3. Введение в язык SQL

Границы моего языка означают границы моего мира.

Людвиг Витгенштейн,
англо-австрийский философ, логик

SQL (Structured Query Language) — это структурированный язык запросов к реляционным базам данных. Базы данных, основанные на реляционной модели данных, являются в настоящее время наиболее широко распространёнными вследствие своей простоты и универсальности методов обработки данных.

Существуют три стандарта SQL [1], принятые ISO (International Standards Organization, Международной организацией по стандартизации): SQL-89 (SQL-1), SQL-92 (SQL-2) и SQL-99/2003 (SQL-3).

Большинство коммерческих систем управления базами данных поддерживают стандарт SQL-92, который принят ISO в качестве международного стандарта. Многие версии (диалекты) имеют свои отличия, касающиеся в основном синтаксиса и некоторых дополнительных возможностей.

SQL является декларативным языком, основанным на операциях реляционной алгебры.

3.1. Операции реляционной алгебры

Операции реляционной алгебры (РА) применимы к реляционным отношениям (таблицам). Результатом выполнения операции реляционной алгебры также является отношение, построенное на основе одного или нескольких исходных отношений. Существует пять основных и три вспомогательных операции РА, которые могут быть выражены через основные.

3.1.1. Основные операции реляционной алгебры

1. **Проекция** (project).

Это унарная операция (выполняемая над одним отношением) для выбора подмножества атрибутов из отношения R. Она уменьшает арность отношения и может уменьшить мощность отношения за счёт исключения одинаковых кортежей.

Пример 1

Пусть имеется отношение $R(A, B, C)$ (рис. 3.1, *а*).

Тогда проекция $\pi_{A,C}(R)$ будет такой, как показано на рис. 3.1, *б*.

Отношение R

A	B	C
a	b	c
c	a	d
c	b	d

а)

Проекция $\pi_{A,C}$(R)

A	C
a	c
c	d

б)

Рис. 3.1. Пример проекции отношения

2. **Селекция** (select).

Это унарная операция, результатом которой является подмножество кортежей исходного отношения, соответствующих условиям, накладываемым на значения определённых атрибутов. Это одна из наиболее часто используемых операций SQL.

Пример 2

Для отношения $R(A, B, C)$ (рис. 3.2, *а*) селекция $\sigma_{C=d}(R)$ (при условии «значение атрибута C равно d») будет такой (рис. 3.2, *б*):

Отношение R

A	B	C
a	v	c
c	a	d
c	b	d

а)

Селекция $\sigma_{C=d}$(R)

A	B	C
c	a	d
c	b	d

б)

Рис. 3.2. Пример селекции отношения

3. **Декартово произведение** (cartesian product).

Это бинарная операция над разносхемными отношениями, соответствующая определению декартова произведения для РМД.

Пример 3

Пусть имеются отношение $R(A, B)$ и отношение $S(C,D,E)$ (рис. 3.3, *а*). Тогда декартово произведение $R \times S$ будет таким (рис. 3.3, *б*):

Отношение R

A	B
1	4
2	5
3	6

Отношение S

C	D	E
g	h	a
a	b	c

а)

Декартово произведение R × S

A	B	C	D	E
1	4	g	h	a
1	4	a	b	c
2	5	g	h	a
2	5	a	b	c
3	6	g	h	a
3	6	a	b	c

б)

Рис. 3.3. Пример декартова произведения отношений

4. **Объединение** (union).

   Объединением двух односхемных отношений R и S называется отношение $T = R \cup S$, которое включает все кортежи обоих отношений без повторов.

ПРИМЕЧАНИЕ

Имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей и типы данных соответствующих полей.

5. **Разность** (except).

   Разностью односхемных отношений R и S называется множество кортежей R, не входящих в S.

