	ис
14.03.02	Ядерные физика и технологии	эф

В этом примере внешний запрос из каждой строки таблицы «Специальность» передает в подзапрос каждый раз одно значение столбца «номер» (номер специальности) по связи Специальность.номер=Заявка.номер.

Внутренний подзапрос формирует список кодов дисциплин, по которым для данной специальности читаются разные курсы.

Внешний запрос проверяет, есть ли в этом списке интересующая нас дисциплина, и в случае положительного результата выводит строку таблицы «Специальность» в выборку.

Задание. Вывести данные о дисциплинах (таблица «Дисциплина»), которые читаются для специальности, заданной ее «номером» (например, «Прикладная информатика» с «номером»=09.03.03). Запрос оформить как запрос с коррелированным подзапросом по таблицам «Дисциплина» и «Заявка».

Коррелированный подзапрос может использоваться не только во фразе WHERE, но и в списке предложения SELECT как вычисляемое данное.

Пример 63. Для каждого студента вывести его регистрационный номер («рег_номер»), фамилию, а также количество сданных экзаменов и средний балл по результатам сдачи. Этот пример уже рассматривался как пример запроса на группировку. Здесь предлагается другой вариант его выполнения.

```
SELECT рег_номер, фамилия,
(SELECT Count (оценка) FROM Экзамен
WHERE Студент.рег_номер=Экзамен.рег_номер) AS [количество],
(SELECT ROUND (AVG (оценка),2) AS [средний балл] FROM Экзамен
WHERE Студент.рег_номер=Экзамен.рег_номер) AS [средний балл]
FROM Студент;
```

Результат выполнения запроса из примера 63

рег_номер	фамилия	количество экзаменов	средний балл
10101	Николаева Н. Н.	4	4,25
10102	Иванов И. И.	2	4
20101	Андреев А. А.	2	3,5

рег_номер	фамилия	количество экзаменов	средний балл
30101	Бондаренко Б. Б.	2	3,5
30102	Цветков К. К.	2	4,5
30103	Петров П. П.	0	
80101	Макаров М. М.	3	4,67
80102	Яковлев Я. Я.	2	4
20102	Федоров Ф. Ф.	1	3
50101	Сергеев С. С.	1	5
50102	Кудрявцев К. К.	1	3
10103	Крюков К. К.	0	

В данном примере внешний запрос передает в подзапрос значение столбца «рег_номер» (регистрационный номер конкретного студента) по связи Студент.рег_номер=Экзамен.рег_номер.

В подзапросе по существу выполняется группировка строк таблицы «Экзамен» для очередного студента. Подзапрос определяет один раз количество сданных экзаменов, другой раз средний балл по результатам сдачи экзаменов этим студентом и передает их объемлющему запросу как вычисляемое данное в список предложения SELECT.

Внешний запрос выводит результаты подзапроса как вычисляемые данные.

Задание. Для каждой кафедры вывести ее шифр («шифр»), название, а также количество сотрудников и среднюю зарплату. Запрос оформить как запрос с коррелированным подзапросом в списке фразы SELECT.

Подзапросы с квантором EXISTS

Квантор EXISTS (существует) — понятие, заимствованное из формальной логики. В языке SQL предикат с квантором существования представляется выражением EXISTS (SELECT * FROM …).

EXISTS может принимать значение «истина» или «ложь» в зависимости от результата, вырабатываемого подзапросом. EXISTS определяет условие, которое зависит от результата подзапроса в целом, а не от отдельного значения, вырабатываемого подзапросом. Выражение EXISTS (SELECT * FROM …) считается истинным только тогда, когда результат вычисления SELECT * FROM … является непустым

множеством, т. е. когда существует какая-либо запись в таблице, указанной во фразе FROM подзапроса, которая удовлетворяет условию WHERE подзапроса.

Подзапрос с EXISTS — это коррелированный подзапрос, т. е. его значение проверяется для каждого кортежа объемлющего запроса.

Замечание. Запрос с подзапросом на основе EXISTS может быть заменен равносильным ему запросом на основе операции соединения.

Пример 64. Вывести данные о сотрудниках (таблица «Сотрудник»), работающих на данном факультете (например, «ит»).

```
SELECT * FROM Сотрудник
WHERE EXISTS
(SELECT * FROM Кафедра
WHERE Кафедра.шифр=Сотрудник.шифр
AND факультет= [Введите аббревиатуру факультета]);
```

Результат выполнения запроса из примера 64

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202

В данном примере внешний запрос передает в подзапрос значение столбца «шифр» для конкретного сотрудника (шифр кафедры) по связи Кафедра.шифр=Сотрудник.шифр.

Внутренний подзапрос определяет, работает ли очередной сотрудник на интересующем нас факультете. Если это так, то подзапрос формирует непустой результат (соответствующую строку из таблицы «Кафедра»), EXISTS принимает значение «истина» и объемлющий запрос выводит информацию о сотруднике. В противном случае этого не происходит.

Как уже отмечалось, вследствие избыточности SQL, тот же результат может быть получен при помощи запроса на основе соединения таблиц.

Пример 65. Вывести данные о сотрудниках (таблица «Сотрудник»), работающих на данном факультете (например, «ит»).

```
SELECT * FROM Сотрудник, Кафедра
WHERE Кафедра.шифр=Сотрудник.шифр
AND факультет= [Введите аббревиатуру факультета];
```

При тех же исходных данных результат запроса должен выглядеть так, как показано в таблице примера 64.

Задание. Выполнить как запрос с EXISTS вывод данных о студентах (таблица «Студент»), обучающихся по специальностям (таблица «Специальность») кафедры, заданной ее шифром (например, «пи»).

Задание. Выполнить как запрос с EXISTS вывод данных о студентах (таблица «Студент»), сдававших экзамен (таблица «Экзамен») по дисциплине, заданной ее кодом (например, по информатике с «кодом»=102).

Подзапросы в операциях манипулирования данными

Подзапросы могут использоваться не только в командах выборки данных, но и в командах манипулирования данными (удаления и обновления), там где действие команды обусловлено фразой WHERE.

Рассмотрим пример запроса на удаление с простым подзапросом.

Пример 66. Удалить результаты сдачи экзамена по информатике («код»=102) с оценкой 2 для студентов заданной специальности.

```
DELETE FROM Экзамен
WHERE код = [Введите код дисциплины]
AND оценка =2 AND рег_номер IN
(SELECT рег_номер FROM Студент
WHERE номер = [Введите номер специальности]);
```

В данном примере подзапрос формирует список регистрационных номеров студентов, обучающихся на данной специальности. Этот ре-

зультат передается в объемлющий запрос и в конъюнкции с другими условиями готовит выборку строк таблицы «Экзамен», подлежащих удалению.

Рассмотрим пример запроса на удаление с коррелированным подзапросом.

Пример 67. Удалить результаты сдачи экзаменов тех студентов, чей средний балл меньше заданного.

```
DELETE FROM Экзамен
WHERE [Введите границу среднего балла]>
(SELECT AVG (оценка)
FROM Экзамен Экзамен1
WHERE Экзамен.рег_номер= Экзамен1. рег_номер);
```

В данном примере в запросе и подзапросе используется одна и та же таблица «Экзамен». Для того чтобы отличить эти два разных ее состояния, в данном примере наряду с самой таблицей «Экзамен» во внешнем запросе используется ее псевдоним «Экзамен 1» в подзапросе.

