ИНФОРМАЦИОННЫЕ и КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

СУБД:
ЯЗЫК SQL
в примерах и задачах

Допущено Министерством образования и науки Российской Федерации в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки и специальности «Прикладная математика и информатика»

МОСКВА
ФИЗМАТЛИТ®
2009

УДК 681.066
ББК 22.18
С89

Астахова И. Ф., Мельников В. М., Толстобров А. П., Фертиков В. В. **СУБД: язык SQL в примерах и задачах.** — М.: ФИЗМАТЛИТ, 2009. — 168 с. — ISBN 978-5-9221-0816-4.

Учебное пособие содержит подборку примеров и упражнений различной степени сложности для практических занятий по изучению основ языка SQL в рамках учебного курса, посвященного информационным системам с базами данных.

Допущено Министерством образования и науки Российской Федерации в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки и по специальности «Прикладная математика и информатика».

ISBN 978-5-9221-0816-4

ОГЛАВЛЕНИЕ

Введение

Информационные системы, использующие базы данных, в настоящее время представляют собой одну из важнейших областей современных компьютерных технологий. С этой сферой связана большая часть современного рынка программных продуктов. Одной из общих тенденций в развитии таких систем являются процессы интеграции и стандартизации, затрагивающие структуры данных и способы их обработки и интерпретации, системное и прикладное программное обеспечение, средства разработки взаимодействия компонентов баз данных и т. п. Современные системы управления базами данных (СУБД) основаны на реляционной модели представления данных — в большой степени благодаря простоте и четкости ее концептуальных понятий и строгому математическому обоснованию.

Неотъемлемая и важная часть любой системы, включающей базы данных, — языковые средства, предоставляющие возможность доступа к данным для получения необходимой информации и осуществления необходимых действий над содержимым данных, определения их структур, способов использования и интерпретации. Язык SQL появился в 70-е годы XX века как одно из таких средств. Его прототип был разработан фирмой IBM и известен под названием SEQUEL (Structured English QUEry Language). SQL вобрал в себя достоинства реляционной модели, в частности, достоинства лежащего в ее основе математического аппарата реляционной алгебры и реляционного исчисления, используя при этом сравнительно небольшое число операторов и относительно простой синтаксис. Благодаря своим качествам язык SQL стал — вначале де-факто, а затем и официально — утвержденным в качестве стандарта языком работы с реляционными базами данных.

Учитывая место, занимаемое языком SQL в современных информационных технологиях, его знание необходимо любому специалисту, работающему в этой области. Поэтому его практическое освоение является неотъемлемой частью учебных курсов, направленных на изучение информационных систем с базами данных. В настоящее время такие курсы входят в учебные планы ряда университетских специальностей. Несомненно, что для получения студентами устойчивых навыков владения языком SQL, соответствующий учебный курс, помимо теоретиче-

ского ознакомления с основами языка, должен обязательно содержать достаточно большой объем лабораторных занятий по его практическому использованию. Предлагаемое учебное пособие направлено в первую очередь на методическое обеспечение именно такого рода занятий. В связи с этим в нем основное внимание уделяется подбору практических примеров, задач и упражнений различной степени сложности по составлению SQL-запросов, позволяющих обеспечить проведение практических занятий по изучению языка в течение учебного семестра. При этом описание конструкций языка и тонкостей его применения в пособии приводится в минимальном объеме, необходимом для понимания студентами предлагаемых для решения упражнений и задач, и, возможно, с некоторым ущербом в строгости изложения материала.

Все приведенные в пособии задачи и упражнения составлены на примере использования одной общей базы данных, структура которой специально подобрана для обеспечения практической реализации и иллюстрации изучаемых конструкций языка. Для облегчения практической организации занятий по изучению языка с использованием предлагаемых в пособии упражнений в компьютерном классе на реальной базе данных дополнением пособия служит файл, содержащий SQL-сценарий создания и наполнения данными учебной базы данных, использованной в книге. Файл размещен на сайте издательства <http://www.fml.ru>. Первая часть сценария состоит из последовательности операторов DDL для создания таблиц, первичных и внешних ключей. Остальная часть образована командами DML, наполняющими таблицы учебной информацией. Авторы надеются, что возможные случайные совпадения с реальными данными не вызовут раздражения читателей.