   Пример 4

   Пусть имеются отношение $R(A, B, C)$ и отношение $S(A, B, C)$ (рис. 3.4, *а*). Тогда разность $R–S$ будет такой (рис. 3.4, *б*):

Отношение R

A	B	C
a	b	c
c	a	d
c	h	c

Отношение S

A	B	C
g	h	a
a	b	c
h	d	d

а)

Разность R – S

A	B	C
c	a	d
c	h	c

б)

Рис. 3.4. Пример разности отношений

3.1.2. Вспомогательные операции реляционной алгебры

1. *Пересечение* (intersect).

   Пересечением двух односхемных отношений R и S является подмножество кортежей, принадлежащих обоим отношениям. Это можно выразить через разность:

$$R \cap S = R - (R - S).$$

2. *Соединение* (join).

Эта операция определяет подмножество декартова произведения двух разносхемных отношений. Кортеж декартова произведения входит в результирующее отношение, если для атрибутов *разных* исходных отношений выполняется некоторое *условие соединения F*. Соединение может быть выражено следующим образом:

$$R \triangleright\triangleleft_F S = \sigma_F (R \times S).$$

Если условием является равенство значений двух атрибутов исходных отношений, такая операция называется *эквисоединением*. *Естественным* называется эквисоединение по одинаковым атрибутам исходных отношений.

Пример 5

Пусть имеются отношения *R*(*A*,*B*,*C*) и *S*(*A*,*D*,*E*) (рис. 3.5, *а*). Тогда естественное соединение $R \triangleright\triangleleft S$ будет таким, как показано на рис. 3.5, *б*.

Отношение R

A	B	C
a	b	c
c	a	d
c	h	c
g	b	d

Отношение S

A	D	E
g	h	a
c	b	c
h	d	d

а)

Соединение R▷◁S

A	B	C	D	E
c	a	d	b	c
c	h	c	b	c
g	b	d	h	a

б)

Рис. 3.5. Пример естественного соединения отношений

3. *Деление* (division).

Пусть отношение *R* содержит атрибуты $\{r_1, r_2, ..., r_k, r_{k+1}, ..., r_n\}$, а отношение *S* — атрибуты $\{r_{k+1}, ..., r_n\}$. Тогда результирующее отношение содержит атрибуты $\{r_1, r_2, ..., r_k\}$. Кортеж отношения *R* включается в результирующее отношение, если его декартово произведение с отношением *S* входит в *R*. Деление может быть выражено следующим образом:

$$R / S = \pi_{r_1,...r_k}(R) - \pi_{r_1,...r_k}((\pi_{r_1,...r_k}(R) \times S) - R).$$

Пример 6

Пусть имеются отношения *R*(*A*,*B*,*C*) и *S*(*A*,*B*) (рис. 3.6, *а*). Тогда частное *R*/*S* будет таким, как показано на рис. 3.6, *б*.

Отношение R

A	B	C	D
a	b	c	b
a	b	g	h
c	f	g	h
c	f	c	b
a	v	c	b
c	v	g	h

Отношение S

C	D
c	b
g	h

а)

Частное R/S

A	B
a	b
c	f

б)

Рис. 3.6. Пример операции деления

3.2. Общие сведения о языке SQL

Язык работы с базами данных предоставляет пользователям возможность:

- создавать базу данных и таблицы с полным описанием их структуры;
- выполнять основные операции манипулирования данными (добавление, изменение, удаление данных);
- выполнять запросы, осуществляющие преобразование данных в необходимую информацию.

Создание базы данных обычно выполняется автоматически при установке самой СУБД, поэтому в данном пособии не рассматривается.

Для реализации этих функций SQL включает три группы средств:

- DDL (Data Definition Language) — язык определения данных;
- DML (Data Manipulation Language) — язык манипулирования данными;
- DCL (Data Control Language) — язык управления данными.

По стандарту ANSI подмножество команд DCL является частью DDL.

В командах SQL не различаются прописные и строчные буквы (за исключением строчных литералов). Символы и строки символов заключаются в одинарные кавычки, например, 'N', 'учебник'. Однострочные комментарии начинаются с двух дефисов (--), многострочные заключаются в символы /* и */.