Внешний запрос передает в подзапрос каждый раз одно значение столбца «рег_номер» (регистрационный номер конкретного студента) по связи Экзамен.рег_номер=Экзамен1.рег_номер.

Внутренний подзапрос вычисляет средний балл по результатам всех экзаменов данного студента и передает его внешнему запросу.

Внешний запрос сравнивает введенное граничное значение среднего балла со средним баллом данного студента и в случае положительного результата удаляет соответствующую строку из таблицы «Экзамен».

Рассмотрим пример запроса на обновление с простым подзапросом.

Пример 68. Перепоручить ведение дисциплины, заданной ее «кодом», для специальности, заданной ее «номером», кафедре, заданной ее шифром в качестве исполнителя («исполнитель»).

```
UPDATE Дисциплина SET исполнитель = [Введите шифр кафедры-исполнителя]
WHERE код= [Введите код дисциплины] AND заявитель IN
(SELECT Кафедра.шифр
FROM Кафедра INNER JOIN Специальность
ON Кафедра.шифр=Специальность.шифр
WHERE номер = [Введите номер специальности]);
```

В данном примере в подзапросе используется соединение таблиц «Кафедра» и «Специальность» для того, чтобы были доступны шифр выпускающей кафедры и номер специальности, а результатом является шифр выпускающей кафедры («заявитель»), которая готовит специалистов по заданной специальности. В объемлющем запросе результат подзапроса в конъюнкции с кодом дисциплины определяет условие, которому должна удовлетворять дисциплина, подлежащая переадресации на нового исполнителя.

Рассмотрим пример запроса на обновление с коррелированным подзапросом.

Пример 69. Перевести студентов, чей средний балл больше заданного значения, на специальность с заданным номером.

```
UPDATE Студент SET номер = [Введи номер специальности]
WHERE [Введите границу среднего балла]<
(SELECT AVG (оценка) FROM Экзамен
WHERE Студент.рег_номер=Экзамен.рег_номер);
```

Внешний запрос, адресованный к таблице «Студент», передает в подзапрос каждый раз одно значение столбца «рег_номер» (регистрационный номер конкретного студента) по связи Студент.рег_номер=Экзамен.рег_номер.

Внутренний подзапрос вычисляет средний балл по результатам всех экзаменов данного студента и передает его внешнему запросу.

Внешний запрос сравнивает введенное граничное значение среднего балла со средним баллом данного студента и в случае положительного результата обновляет номер специальности студента.

Теоретико-множественные операции

В языке SQL можно использовать обычные операции над множествами — объединение, пересечение и вычитание, позволяющие комбинировать результаты выполнения двух и более запросов в единую результирующую таблицу. На таблицы, которые могут комбинироваться с помощью теоретико-множественных операций, накладываются определенные ограничения, упоминавшиеся ранее. В частности, таблицы-операнды должны быть совместимы по типу, т. е. иметь

одинаковую структуру. Контроль за соблюдением этого требования возлагается на разработчика запросов.

Запрос на объединение UNION

Операция UNION создает запрос на объединение, который объединяет результаты нескольких независимых запросов или таблиц. В одной операции UNION можно объединить результаты нескольких запросов, таблиц и инструкций SELECT. Наиболее общий формат команды имеет следующий вид:

[TABLE] <запрос1> UNION [ALL] [TABLE] <запрос2>;

Здесь <запросN> — инструкция SELECT, имя сохраненного запроса или имя сохраненной таблицы, перед которым стоит зарезервированное слово TABLE.

Все запросы, включенные в операцию UNION, должны отбирать одинаковое число идентичных столбцов; при этом операнды должны быть соразмерны.

Несколькими операциями UNION можно объединить любое число предложений SELECT.

Пример 70. Вывести из таблицы «Сотрудник» номера («таб_номер»), фамилии, должности и шифры кафедр сотрудников, которые работают на заданной кафедре (например, с шифром «пи») *или/и* работают на заданной должности (например, «инженер»).

```
SELECT таб_номер, фамилия, должность, шифр FROM Сотрудник
WHERE шифр= [Введите шифр кафедры]
UNION
SELECT таб_номер, фамилия, должность, шифр FROM Сотрудник
WHERE должность= [Введите должность];
```

Результат выполнения запроса из примера 70

таб_номер	фамилия	должность	шифр
101	Прохоров П. П.	зав.кафедрой	пи
102	Семенов С. С.	преподаватель	пи
105	Петров П. П.	преподаватель	пи
153	Сидорова С. С.	инженер	пи
241	Глухов Г. Г.	инженер	ис

таб_номер	фамилия	должность	шифр
242	Чернов Ч. Ч.	инженер	ис
435	Лисин Л. Л.	инженер	оф
614	Григорьев Г. Г.	инженер	эф

В этом примере одно множество представлено сотрудниками заданной кафедры («Прикладная информатика» с шифром «пи»), а второе — сотрудниками заданной должности — «инженер». Как видно, в соответствии с определением операции объединения в выборке представлены или сотрудники заданной кафедры, или сотрудники заданной должности, или сотрудники, удовлетворяющие обоим условиям.

В качестве операндов операции объединения могут использоваться не только отдельные, но и связанные таблицы.

Пример 71. Вывести из таблицы «Сотрудник» номера («таб_номер»), фамилии и должности сотрудников, которые работают на заданной кафедре (например, с шифром «пи») *или/и* являются инженерами (таблица «Инженер») по заданной специальности (например, «программист»).

```
SELECT Сотрудник.таб_номер, фамилия, должность, шифр, специальность
FROM Сотрудник LEFT JOIN Инженер ON Инженер.таб_номер = Сотрудник.таб_номер
WHERE шифр= [Введите шифр кафедры]
UNION SELECT Сотрудник.таб_номер, фамилия, должность, шифр, специальность
FROM Сотрудник LEFT JOIN Инженер ON Инженер.таб_номер = Сотрудник.таб_номер
WHERE специальность= [Введите специальность];
```

Результат выполнения запроса из примера 71

таб_номер	фамилия	должность	шифр	специальность
101	Прохоров П. П.	зав.кафедрой	пи	
102	Семенов С. С.	преподаватель	пи	
105	Петров П. П.	преподаватель	пи	
153	Сидорова С. С.	инженер	пи	электроник

таб_номер	фамилия	должность	шифр	специальность
242	Чернов Ч. Ч.	инженер	ис	программист
614	Григорьев Г. Г.	инженер	эф	программист

Левое соединение LEFT JOIN здесь используется для того, чтобы в выборке можно было увидеть специальности сотрудников-инженеров, а также сотрудников, работающих на заданной кафедре, но не являющихся инженерами заданной специальности.

Задание. Выполнить запрос на объединение (на основе команды UNION): вывести данные о дисциплинах (таблица «Дисциплина»), для которых исполнителем является кафедра, заданная ее шифром, *или/и* о дисциплинах, объем которых превышает заданную величину.

Запрос на пересечение

Не все диалекты SQL (в том числе SQL Microsoft Jet) поддерживают непосредственно операции пересечения и вычитания. Специальной инструкции для выполнения пересечения в рассматриваемой версии SQL нет. Но операция может быть выполнена на основе других инструкций. Запрос на пересечение может быть выполнен несколькими способами. В наиболее распространенном способе используется предикат — квантор EXISTS.

Рассмотрим пример операции пересечения, определенного на множестве сотрудников данной кафедры и множестве сотрудников данной должности.