Сценарий ориентирован на сервер Oracle, и для его использования с другой СУБД, по-видимому, потребуется учет специфики. Следует обратить внимание на типы полей таблиц, заданные DDL-операторами сценария, а также на использованный формат представления строк и дат в командах DML. В самом трудном случае адаптация сценария, возможно, потребует использования какого-либо текстового процессора (например, для изменения длины строк в запросах INSERT). Кроме данной преодолимой трудности, авторы не предвидят препятствий к использованию сценария: достаточно квалификации пользователя СУБД, прочитавшего нашу книгу.

Возможность составления студентами запросов к реальной базе данных с достаточно большим объемом специально подобранных данных позволяет существенно повысить продуктивность занятий, более наглядно увидеть особенности выполнения конкретных видов SQL-запросов, в частности, оценить реальное время их выполнения.

В пособии приведены ответы на большинство приведенных в нем задач, облегчающие преподавателю проверку результатов выполнения заданий и позволяющие использовать пособие для самостоятельной работы студентов. Примеры и задачи протестированы с использованием

СУБД Oracle и практически опробованы при проведении занятий в Воронежском госуниверситете на факультете компьютерных наук и факультете прикладной механики, математики и информатики.

В приложении приведены тексты дополнительных задач по проектированию баз данных. Эти задачи также могут использоваться при выполнении курсовых работ и самостоятельной работы студентов.

Авторы надеются, что пособие окажется полезным не только преподавателям и студентам, но и другим читателям, заинтересованным в получении начальных практических навыков использования языка SQL.

Авторы выражают искреннюю благодарность всем, кто помогал в создании этой книги, преподавателям ВГУ, использующим ее материалы при проведении занятий по языку SQL, за обсуждение книги и пожелания по ее содержанию. Особая благодарность Сергею Дмитриевичу Кузнецову, профессору кафедры системного программирования факультета вычислительной математики и кибернетики МГУ, который не пожалел времени на внимательное прочтение рукописи и сделал большое число ценных замечаний, позволивших значительно улучшить эту книгу.

Авторы с благодарностью примут любые замечания, пожелания, исправления, которые будут способствовать улучшению качества пособия, по адресу: 394693, Университетская пл., 1, Воронеж, Россия; электронный адрес: tap@main.vsu.ru.

Глава 1

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

1.1. Основные понятия реляционных баз данных

Основой современных систем, использующих базы данных, является *реляционная* модель данных. В этой модели данные, представляющие информацию о предметной области, организованы в виде двумерных таблиц, называемых *отношениями*. На рис. 1 приведен пример такой таблицы-отношения и поясняются основные термины реляционной модели.

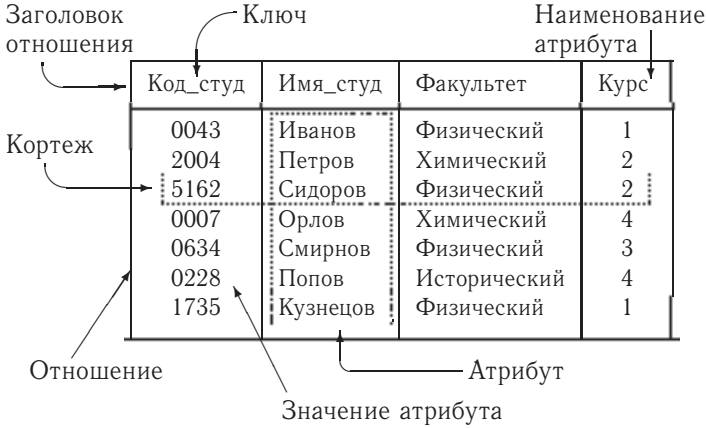

Рис. 1

- *Отношение* — это таблица, подобная приведенной на рис. 1, и состоящая из строк и столбцов. Верхняя строка таблицы-отношения называется *заголовком отношения*. Термины *отношение* и *таблица* обычно употребляются как синонимы, однако в языке SQL используется термин *таблица*.