Каждая команда заканчивается точкой с запятой ';'. Значения параметров команд, принятые по умолчанию, выделены подчеркиванием, например *ALL*.

ПРИМЕЧАНИЕ

Примем следующие обозначения для описания синтаксиса:

- { } — содержимое фигурных скобок рассматривается как единое целое для остальных символов;
- | — заменяет слово ИЛИ;
- [ ] — содержимое квадратных скобок является необязательным;
- < > — содержимое угловых скобок заменяется соответствующими ключевыми словами, литералами, идентификаторами или выражениями (в зависимости от контекста);
- ... — всё, что предшествует многоточию, может повторяться произвольное число раз;
- .,.. — всё, что предшествует этим символам, может повторяться произвольное число раз, каждое вхождение отделяется запятой.

В соответствии со стандартов ISO идентификатор определяется как последовательность символов длиной не более 128, начинающаяся с буквы латинского алфавита и содержащая буквы латинского алфавита, цифры и символ подчеркивания (_). В большинстве СУБД накладываются более жёсткие ограничения на длину идентификатора.

Синтаксис команд и примеры, приведённые в данном пособии, соответствуют синтаксису СУБД Oracle 9i и выше.

3.3. Создание таблиц

Создание нового пустого отношения (таблицы) выполняется с помощью команды DDL CREATE TABLE. Упрощённый синтаксис этой команды:

```
CREATE TABLE <имя таблицы>
  ( <имя поля1> <тип данных> [ (<размер>) ]
      [ [NOT] NULL] [ DEFAULT <выражение> ]
      [<ограничения целостности поля>…]
  [, <имя поля2> <тип данных> [ (<размер>) ]
      [ [NOT] NULL] [ DEFAULT <выражение> ]
      [<ограничения целостности поля>] .,..]
  [, <ограничения целостности таблицы>] );
```

Расшифровка элементов описания приведена в табл. 3.1.

Таблица 3.1. Описание элементов команды CREATE TABLE

Элемент	Описание
<имя таблицы>	Имя таблицы, обычный идентификатор. Должно быть уникальным в рамках базы данных (подсхемы базы данных)
<имя поля>	Имя поля (столбца) таблицы, обычный идентификатор
<тип данных>	Тип данных поля. Можно использовать один из следующих типов: ✦ INTEGER, INT, SMALLINT — целые числа; ✦ NUMERIC[(длина [, точность])], NUMBER[(длина [, точность])] — числа с фиксированной запятой; ✦ FLOAT, REAL — вещественные числа; ✦ CHAR[(длина)], VARCHAR(длина) — символьные строки фиксированной и переменной длины; ✦ DATE, DATETIME — дата/время; ✦ TIME — время (есть не во всех СУБД)
<размер>	Размер поля в символах (для текста и чисел)

продолжение ⇩

Таблица 3.1 *(продолжение)*

Элемент	Описание
<ограничения целостности>	✦ PRIMARY KEY — первичный ключ (обязательный, уникальный и единственный на таблицу); ✦ UNIQUE — уникальность значения поля в пределах столбца таблицы; ✦ CHECK (<условие>) — условие, которому должно удовлетворять значение поля; ✦ REFERENCES <имя таблицы> [(<имя столбца>)] — внешний ключ
<ограничения целостности таблицы>	✦ CHECK (<условие>) — условие, которому должны удовлетворять значения одного или нескольких полей; ✦ FOREIGN KEY [(<список полей>)] REFERENCES; <имя таблицы> [(<список полей>)] — внешний ключ; ✦ PRIMARY KEY (<список полей>) — первичный ключ; ✦ UNIQUE (<список полей>) — уникальное значение комбинации полей в пределах таблицы

Если размер поля не указан, то принимается значение, принятое в данной СУБД по умолчанию для указанного типа. Для всех СУБД точность для числовых типов по умолчанию равна 0, размер поля типа CHAR по умолчанию равен 1, а для типа VARCHAR размер указывать обязательно.