Пример 72. Вывести из таблицы «Сотрудник» номера («таб_номер»), фамилии, должности и шифры кафедр сотрудников, которые работают на заданной кафедре (например, с шифром «пи») *и* при этом работают на заданной должности (например, «инженер»).

```
SELECT таб_номер, фамилия, должность, шифр
FROM Сотрудник
WHERE шифр = [Введите шифр кафедры] AND
EXISTS (SELECT * FROM Сотрудник Сотрудник1
WHERE Сотрудник.таб_номер = Сотрудник1.таб_номер
AND должность= [Введите должность]);
```

Результат выполнения запроса из примера 72

таб_номер	фамилия	должность	шифр
153	Сидорова С. С.	инженер	пи

Подзапрос на основе EXISTS всегда является коррелированным. Если объемлющий запрос и подзапрос обращаются к одной и той же таблице (например, «Сотрудник», как в нашем случае), то для отличия этих ее разных состояний в подчиненном запросе следует использовать псевдоним таблицы (например, «Сотрудник1») для ссылки на ту же таблицу, указанную в объемлющем запросе по условию связи Сотрудник.таб_номер = Сотрудник1.таб_номер.

Как видно из вышеприведенной таблицы, в соответствии с определением операции пересечения в выборке представлены сотрудники, удовлетворяющие обоим условиям.

В качестве операндов операции пересечения могут использоваться результаты соединения таблиц.

Пример 73. Выполнить запрос на пересечение: вывести из таблицы «Сотрудник» табельные номера («таб_номер»), фамилии, должности, шифры кафедр и специальности сотрудников, которые работают на заданной кафедре (например, с шифром «пи») *и* являются инженерами (таблица «Инженер») по заданной специальности (например, «электроник»).

```
SELECT Сотрудник.таб_номер, фамилия, должность, шифр, специальность
FROM Сотрудник INNER JOIN Инженер
ON Инженер.таб_номер = Сотрудник.таб_номер
WHERE шифр= [Введите шифр кафедры] AND
    EXISTS (SELECT * FROM Инженер
            WHERE Инженер.таб_номер = Сотрудник.таб_номер
                AND специальность= [Введите специальность]);
```

Результат выполнения запроса из примера 73

таб_номер	фамилия	должность	шифр	специальность
153	Сидорова С. С.	инженер	пи	электроник

Естественное соединение INNER JOIN здесь используется для того, чтобы в выборке можно было увидеть специальности сотрудников-ин-

женеров. Сотрудников, работающих на заданной кафедре, но не являющихся инженерами заданной специальности по определению операции пересечения не может быть.

Задание. Выполнить запрос на пересечение (на основе предиката EXISTS): вывести данные о дисциплинах (таблица «Дисциплина»), для которых заявителем является кафедра, заданная ее шифром *и* при этом объем которых превышает заданную величину.

Запрос на вычитание

Специальной инструкции для выполнения разности в рассматриваемой версии SQL нет. Но операция может быть выполнена на основе других инструкций. Запрос на вычитание может быть выполнен несколькими способами. Наиболее распространенный использует предикат — квантор NOT EXISTS.

Рассмотрим пример операции вычитания, определенной на множестве сотрудников данной кафедры и множестве сотрудников данной должности.

Пример 74. Выполнить запрос на вычитание: вывести из таблицы «Сотрудник» номера («таб_номер»), фамилии, должности и шифры кафедр сотрудников, которые работают на заданной кафедре (например, с шифром «пи»), *но не* являются сотрудниками заданной должности (например, «инженер»).

```
SELECT таб_номер, фамилия, должность, шифр
FROM Сотрудник
WHERE шифр = [Введите шифр кафедры] AND
        NOT EXISTS (SELECT * FROM Сотрудник Сотрудник1
        WHERE Сотрудник.таб_номер = Сотрудник1.таб_номер
                AND должность= [Введите должность]);
```

Результат выполнения запроса из примера 74

таб_номер	фамилия	должность	шифр
101	Прохоров П. П.	зав.кафедрой	пи
102	Семенов С. С.	преподаватель	пи
105	Петров П. П.	преподаватель	пи

Замечание, сделанное по поводу использования псевдонима таблицы в подзапросе на основе EXISTS в примере операции пересечения, будет справедливо и в данном случае.

В рассмотренном примере множество сотрудников заданной кафедры является уменьшаемым, а множество сотрудников заданной должности является вычитаемым. В соответствии с определением операции вычитания в выборке представлены сотрудники, работающие на заданной кафедре, но не являющиеся инженерами по должности.

В качестве операндов вычитания могут использоваться результаты соединения таблиц.

Пример 75. Выполнить запрос на вычитание: вывести из таблицы «Сотрудник» номера («таб_номер»), фамилии, должности, шифры кафедр и специальности сотрудников-инженеров, которые работают на заданной кафедре (например, с шифром «пи»), *но не* являются инженерами (таблица «Инженер») по заданной специальности (например, «программист»).

```
SELECT Сотрудник.таб_номер, фамилия, должность, шифр, специальность
FROM Сотрудник INNER JOIN Инженер
ON Инженер.таб_номер=Сотрудник.таб_номер
WHERE шифр= [Введите шифр кафедры] AND
      NOT EXISTS (SELECT * FROM Инженер
      WHERE Инженер.таб_номер = Сотрудник.таб_номер
            AND специальность= [Введите специальность]);
```

Результат выполнения запроса из примера 75

таб_номер	фамилия	должность	шифр	специальность
153	Сидорова С. С.	инженер	пи	электроник

В рассмотренном примере множество сотрудников заданной кафедры является уменьшаемым, а множество сотрудников-инженеров заданной специальности является вычитаемым. В соответствии с определением операции вычитания в выборке представлены сотрудники, работающие на заданной кафедре, но не являющиеся инженерами по заданной специальности.

Задание. Выполнить запрос на вычитание (на основе предиката NOT EXISTS): вывести данные о дисциплинах (таблица «Дисциплина»), для которых исполнителем является кафедра, заданная ее шифром, *но* при этом объем которых *не* превышает заданную величину.

В отличие от операций объединения и пересечения операция вычитания не является коммутативной, т. е. имеет значение, какое множество является уменьшаемым, а какое вычитаемым.

Пример 76. Выполнить запрос на вычитание: вывести из таблицы «Сотрудник» номера («таб_номер»), фамилии, должности и шифры кафедр сотрудников, которые являются сотрудниками заданной должности (например, «инженер»), *но не* работают на заданной кафедре (например, с шифром «пи»).

```
SELECT таб_номер, фамилия, должность, шифр
FROM Сотрудник
WHERE должность= [Введите должность] AND
    NOT EXISTS (SELECT * FROM Сотрудник Сотрудник1
          WHERE Сотрудник.таб_номер = Сотрудник1.таб_номер
                AND шифр= [Введите шифр кафедры]);
```

Результат выполнения запроса из примера 76

таб_номер	фамилия	должность	шифр
241	Глухов Г. Г.	инженер	ис
242	Чернов Ч. Ч.	инженер	ис
435	Лисин Л. Л.	инженер	оф
614	Григорьев Г. Г.	инженер	эф

Замечание, сделанное по поводу использования псевдонима таблицы в подзапросе на основе EXISTS в примере операции пересечения, будет справедливо и в данном случае.