- Строки таблицы-отношения называются *кортежами*, или *записями*. Столбцы называются *атрибутами*. Термины: атрибут, столбец, колонка, поле — обычно используются как синонимы. Каждый атрибут имеет наименование (имя), которое должно быть уникальным в конкретной таблице-отношении, однако в разных таблицах имена атрибутов могут совпадать.
- Количество кортежей в таблице-отношении называется *кардинальным числом* отношения, а количество атрибутов называется *степенью* отношения.
- *Ключ*, или *первичный ключ* отношения — это уникальный идентификатор строк (кортежей), т. е. такой атрибут (набор атрибутов), для которого в любой момент времени в отношении не существует строк с одинаковыми значениями этого атрибута (набора атрибутов). На рис. 1 таблицы ячейка с именем ключевого атрибута имеет нижнюю границу в виде двойной черты.
- *Домен* отношения — это совокупность значений, из которых могут выбираться значения конкретного атрибута, т. е. конкретный набор имеющихся в таблице значений атрибута в любой момент времени должен быть подмножеством множества значений домена, на котором определен этот атрибут. В общем случае на одном и том же домене могут быть определены значения разных атрибутов. Важным является то, что домены вводят ограничения на операции сравнения значений различных атрибутов. Эти ограничения состоят в том, что корректным образом можно сравнивать между собой только значения атрибутов, определенных на одном и том же домене.

Отношения реляционной базы данных обладают следующими свойствами:

- в отношениях не должно быть кортежей-дубликатов;
- кортежи отношений неупорядочены;
- атрибуты отношений также неупорядочены.

Из этих свойств отношения вытекают следующие важные следствия.

- Из уникальности кортежей следует, что в отношении *всегда* имеется атрибут или набор атрибутов, позволяющий *идентифицировать* кортеж; другими словами, в отношении ***всегда*** есть первичный ключ.
- Из неупорядоченности кортежей следует, во-первых, что в отношении не существует другого способа адресации кортежей, кроме адресации *по ключу*; во-вторых, что в отношении не существует таких понятий как первый кортеж, последний, предыдущий, следующий и т. п.
- Из неупорядоченности атрибутов следует, что единственным способом их адресации в запросах является использование наименования атрибута.

Относительно свойства реляционного отношения, касающегося отсутствия кортежей-дубликатов, следует сделать важное замечание. В этом пункте SQL не полностью соответствует реляционной модели. А именно, в отношениях, являющихся результатами запросов, SQL *допускает* наличие одинаковых строк. Для их устранения в запросе используется ключевое слово **DISTINCT** (см. ниже).

Информация в реляционных базах данных, как правило, хранится не в одной таблице-отношении, а в нескольких. При создании нескольких таблиц взаимосвязанной информации появляется возможность выполнения более сложных операций с данными, т. е. более сложной обработки данных. Для работы со связанными данными из нескольких таблиц важным является понятие так называемых *внешних ключей*.

Внешним ключом таблицы называется атрибут или набор атрибутов этой таблицы, каждое значение которых в текущем состоянии таблицы всегда совпадает со значением атрибутов, являющихся ключом, в другой таблице. Внешние ключи используются для связывания значений атрибутов из разных таблиц. С помощью внешних ключей обеспечивается так называемая *ссылочная целостность* базы данных, т. е. согласованность данных, описывающих одни и те же объекты, но хранящихся в разных таблицах.

1.2. Отличие SQL от процедурных языков программирования

Язык SQL относится к классу непроцедурных языков программирования. В отличие от универсальных процедурных языков, которые также могут быть использованы для работы с базами данных, язык SQL ориентирован не на *записи*, а на *множества*.

Это означает следующее. В качестве входной информации для формулируемого на языке SQL запроса к базе данных используется *множество кортежей-записей* одной или нескольких таблиц-отношений. В результате выполнения запроса также образуется *множество кортежей* результирующей таблицы-отношения. Другими словами, в SQL результатом любой операции над отношениями также является отношение. Запрос SQL задает не процедуру, т. е. последовательность действий, необходимых для получения результата, а условия, которым должны удовлетворять кортежи результирующего отношения, сформулированные в терминах входного отношения (входных отношений).

1.3. Интерактивный и встроенный SQL

Существуют и используются две формы языка SQL: *интерактивный SQL* и *встроенный SQL*.

Интерактивный SQL используется для непосредственного ввода SQL-запросов пользователем и получения результата в интерактивном режиме.

Встроенный SQL состоит из команд SQL, встроенных внутрь программ, которые обычно написаны на некотором другом языке (Паскаль, C, C++ и др.). Это делает программы, написанные на таких языках, более мощными, гибкими и эффективными, обеспечивая их применение для работы с данными, хранящимися в реляционных базах. При этом, однако, требуются дополнительные средства обеспечения интерфейса SQL с языком, в который он встраивается.