Для типа DATE поддерживается арифметика дат, например:

```
(<текущая дата> + 1) — завтра
(<дата1> - <дата2>) — количество дней, прошедших между двумя датами;
(<текущая дата> + 1/24) — через час (для типа дата-время).
```

Получить текущую дату можно с помощью специальной функции, имя которой зависит от СУБД: sysdate — для Oracle; now() — для Access и MySQL; getdate() — для MS SQL Server и т. д.

Стандарт SQL включает понятие неопределённого или неизвестного значения — NULL-значения. Для обязательных полей устанавливается ограничение NOT NULL. Это означает, что при изменении значения этого поля или при добавлении новых записей таблицы значение этого поля не может быть неопределенным (NULL). Ограничение NOT NULL можно наложить на поле только один раз, иначе возникает ошибка.

Для любого поля с помощью конструкции DEFAULT <выражение> может быть задано значение по умолчанию. Оно используется в тех случаях, когда при добавлении данных в таблицу значение этого поля не указывается.

Для ограничений целостности можно задавать имена:

```
CONSTRAINT <имя> <ограничение целостности>
```

Примеры создания таблиц

1. Таблица «Отделы» с полями «Номер отдела» (ПК), «Название отдела»:

```
CREATE TABLE depart
(depno NUMERIC(2) CONSTRAINT pk_dep PRIMARY KEY,
    name VARCHAR(80) NOT NULL);
```

2. Таблица «Сотрудники» с полями «Номер отдела» (внешний ключ), «Табельный номер сотрудника» (ПК), «ФИО сотрудника», «Должность», «Оклад», «Дата рождения», «Телефон», «Дата поступления на работу»:

```
CREATE TABLE emp
  (depno NUMERIC(2) CONSTRAINT ref_dep REFERENCES depart,
      tabno CHAR(5) CONSTRAINT pk_emp PRIMARY KEY,
      name VARCHAR(50)NOT NULL,
      post VARCHAR(35)NOT NULL,
      salary NUMERIC(7,2) NOT NULL
              CONSTRAINT check_salary CHECK (salary > 4600),
          born DATENOT NULL,
          phone CHAR(11),
          edate DATE DEFAULT trunc(sysdate));
```

ПРИМЕЧАНИЕ

Функция sysdate возвращает дату и время, поэтому следует с помощью функции trunc (сокращение от truncate) устанавливать время в 0 часов, 0 минут, 0 секунд.

3. Таблица «Дети сотрудников» с полями «Табельный номер родителя» (внешний ключ), «Имя ребенка», «Пол», «Дата рождения»:

```
CREATE TABLE children
(tabno CHAR(5) CONSTRAINT ref_emp REFERENCES emp(tabno),
      name VARCHAR(50)NOT NULL,
      sex CHAR,
      born DATE,
      CONSTRAINT pk_child PRIMARY KEY (tabno, name),
                  /* составной первичный ключ */
          CONSTRAINT check_sex CHECK (sex IN ('м', 'ж')));
```

При выполнении команды создания таблицы в общем случае СУБД делает следующее:

1. Проверяет команду на синтаксическую правильность: при наличии ошибок выдаёт пользователю сообщение о первой выявленной ошибке, в противном случае переходит к следующему пункту.
2. Проверяет права пользователя, запустившего эту команду: если права на выполнение подобной операции есть, то переходит к следующему пункту, в противном случае выдаёт сообщение о недостаточности прав.

3. Записывает в словарь-справочник данных сведения о создаваемой таблице (имя, перечень полей с типами данных, размерами и ограничениями целостности, а также некоторую дополнительную информацию).
4. Создаёт отдельный файл для хранения данных или выделяет отдельную область памяти (это зависит от того, как конкретная СУБД хранит данные; подробнее об этом рассказано в главе 5 «Физическая организация баз данных»).
5. Создаёт дополнительные объекты базы данных, связанные с данной таблицей (например, индексы, о которых рассказано также в главе 5).