В рассмотренном примере множество сотрудников заданной должности является уменьшаемым, а множество сотрудников заданной кафедры является вычитаемым. В соответствии с определением операции вычитания в выборке представлены сотрудники-инженеры, не работающие на заданной кафедре.

В качестве операндов операции вычитания могут использоваться результаты соединения таблиц.

Пример 77. Выполнить запрос на вычитание: вывести из таблиц «Сотрудник» и «Инженер» номера («таб_номер»), фамилии, должности, шифры кафедр и специальности сотрудников-инженеров, которые являются инженерами по заданной специальности (например, «программист»), *но не* работают на заданной кафедре (например, с шифром «пи»).

```
SELECT Сотрудник.таб_номер, фамилия, должность, шифр, специальность
FROM Сотрудник INNER JOIN Инженер ON Инженер.таб_номер
= Сотрудник.таб_номер
WHERE специальность= [Введите специальность] AND
    NOT EXISTS (SELECT * FROM Сотрудник
        WHERE Инженер.таб_номер = Сотрудник.таб_номер AND
            шифр= [Введите шифр кафедры]);
```

Результат выполнения запроса из примера 77

таб_номер	фамилия	должность	шифр	специальность
614	Григорьев Г. Г.	инженер	эф	программист
242	Чернов Ч. Ч.	инженер	ис	программист

В рассмотренном примере множество сотрудников-инженеров заданной специальности является уменьшаемым, а множество сотрудников заданной кафедры является вычитаемым. В соответствии с определением операции вычитания в выборке представлены сотрудники-инженеры, работающие по заданной специальности, но не на заданной кафедре.

Задание. Выполнить запрос на вычитание (на основе предиката NOT EXISTS): вывести данные о дисциплинах (таблица «Дисциплина»), объем которых превышает заданную величину, *но* при этом для которых заявителем *не* является кафедра, заданная ее шифром.

Представления

Представление — это виртуальная таблица, которая сама по себе не существует, но для пользователя выглядит таким образом, как будто она существует. В противоположность этому **базовая таблица** — это реальная таблица, каждой строке которой соответствуют хранимые данные. Представление не поддерживается своими собственными физическими хранимыми данными. Вместо этого в каталоге таблиц хранится определение, оговаривающее, из каких столбцов и строк других таблиц оно должно быть сформировано при реализации SQL-предложения на получение данных из представления или на модификацию таких данных.

Для чего нужны представления? Самая важная причина — это обеспечение *логической независимости данных*. Если физическая независимость данных обеспечивает независимость от изменений физической структуры базы данных, то логическая независимость обеспечивает независимость пользовательских программ от изменения логической структуры базы данных при ее реструктуризации. Кроме того, представления дают возможность различным пользователям по-разному видеть одни и те же данные, возможно, даже в одно и то же время. Это особенно ценно при работе различных категорий пользователей с единой интегрированной базой данных. Пользователям предоставляют только интересующие их данные в наиболее удобной для них форме (окно в таблицу или в любое соединение любых таблиц).

Наконец, от определенных пользователей могут быть скрыты некоторые данные, невидимые через предложенное им представление. Таким образом, принуждение пользователя осуществлять доступ к базе данных через представления является простым, но эффективным механизмом для управления санкционированием доступа.

Представления можно создавать и удалять. Соответствующие команды относятся к подмножеству языка определения данных.

Создание представлений

Обобщенный синтаксис команды создания представления имеет следующий вид

```
CREATE VIEW <имя представления>
    [<список имен столбцов представления>]
    AS <подзапрос>
    [WITH CHECK OPTION];
```

Здесь подзапрос, следующий за AS, является определением данного представления, не исполняется, а просто сохраняется в каталоге.

Необязательная фраза WITH CHECK OPTION (с проверкой) указывает на то, что для операций INSERT и UPDATE над этим представлением должна осуществляться проверка, обеспечивающая удовлетворение WHERE фразы подзапроса.

Список имен столбцов может быть пустым, в этом случае представление наследует имена столбцов из подзапроса. Список имен столбцов должен быть обязательно определен лишь в двух случаях:

а) когда хотя бы один из столбцов подзапроса не имеет имени (создается с помощью выражения, SQL-функции или константы);

б) два или более столбцов подзапроса имеют одно и то же имя.

Если же список отсутствует, то представление наследует имена столбцов из подзапроса, при этом список столбцов представления и список элементов предложения SELECT должны совпадать по количеству, по порядку следования и по типу.

Как обычно, подзапрос не может включать ни команду UNION, ни фразу ORDER BY.

Инструкция SELECT, определяющая представление, не должна содержать каких-либо параметров.

Имя представления не должно совпадать с именем таблицы.

Если запрос, определенный инструкцией SELECT, является обновляемым, то представление также может обновляться. В противном случае оно предназначено только для чтения.

С учетом этих ограничений любая таблица, выборка которой может быть осуществлена с помощью предложения SELECT, может быть определена как представление. Рассмотрим конкретные варианты:

1. Представление может быть создано с вычислением арифметического выражения.

Пример 78. Сформировать как представление данные о годовых зарплатах сотрудников.

```
CREATE VIEW Годовая_зарплата_сотрудников ([табельный номер], фамилия, [годовая зарплата])
AS SELECT таб_номер, фамилия, зарплата*12
FROM Сотрудник;
```

Представление «Годовая_зарплата_сотрудников»

табельный номер	фамилия	годовая зарплата
101	Прохоров П. П.	420 000,00 р.
102	Семенов С. С.	300 000,00 р.
105	Петров П. П.	300 000,00 р.
153	Сидорова С. С.	198 000,00 р.
201	Андреев А. А.	420 000,00 р.
202	Борисов Б. Б.	300 000,00 р.
241	Глухов Г. Г.	264 000,00 р.
242	Чернов Ч. Ч.	198 000,00 р.
301	Басов Б. Б.	420 000,00 р.
302	Сергеева С. С.	300 000,00 р.
401	Волков В. В.	420 000,00 р.
402	Зайцев З. З.	300 000,00 р.
403	Смирнов С. С.	180 000,00 р.
435	Лисин Л. Л.	264 000,00 р.
501	Кузнецов К. К.	420 000,00 р.
502	Романцев Р. Р.	300 000,00 р.
503	Соловьев С. С.	300 000,00 р.
601	Зверев З. З.	420 000,00 р.
602	Сорокина С. С.	300 000,00 р.
614	Григорьев Г. Г.	264 000,00 р.

В этом примере столбец представления «годовая зарплата» продуцируется на основе вычисления арифметического выражения.

2. Представление может формироваться на основе функции агрегации.

Пример 79. Сформировать как представление данные о месячном фонде зарплаты сотрудников.

CREATE VIEW Месячный_фонд_зарплаты_сотрудников ([фонд зарплаты])
AS SELECT Sum (зарплата)
FROM Сотрудник;

Представление «Месячный_фонд_зарплаты_сотрудников»

фонд зарплаты
524 000,00 р.

3. Представление может формироваться как проекция базовой таблицы.

Пример 80. Сформировать как представление список шифров кафедр и должностей работающих на них сотрудников.