Данная книга посвящена интерактивному SQL, поэтому в ней не обсуждаются вопросы построения интерфейсов, позволяющих связать SQL с другими языками программирования.

1.4. Составные части SQL

И интерактивный, и встроенный SQL подразделяются на следующие составные части.

Язык Определения Данных — DDL (Data Definition Language): дает возможность создания, изменения и удаления различных объектов базы данных (таблиц, индексов, пользователей, привилегий и т. п.).

К числу дополнительных функций DDL могут быть отнесены средства определения ограничений целостности данных, определения порядка структур хранения данных, описания элементов физического уровня хранения данных.

Язык Обработки Данных — DML (Data Manipulation Language): предоставляет возможность выборки информации из базы данных и ее преобразования.

Тем не менее это не два различных языка, а компоненты единого SQL.

1.5. Типы данных

В языке SQL имеются средства, позволяющие для каждого атрибута указывать тип данных, которому должны соответствовать все значения этого атрибута.

Следует отметить, что определение типов данных является той частью, в которой коммерческие реализации языка не полностью согласуются с требованиями официального стандарта SQL. Это объясняется, в частности, желанием сделать SQL совместимым с другими языками программирования.

1.5.1. Тип данных “строка символов”. Тип данных **CHARACTER** или **CHAR** представляет символьные строки фиксированной длины. Его синтаксис имеет вид:

CHARACTER[(<*длина*>)] или
CHAR[(<*длина*>)].

Текстовые значения поля таблицы, для которого определен тип **CHAR**, имеют *фиксированную* длину, которая определяется параметром

<*длина*>. Этот параметр может принимать значения от 1 до 255, т. е. строка может содержать до 255 символов. Если во вводимой в поле текстовой константе фактическое число символов меньше числа, определенного параметром <*длина*>, то эта константа автоматически дополняется справа пробелами до заданного числа символов. Квадратные скобки указывают на то, что значение параметра <*длина*> может не указываться явно. В этом случае длина строки полагается равной одному символу.

Тип данных для строк переменной длины может обозначаться ключевыми словами **VARCHAR**, **CHARACTER VARYING** или **CHAR VARYING**. Он описывает текстовую строку, которая может иметь *произвольную* длину до определенного конкретной реализацией SQL максимума (в Oracle до 2000 символов). В отличие от типа **CHAR**, в этом случае при вводе текстовой константы, фактическая длина которой меньше заданной, ее дополнение пробелами до заданного максимального значения не производится.

Константы, имеющие тип **CHARACTER** или **VARCHAR**, в выражениях SQL заключаются в одиночные кавычки, например, '<*текст*>'.

Следующие предложения эквивалентны:

VARCHAR[(<*длина*>)], **CHAR VARYING**[(<*длина*>)], **CHARACTER VARYING**[(<*длина*>)].

Если длина строки не указана явно, она полагается равной одному символу во всех случаях.

По сравнению с типом **CHAR** тип данных **VARCHAR** позволяет более экономно использовать память, выделяемую для хранения текстовых значений, и оказывается более удобным при выполнении операций, связанных со сравнением текстовых констант.

1.5.2. Числовые типы данных. Стандартными числовыми типами данных SQL являются:

- **INTEGER** — используется для представления целых чисел в диапазоне от -2^{31} до $+2^{31}$;
- **SMALLINT** — используется для представления целых чисел в диапазоне меньшем, чем для **INTEGER**, а именно от -2^{15} до $+2^{15}$;
- **DECIMAL**(<*точность*>[, <*масштаб*>]) — десятичное число с фиксированной точкой; точность указывает, сколько значащих цифр имеет число. Масштаб указывает максимальное число цифр справа от точки;
- **NUMERIC**(<*точность*>[, <*масштаб*>]) — десятичное число с фиксированной точкой, такое же, как и **DECIMAL**;
- **FLOAT**[(<*точность*>)] — число с плавающей точкой и указанной минимальной точностью;
- **REAL** — такое же число, как и **FLOAT**, за исключением того, что точность устанавливается по умолчанию в зависимости от конкретной реализации SQL.

- **DOUBLE PRECISION** — такое же число, как и **REAL**, но точность в два раза превышает точность для **REAL**.

СУБД Oracle использует дополнительно тип данных **NUMBER** для представления всех числовых данных: целых, с фиксированной или плавающей точкой. Его синтаксис:

NUMBER[(<*точность*>[, <*масштаб*>])] .