ОБРАТИТЕ ВНИМАНИЕ

- Общие ограничения целостности и составные ключи указываются через запятую после последнего поля.
- Если внешний ключ ссылается на первичный ключ (ПК) другого отношения, имена полей ПК можно не указывать (см. создание таблицы *emp*).
- Типы полей внешнего ключа должны совпадать с типами полей первичного (или уникального) ключа, на который он ссылается.
- Если внешний ключ составной, список полей, входящий в ключ, указывается после перечисления всех полей таблицы с ключевым словом FOREIGN KEY:

```
create table exam-- «Расписание зачетов», основная таблица
 (eclass NUMERIC(3),-- аудитория
    edate DATE,-- дата
    edisc VARCHAR(60),-- дисциплина
    UNIQUE (eclass, edate));
create table tab-- «Зачеты», подчинённая таблица
 (tid NUMERIC(6) PRIMARY KEY,
    tclass NUMERIC(3),-- аудитория
    tdate DATE,-- дата
    tgroup CHAR(6),-- группа
    FOREIGN KEY (tclass, tdate) REFERENCES exam(eclass, edate));
```

3.4. Команды модификации данных

К командам модификации данных (DML) относятся добавление, удаление и изменение (обновление) записи (строки таблицы). При выполнении этих команд СУБД делает следующее:

1. Проверяет команду на синтаксическую правильность: при наличии ошибки выдаёт пользователю сообщение о первой выявленной ошибке, иначе — переходит к следующему пункту.

2. Проверяет права пользователя, запустившего команду: если права на выполнение подобной операции есть, то переходит к следующему пункту, иначе — выдаёт сообщение о недостаточности прав.
3. Выполняет операции, предусмотренные командой модификации данных. При выполнении каждой операции (добавлении/изменении/удалении каждой строки) проверяются все установленные для таблицы ограничения целостности. В случае нарушения любого ограничения целостности или возникновения других проблем (переполнения памяти, например) команда DML не выполняется (отменяются все изменения, выполненные в рамках этой команды) и выдаётся сообщение об ошибке. Если же команда выполнилась успешно, выдаётся информация о количестве обработанных строк.

INSERT — добавление записи в таблицу. Синтаксис команды:

```
INSERT INTO <имя таблицы> [(<имя поля>.,..)]
      VALUES (<список выражений>) | <запрос>;
```

Под <запросом> подразумевается команда SELECT (см. ниже), результаты работы которой добавляются в указанную таблицу.

В предложении VALUES указываются выражения, порождающие значения атрибутов новой записи таблицы. Выражение может включать вызовы функций, определенных в данной СУБД, константы, знак операций конкатенации строк (||) или знаки арифметических операций: –, +, *, /. Типы значений выражений должны соответствовать типам полей таблицы. Строки и даты должны заключаться в одинарные кавычки. Формат даты должен соответствовать тому, который установлен в СУБД по умолчанию.

Если значения устанавливаются не для всех полей или порядок значений не соответствует тому порядку полей, который был установлен при создании таблицы, то после имени таблицы в скобках приводится список полей в соответствии со списком значений.

В тех случаях, когда при добавлении записи значение какого-либо поля неизвестно, его можно не устанавливать, пропустив это поле в списке полей или указав для него значение NULL (но только для тех полей, на которые не наложено ограничение целостности NOT NULL).

Если в списке полей отсутствует какое-либо поле таблицы, то ему будет присвоено значение NULL или значение по умолчанию (DEFAULT), если оно определено в командах CREATE TABLE или ALTER TABLE.

Пример

Добавить в таблицу «Сотрудники» новую запись:

```
INSERT INTO emp (depno, tabno, name, post, salary, born, phone)
VALUES(3, '00112', 'Попов В.Г.', 'экономист', 45400, '1979-12-23', '115-
34-11');
```

Одна строка создана.