CREATE VIEW Должности_сотрудников (кафедра, должность)
AS SELECT DISTINCT шифр, должность
FROM Сотрудник;

Представление «Должности_сотрудников»

кафедра	должность
вм	зав.кафедрой
вм	преподаватель
ис	зав.кафедрой
ис	инженер
ис	преподаватель
мм	зав.кафедрой
мм	преподаватель
оф	зав.кафедрой
оф	инженер
оф	преподаватель
пи	зав.кафедрой
пи	инженер
пи	преподаватель
эф	зав.кафедрой
эф	инженер
эф	преподаватель

В этом примере производится выборка столбцов таблицы «Сотрудник» без дублей строк. Имена столбцов представления «Должности_сотрудников» «кафедра» и «должность» соответствуют столбцам выборки «шифр» и «должность» из базовой таблицы «Сотрудник».

Задание. Сформировать представление «Номера_студентов» как проекцию таблицы «Экзамен» на столбец «рег_номер». В списке должны быть представлены в единственном экземпляре регистрационные номера студентов, сдававших экзамены.

4. Представление может быть сформировано на основе простой операции селекции по условию.

Пример 81. Сформировать представление с именем «Инженеры_вуза» на основе таблицы «Сотрудник» как выборку сотрудников, чья должность должна быть 'инженер'.

```
CREATE VIEW Инженеры_вуза ([табельный номер], кафедра, фамилия, должность, зарплата, начальник)
AS SELECT *
FROM Сотрудник
WHERE должность='инженер';
```

Представление «Инженеры_вуза»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Задание. Сформировать представление с именем «Преподаватели_вуза» на основе таблицы «Сотрудник» как выборку сотрудников, чья должность может быть или 'преподаватель' или 'зав.кафедрой'.

5. Основанием для формирования представления может быть запрос на группировку данных.

Пример 82. Сформировать как представление с именем «Гуппировка_сотрудников» результат группировки сотрудников по кафедрам с представлением агрегатных характиристик «средняя зарплата» и «количество сотрудников». Имен, которые могли бы наследоваться от второго и третьего столбцов подзапроса, нет. Поэтому имена соответствующих столбцов представления должны быть заданы явным образом.

```
CREATE VIEW Гуппировка_сотрудников (кафедра, [средняя зарплата], [количество сотрудников])
AS SELECT шифр, AVG (зарплата), Count (зарплата)
FROM Сотрудник
GROUP BY шифр;
```

Представление «Гуппировка_сотрудников»

кафедра	средняя зарплата	количество сотрудников
вм	28 333,33 р.	3
ис	24 625,00 р.	4
мм	30 000,00 р.	2
оф	24 250,00 р.	4
пи	25 375,00 р.	4
эф	27 333,33 р.	3

В отличие от предыдущих примеров представление «Группировка_сотрудников» не является просто подмножеством базовой таблицы «Сотрудник». Оно представляет собой некоторый результат статистической обработки базовой таблицы.

Задание. Сформировать представление «Группировка_студентов_по_специальностям» со столбцами « [номер специальности]», «[количество студентов]» на основе запроса на группировку строк таблицы «Студент» по столбцу «номер» с вычислением агрегатной функции Count (рег_номер).

6. Представление может быть сформировано на основе результата соединения базовых таблиц.

Пример 83. Сформировать представление с именем «Инженеры» на основе естественного соединения базовых таблиц «Сотрудник» и «Инженер» по столбцам связи «таб_номер». Имена столбцам представления присвоить по предлагаемому в примере образцу.

CREATE VIEW Инженеры ([табельный номер], кафедра, фамилия, должность, специальность, зарплата, начальник)

AS SELECT Сотрудник.таб_номер, шифр, фамилия, должность, специальность, зарплата, шеф

FROM Сотрудник, Инженер

WHERE Сотрудник.таб_номер=Инженер.таб_номер;

Представление «Инженеры»

табельный номер	кафедра	фамилия	должность	специальность	зарплата	начальник
614	эф	Григорьев Г. Г.	инженер	программист	22 000,00 р.	602
153	пи	Сидорова С. С.	инженер	электроник	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	электроник	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	программист	16 500,00 р.	202
435	оф	Лисин Л. Л.	инженер	электроник	22 000,00 р.	402

Задание. Сформировать представление «Выпускающие_кафедры» на основе внутреннего соединения базовых таблиц «Кафедра» и «Специальность» по столбцу связи «шифр». Результатом соединения будут данные только о тех кафедрах, на которых открыты специальности (о выпускающих кафедрах). В списке имен столбцов представления предусмотреть столбцы «кафедра», «название», « [номер специальности]», «направление», соответствующие столбцам соединенных таблиц: «шифр», «название», «номер», «направление».

7. Представление может быть определено на основе выборки по запросу с подзапросом.

Пример 84. Сформировать представление с именем «Сотрудники_факультета» на основе запроса с простым подзапросом. В под-

запросе на основе таблицы «Кафедра» по аббревиатуре факультета (столбец «Кафедра.факультет) сформировать список шифров кафедр факультета, заданного явно по столбцу «факультет»='ит'. В объемлющем запросе произвести выборку всех данных о сотрудниках, у которых «шифры» попадают в список «шифров» кафедр данного факультета. Список имен столбцов представления задать по образцу демонстрационного примера.

```
CREATE VIEW Сотрудники_факультета ([табельный номер], [место работы], фамилия, должность, зарплата, начальник)
AS SELECT *
FROM Сотрудник
WHERE шифр IN (SELECT шифр FROM Кафедра
WHERE факультет='ит');
```

Представление «Сотрудники_факультета»

табельный номер	место работы	фамилия	должность	зарплата	начальник
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202

Задание. Сформировать представление с именем «Студенты_кафедры» на основе запроса с простым подзапросом. В подзапросе на основе таблицы «Специальность» по шифру кафедры (столбец «Специальность.шифр) сформировать список номеров специальностей кафедры, заданной явно по столбцу «шифр»= 'пи'. В объемлющем запросе произвести выборку всех данных о студентах, у которых «номера» попадают в список «номеров» специальностей данной кафедры. В списке имен представления задать столбцы: «регистрационный номер», «фамилия», «специальность».

Удаление представлений

Представление удаляется из базы данных при помощи следующей команды:

DROP VIEW <имя представления>;

В результате выполнения этой команды уничтожается и само представление, и все представления, определенные с его использованием.

Пример 85. Удалить представление «Месячный_фонд_зарплаты_сотрудников»

DROP VIEW Месячный_фонд_зарплаты_сотрудников;

Представление будет также уничтожено при удалении базовой таблицы, лежащей в его основе.

Операции над представлениями

Представления можно использовать, как и базовые таблицы, для создания производных представлений.

Пример 86. Сформировать представление с именем «Инженеры_Прикладной_информатики» как выборку виртуальной таблицы (представления) «Инженеры_вуза», удовлетворяющую условию «кафедра»='пи'.

```
CREATE VIEW Инженеры_Прикладной_информатики
AS SELECT *
FROM Инженеры_вуза
WHERE кафедра='пи';
```

Представление «Инженеры_Прикладной_информатики»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102

Задание. Сформировать представление с именем «Преподаватели_Прикладной_информатики» как выборку виртуальной табли-

цы (представления) «Преподаватели_вуза», удовлетворяющую условию «кафедра»='пи'.

На основе представлений, также как и базовых таблиц, можно создавать запросы-выборки. При этом операции выборки данных над представлением преобразуются в эквивалентные операции над лежащей в его основе базовой таблицей. Общий принцип формирования такого рода запросов состоит в том, чтобы SELECT-предложение такого обобщенного запроса было бы допустимым правилами языка SQL предложением.

Для иллюстрации процесса слияния запроса на основе представления с определяющим его запросом рассмотрим следующий пример.