Если значение параметра <*точность*> не указано явно, оно полагается равным 38. Значение параметра <*масштаб*> по умолчанию предполагается равным 0. Значение параметра <*точность*> может изменяться от 1 до 38; значение параметра <*масштаб*> может изменяться от −84 до 128. Использование отрицательных значений масштаба означает сдвиг десятичной точки в сторону старших разрядов. Например, определение **NUMBER**(7, −3) означает округление до тысяч.

Типы **DECIMAL** и **NUMERIC** полностью эквивалентны типу **NUMBER**. Синтаксис:

DECIMAL[(<*точность*>[, <*масштаб*>])],
DEC[(<*точность*>[, <*масштаб*>])],
NUMERIC[(<*точность*>[, <*масштаб*>])].

Напоминаем, что квадратные скобки указывают на необязательность заключенных в них параметров.

1.5.3. Дата и время. Представление дат и времени в SQL зависит от конкретной СУБД. В Oracle тип данных **DATE** используется для представления даты и времени. Наличие типа данных для хранения даты позволяет поддерживать специальную арифметику дат. Добавление к переменной типа **DATE** целого числа означает увеличение даты на соответствующее число дней, а вычитание соответствует определению более ранней даты.

Константы типа **DATE** записываются в зависимости от формата, принятого в конкретной системе. Например, '03.05.1999' или '12/06/1989', или '03-nov-1999', или '03-apr-99'.

1.5.4. Неопределенные или отсутствующие данные (NULL). Для обозначения отсутствующих, пропущенных или неизвестных значений атрибута в SQL используется ключевое слово **NULL**. Довольно часто можно встретить словосочетание "*атрибут имеет значение* **NULL**". Строго говоря, **NULL** не является значением в обычном понимании, а используется именно для обозначения того факта, что действительное значение атрибута на самом деле по каким-либо причинам отсутствует. Это приводит к ряду особенностей, что следует учитывать при использовании значений атрибутов, которые могут находиться в состоянии **NULL**.

- В агрегирующих функциях, позволяющих получать сводную информацию по множеству значений атрибута, например, суммарное или среднее значение, для обеспечения точности и однозначности

толкования результатов отсутствующие или **NULL**-значения атрибутов игнорируются.

- Условные операторы расширяются от булевой двузначной логики **истина/ложь** до трехзначной логики **истина/ложь/неизвестно**.
- Все операторы возвращают пустое значение (**NULL**), если значение любого из операндов отсутствует (имеет "значение **NULL**").
- Для проверки на пустое значение следует использовать операторы **IS NULL** и **IS NOT NULL** (использование для этого оператора сравнения "=" является ошибкой).
- Функции преобразования типов, имеющие **NULL** в качестве аргумента, возвращают пустое значение (**NULL**).

1.6. Используемые термины и обозначения

Ключевые слова — это используемые в выражениях SQL слова, имеющие специальное назначение (например, они могут обозначать конкретные команды SQL). Ключевые слова нельзя использовать для других целей, к примеру, в качестве имен объектов базы данных. В книге они выделяются шрифтом: **КЛЮЧЕВОЕСЛОВО**.

Команды, или *предложения*, являются инструкциями, с помощью которых SQL обращается к базе данных. Команды состоят из нескольких (одной или более) логических частей, называемых предложениями. Предложения начинаются ключевым словом и состоят из ключевых слов и аргументов.

Объекты базы данных, имеющие имена (таблицы, атрибуты и др.), в книге также выделяются особым образом: `ТАБЛИЦА1`, `АТРИБУТ_2`.

В описании синтаксиса команд SQL оператор определения "**::=**" разделяет определяемый элемент (слева от оператора) и собственно его определение (справа от оператора); квадратные скобки "[ ]" указывают *необязательный* элемент синтаксической конструкции; многоточие "…" указывает, что выражение, предшествующее ему, может повторяться любое число раз; фигурные скобки "{ }" объединяют последовательность элементов в *логическую группу*, один из элементов которой должно быть обязательно использован; вертикальная черта "|" указывает, что часть определения, следующая за этим символом, является одним из возможных вариантов; в угловые скобки "< >" заключаются элементы, которые объясняются по мере того, как вводятся.

1.7. Учебная база данных

В приводимых в пособии примерах построения SQL-запросов и контрольных упражнениях используется база данных, состоящая из следующих таблиц.