Пример 87. Из таблицы-представления «Инженеры_вуза» сформировать список инженеров, работающих на кафедре «Информационные системы» по условию «кафедра»='ис'.

```
SELECT *
FROM Инженеры_вуза
WHERE кафедра='ис';
```

Результат запроса-выборки на основе представления «Инженеры_вуза»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	16 500,00 р.	202

Запрос к представлению выполняется следующим образом.

1. Имена столбцов, указанные в списке предложения SELECT запроса, транслируются в соответствующие им имена столбцов определяющего запроса. В результате предложение SELECT приобретает следующий вид:

```
SELECT таб_номер, AS [табельный номер], шифр AS кафедра, фамилия, должность, зарплата, шеф
```

2. Имя представления, указанное в предложении FROM запроса, заменяется соответствующим именем базовой таблицы из предложения FROM определяющего запроса.

```
FROM Сотрудник
```

3. Предложение WHERE исходного запроса объединяется с предложением WHERE из определяющего запроса

```
WHERE кафедра='ис' AND должность='инженер'
```

4. В результате всех перечисленных операций объединенный запрос приобретет следующий вид:

SELECT таб_номер AS [табельный номер], шифр AS кафедра, фамилия, должность, зарплата, шеф

FROM Сотрудник

WHERE шифр='ис' AND должность='инженер';

Объединенный запрос удовлетворяет правилам SQL. Результат его выполнения должен совпасть с результатом, приведенным в таблице примера 87.

Задание. Сформировать список заведующих кафедрами как выборку из таблицы-представления «Преподавтели_вуза», удовлетворяющую условию «должность»='зав.кафедрой'.

Обновление представлений

Все обновления, которые выполняются в базовой таблице при помощи команд INSERT, DELETE, UPDATE, немедленно отображаются во всех представлениях, обращающихся к этой таблице. Однако обратное не верно, так как не все представления являются обновляемыми.

Пример 88. Выполнить обновление базовой таблицы «Сотрудник», перевести сотрудника с табельным номером «таб_номер»='242' (инженера Чернова Ч. Ч.) на кафедру «Прикладная информатика» («шифр»='пи') в той же должности.

UPDATE Сотрудник SET шифр ='пи'

WHERE таб_номер='242';

Пример 89. Проверить, как сказалось обновление таблицы «Сотрудник» на представлении «Инженеры_вуза».

По результату сравнения базовой таблицы «Сотрудник» и определенного на ней представления «Инженеры_вуза» видно, что изменение базовой таблицы транслируются в соответствующие изменения представления.

Результат обновления базовой таблицы «Сотрудник»

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	**пи**	**Чернов Ч. Ч.**	**инженер**	**16 500,00 р.**	**202**
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Результат обновления представления «Инженеры_вуза»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	**пи**	**Чернов Ч. Ч.**	**инженер**	**16 500,00 р.**	**202**
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

К представлениям также применимы команды INSERT, DELETE, UPDATE. Однако далеко не все представления обновляемы. Обновляемым считается представление, для которого имеется возможность однозначно отобразить его строку или столбец на соответствующую строку или столбец его базовой таблицы [3].

В соответствии со стандартом SQL обновляемыми являются представления, которые удовлетворяют следующим условиям [3]:

- каждый элемент в списке предложения SELECT определяющего запроса представляет собой имя столбца, а не константу, выражение или агрегатную функцию;
- в определении представления не используется спецификатор DISTINCT, т. е. из результата определяющего запроса не исключаются дубли строк;
- определяющий запрос не должен содержать предложений GROUP BY и HAVING;
- в предложении FROM определяющего запроса указана только одна таблица. Данное требование исключает возможность обновления любых представлений, построенных на основе соединения, объединения, пересечения или вычитания таблиц;
- предложение WHERE не должно включать никаких подзапросов, которые ссылались бы на таблицу, указанную в предложении FROM.

Представления являются виртуальными таблицами. Их обновление по существу сводится к обновлению лежащих в их основе базовых таблиц. Поэтому следует признать обновляемым такое представление, по спецификации которого можно однозначно определить и обновить его базовую таблицу. Из множества рассмотренных нами примеров этому требованию удовлетворяет только представление «Инженеры_вуза», полученное на основе выборки по условию из единственной таблицы.

Рассмотрим примеры операций добавления, обновления и удаления на примере этого представления. Так как это представление является обновляемым, то выполнение операций INSERT, UPDATE, DELETE не должно нарушать целостности данных, установленных для его базовой таблицы.

Пример 90. Вставить новую строку в таблицу-представление «Инженеры_вуза», например строку ('000', 'пи', 'Одинцов О. О.', 'инженер', 10000, '101'). Проверить и убедиться, что фактически эта строка вставляется в базовую таблицу «Сотрудник», лежащую в основе представления.

```
INSERT INTO Инженеры_вуза ([табельный номер], кафедра, фамилия, должность, зарплата, начальник)
VALUES ('000', 'пи', 'Одинцов О. О.', 'инженер', 10000, '101');
```

Результат вставки строки в представление «Инженеры_вуза»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	пи	Чернов Ч. Ч.	инженер	16 500,00 р.	202
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602
000	**пи**	**Одинцов О. О.**	**инженер**	**10 000,00 р.**	**101**

Результат обновления базовой таблицы «Сотрудник» после вставки

таб_номер	шифр	фамилия	должность	зарплата	шеф
000	**пи**	**Одинцов О. О.**	**инженер**	**10 000,00 р.**	**101**
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	пи	Чернов Ч. Ч.	инженер	16 500,00 р.	202
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Задание. Вставить новую строку в таблицу-представление «Преподавтели_вуза», например строку ('000', 'пи', 'Одинцов О. О.', 'преподаватель', 18000, '101'). Проверить и убедиться, что фактически эта строка вставляется в базовую таблицу «Сотрудник», лежащую в основе представления.

Пример 91. В таблице-представлении «Инженеры_вуза» обновить столбец «зарплата» (присвоить значение 20000) в новой строке, задав ее по номеру сотрудника («табельный номер»='000'). Проверить и убедиться, что и в этом случае фактически обновляется соответствующий столбец базовой таблицы «Сотрудник».

```
UPDATE Инженеры_вуза SET зарплата = [Введите новую зарплату]
WHERE [табельный номер]= [Введите табельный номер сотрудника];
```

Результат обновления строки в представление «Инженеры_вуза»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	пи	Чернов Ч. Ч.	инженер	16 500,00 р.	202
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602
000	пи	Одинцов О. О.	инженер	**20 000,00 р.**	101

Результат обновления базовой таблицы «Сотрудник»

таб_номер	шифр	фамилия	должность	зарплата	шеф
000	пи	Одинцов О. О.	инженер	**20 000,00 р.**	101
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	пи	Чернов Ч. Ч.	инженер	16 500,00 р.	202
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501

таб_номер	шифр	фамилия	должность	зарплата	шеф
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Задание. В таблице-представлении «Преподавтели_вуза» обновить столбец «зарплата» (присвоить значение 20000) в новой строке, задав ее по номеру сотрудника («табельный номер»= '000'). Проверить и убедиться, что и в этом случае фактически обновляется соответствующий столбец базовой таблицы «Сотрудник».

Пример 92. Из таблицы-представления «Инженеры_вуза» удалить вновь введенную строку, задав ее по номеру сотрудника («табельный номер»='000'). Проверить и убедиться, что и в этом случае фактически удаляется строка из базовой таблицы «Сотрудник», лежащей в основе представления.

```
DELETE *
FROM Инженеры_вуза
WHERE [табельный номер]= [Введите табельный номер сотрудника];
```

Результат удаления строки в представление «Инженеры_вуза»

табельный номер	кафедра	фамилия	должность	зарплата	начальник
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	пи	Чернов Ч. Ч.	инженер	16 500,00 р.	202
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Результат обновления базовой таблицы «Сотрудник» после удаления

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101

таб_номер	шифр	фамилия	должность	зарплата	шеф
153	пи	Сидорова С. С.	инженер	16 500,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	22 000,00 р.	201
242	пи	Чернов Ч. Ч.	инженер	16 500,00 р.	202
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	22 000,00 р.	402
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	22 000,00 р.	602

Задание. Из таблицы-представления «Преподаватели_вуза» удалить новую строку, задав ее по номеру сотрудника («табельный номер»='000'). Проверить и убедиться, что и в этом случае фактически удаляется строка из базовой таблицы «Сотрудник», лежащую в основе представления.

Индексы

Индекс — это упорядоченный (буквенный или числовой) список столбцов или групп столбцов в таблице. Таблицы могут иметь большое количество строк. А так как строки не находятся в каком-нибудь определенном порядке, то их поиск по указанному значению может потребовать времени.

Скажем, что вы, например, хотите обрабатывать таблицу «Сотрудник» в такой последовательности, чтобы значения в столбце «фамилия» располагались в алфавитном порядке. Но на такую сортировку записей таблицы требуется определенное время. И чем таблица боль-

ше, тем сортировка проходит дольше. Индексы могут оказаться прекрасным средством экономии времени. Индекс — это подчиненная таблица, или таблица поддержки, которая сопровождает свою таблицу данных. Каждой строке таблицы данных соответствует определенная строка таблицы индекса. Индекс содержит ссылки на строки таблицы в заданном порядке и позволяет получить доступ к строкам в соответствующей последовательности.

Индексы могут быть простыми, т. е. задавать порядок доступа к данным по одному столбцу, или составными с порядком доступа, определяемым последовательностью столбцов в индексе. Упорядочение доступа может задаваться как по возрастанию, так и по убыванию значений индекса.

Индексы могут создаваться только для базовых таблиц, не для представлений. Однако при создании представлений могут быть использованы индексированные таблицы. Нужные индексы создаются заранее в предвидении будущих запросов. Однако решение о их использовании принимает не пользователь, а система.

Хотя индексы ускоряют доступ к актуальным данным, замедляют процедуру обновления базовой таблицы при выполнении вставки, обновления или удаления данных. Замедление связано с тем, что индексы должны отслеживать все изменения, происходящие с наполнением таблиц.

Индексы, подобно базовым таблицам, могут создаваться и уничтожаться в любое время. Соответствующие команды относятся к подмножеству языка определения данных.

Создание индекса выполняется по команде

CREATE [UNIQUE] INDEX <имя индекса> ON <имя базовой таблицы> (<имя столбца> [{ASC|DESC}] [, <имя столбца> [{ASC|DESC}], ...]) [WITH {PRIMARY | DISALLOW NULL | IGNORE NULL}]

Здесь: <имя индекса> — имя индекса;
<имя базовой таблицы> — имя таблицы, для которой создается индекс;
<имя столбца> — имена столбцов, по которым создается индекс;
[{ASC|DESC}] — необязательный параметр задает способ упорядочения:

- ASC — по возрастанию индекса (умолчание);
- DESC — по убыванию;

[UNIQUE] — необязательный параметр уникальности индекса ограничивает доступ только к тем строкам, которые имеют неповторяющиеся значения индекса;

PRIMARY — присваивает индексу статус первичного ключа;
DISALLOW NULL — запрещает значение NULL в столбцах индекса новых записей;
IGNORE NULL — запрещает включение записей, имеющих значение NULL в столбцах индекса.

Пример 93. Создать в таблице «Сотрудник» индекс по алфавиту значений столбца «фамилия».

CREATE INDEX Фамилия ON Сотрудник (фамилия);

Уничтожение индекса осуществляется по команде

DROP INDEX <имя индекса>;

Индекс, как и представление, будет также автоматически уничтожен при уничтожении его базовой таблицы.

Заключение

Рассмотренные в учебном пособии примеры и рекомендованные для самостоятельного выполнения задания были адресованы начинающему пользователю SQL. Хотелось бы надеяться, что они в достаточной степени продемонстрировали, сколь богаты и многообразны возможности этого языка. SQL — это и язык описания структуры базы данных. Кроме того, с его помощью осуществляется наполнение базы данных содержательной информацией и манипулирование хранимыми данными. А как многообразны средства спецификации запросов-выборок! Это могут быть и простые выборки данных, и настраиваемые параметрические запросы, и запросы на сортировку и группировку данных. Это могут быть запросы, адресуемые к связанным таблицам, и запросы с подзапросами.

SQL является самым популярным языком баз данных. Будучи непроцедурным языком, он прост в изучении и может использоваться как профессионалами, так и непрофессионалами. Будете ли вы разработчиком прикладных программ, администратором базы данных или просто квалифицированным пользователем, хорошее практическое знание SQL поможет вам взаимодействовать с базами данных.

Библиографический список

1. Кодд Э. Ф. Реляционная модель данных для больших совместно используемых банков данных [Электронный ресурс] /Э. Ф. Кодд; пер. с англ. М. Р. Когаловского//СУБД. 1995. № 1. Режим доступа: http://citforum.ru/database/classics/codd/. Загл. с экрана.
2. Дейт К. Дж. Введение в системы баз данных: пер с англ./ К. Дж. Дейт. 6-е изд. Киев; М.; СПб.: Изд. дом «Вильямс», 1999. 848 с.
3. Коннолли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика : [пер. с англ.]/Т. Коннолли, К. Бегг, А. Страчан. 2-е изд. М.: Изд. дом «Вильямс», 2000. 1120 с.
4. Пушников А. Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных [Электронный ресурс]: учеб. пособие/А. Ю. Пушников. Уфа: изд. Башкир. ун-та, 1999. 108 с. [Электронный ресурс]. Режим доступа: http://www.citforum.ru/database/dblearn/index.shtml. Загл. с экрана.
5. Кириллов В. В. Основы проектирования реляционных баз данных [Электронный ресурс]/В. В. Кириллов. СПб.: Изд-во С.-Петерб. гос. ин-та точной механики и оптики (техн. ун-т) Режим доступа: http://citforum.ru/database/dbguide/. Загл. с экрана.
6. Грабер М. Введение в SQL/М. Грабер. М.: Лори, 2010. 228 с.
7. Дунаев В. В. Базы данных. Язык SQL/В. В. Дунаев. СПб.: БХВ-Петербург, 2006. 288 с.
8. Форта Б. SQL за 10 минут/Б. Форта. 4-е изд. М.: Изд. дом «Вильямс», 2014. 288 с.
9. Кириллов В. В. Структуризированный язык запросов (SQL) [Электронный ресурс] : учеб. пособие/В. В. Кириллов,

Г. Ю. Громов. СПб.: Изд-во С.-Петерб. гос. ин-та точной механики и оптики (техн. ун-т). Режим доступа: http://citforum.ru/database/sql_kg/index.shtml. Загл. с экрана.

10. Кузнецов С. Д. Введение в стандарты языка баз данных SQL [Электронный ресурс]/С. Д. Кузнецов. Б. м.: Центр информационных технологий, 1998. Режим доступа: http://citforum.ru/database/sqlbook/index.shtml. Загл. с экрана.
11. Когаловский М. Р. Абстракции и модели в системах баз данных [Электронный ресурс]/М. Р. Когаловский//СУБД. 1998. № 4,5. Режим доступа: http://www.osp.ru/dbms/1998/04–05/13031594/. Загл. с экрана.
12. MSDN-библиотека. Microsoft Access SQL Reference [Электронный ресурс]. Режим доступа: http://msdn.microsoft.com/en-us/library/bb259125 (v=office.12).aspx
13. Вендров А. М. Проектирование программного обеспечения экономических информационных систем : учебник/А. М. Вендров. 2-е изд., перераб. и доп. [Электронный ресурс]. М.: Финансы и статистика, 2006. 544 с. Режим доступа: http://vernikov.ru/media/K2/item_attachments/vendorov.pdf. Загл. с экрана.

Приложение

Наполнение реляционных таблиц учебного проекта

Таблица П1

«Факультет»

аббревиатура	название
ен	Естественные науки
ит	Информационные технологии
фм	Физико-математический

Таблица П2

«Кафедра»

шифр	название	факультет
вм	Высшая математика	ен
ис	Информационные системы	ит
мм	Математическое моделирование	фм
оф	Общая физика	ен
пи	Прикладная информатика	ит
эф	Экспериментальная физика	фм

Таблица П3

«Сотрудник»

таб_номер	шифр	фамилия	должность	зарплата	шеф
101	пи	Прохоров П. П.	зав.кафедрой	35 000,00 р.	101
102	пи	Семенов С. С.	преподаватель	25 000,00 р.	101
105	пи	Петров П. П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С. С.	инженер	15 000,00 р.	102
201	ис	Андреев А. А.	зав.кафедрой	35 000,00 р.	201
202	ис	Борисов Б. Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г. Г.	инженер	20 000,00 р.	201
242	ис	Чернов Ч. Ч.	инженер	15 000,00 р.	202
301	мм	Басов Б. Б.	зав.кафедрой	35 000,00 р.	301
302	мм	Сергеева С. С.	преподаватель	25 000,00 р.	301

Окончание табл. П3

таб_номер	шифр	фамилия	должность	зарплата	шеф
401	оф	Волков В. В.	зав.кафедрой	35 000,00 р.	401
402	оф	Зайцев З. З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С. С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л. Л.	инженер	20 000,00 р.	402
501	вм	Кузнецов К. К.	зав.кафедрой	35 000,00 р.	501
502	вм	Романцев Р. Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С. С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З. З.	зав.кафедрой	35 000,00 р.	601
602	эф	Сорокина С. С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г. Г.	инженер	20 000,00 р.	602

Таблица П4

«Специальность»

номер	направление	шифр
01.03.04	Прикладная математика	мм
09.03.02	Информационные системы и технологии	ис
09.03.03	Прикладная информатика	пи
14.03.02	Ядерные физика и технологии	эф
38.03.05	Бизнес-информатика	ис

Таблица П5

«Дисциплина»

код	объем	название	исполнитель
101	320	математика	вм
102	160	информатика	пи
103	160	физика	оф
202	120	базы данных	ис
204	160	электроника	эф
205	80	программирование	пи
209	80	моделирование	мм

Таблица П6

«Заявка»

номер	код
01.03.04	101
01.03.04	205
01.03.04	209
09.03.02	101
09.03.02	102
09.03.02	103
09.03.02	202
09.03.02	205
09.03.02	209
09.03.03	101
09.03.03	102
09.03.03	103
09.03.03	202
09.03.03	205
14.03.02	101
14.03.02	102
14.03.02	103
14.03.02	204
38.03.05	101
38.03.05	103
38.03.05	202
38.03.05	209

Таблица П7

«Зав_кафедрой»

таб_номер	стаж
101	15
201	18
301	20
401	10
501	18
601	8

Таблица П8

«Инженер»

таб_номер	специальность
153	электроник
241	электроник
242	программист
435	электроник
614	программист

Таблица П9

«Преподаватель»

таб_номер	звание	степень
101	профессор	д. т.н.
102	доцент	к. ф.-м. н.
105	доцент	к. т.н.
201	профессор	д. ф.-м. н.
202	доцент	к. ф.-м. н.
301	профессор	д. т.н.
302	доцент	к. т.н.
401	профессор	д. т.н.
402	доцент	к. т.н.
403	ассистент	–
501	профессор	д. ф.-м. н.
502	профессор	д. ф.-м. н.
503	доцент	к. ф.-м. н.
601	профессор	д. ф.-м. н.

Таблица П10

«Студент»

рег_номер	номер	фамилия
10101	09.03.03	Николаева Н. Н.
10102	09.03.03	Иванов И. И.
10103	09.03.03	Крюков К. К.
20101	09.03.02	Андреев А. А.
20102	09.03.02	Федоров Ф. Ф.
30101	14.03.02	Бондаренко Б. Б.
30102	14.03.02	Цветков К. К.

Окончание табл. П10

рег_номер	номер	фамилия
30103	14.03.02	Петров П. П.
50101	01.03.04	Сергеев С. С.
50102	01.03.04	Кудрявцев К. К.
80101	38.03.05	Макаров М. М.
80102	38.03.05	Яковлев Я. Я.

Таблица П11

«Экзамен»

дата	код	рег_номер	таб_номер	аудитория	оценка
05.06.2015	102	10101	102	т505	4
05.06.2015	102	10102	102	т505	4
05.06.2015	202	20101	202	т506	4
05.06.2015	202	20102	202	т506	3
07.06.2015	102	30101	105	ф419	3
07.06.2015	102	30102	101	т506	4
07.06.2015	102	80101	102	м425	5
09.06.2015	205	80102	402	м424	4
09.06.2015	209	20101	302	ф333	3
10.06.2015	101	10101	501	т506	4
10.06.2015	101	10102	501	т506	4
10.06.2015	204	30102	601	ф349	5
10.06.2015	209	80101	301	э105	5
10.06.2015	209	80102	301	э105	4
12.06.2015	101	80101	502	с324	4
15.06.2015	101	30101	503	ф417	4
15.06.2015	101	50101	501	ф201	5
15.06.2015	101	50102	501	ф201	3
15.06.2015	103	10101	403	ф414	4
17.06.2015	102	10101	102	т505	5

Оглавление

Учебное издание

Кара-Ушанов Владимир Юрьевич

SQL — язык реляционных баз данных

Редактор И. В. Коршунова
Верстка О. П. Игнатьевой

Подписано в печать 02.02.2016. Формат 70×100/16.
Бумага писчая. Плоская печать. Гарнитура Charter.
Уч.-изд. л. 7,58. Усл. печ. л. 12,6. Тираж 50 экз.
Заказ 9

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5.
Тел.: 8(343)375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620075, Екатеринбург, ул. Тургенева, 4.
Тел.: 8(343) 350-56-64, 350-90-13. Факс: 8(343) 358-93-06
E-mail: press-urfu@mail.ru


ЛИДЕР УНИВЕРСИТЕТСКОГО КНИГОИЗДАНИЯ
УНИВЕРСИТЕТСКАЯ КНИГА
2015


9 785799 616229