

Лекция 7

Управление реляционными базами данных

Краткая история языка SQL

Язык реляционных баз данных SQL был разработан в середине 70-х годов в рамках исследовательского проекта экспериментальной реляционной СУБД System R компании IBM. Данный проект включал в себя разработку реляционной системы управления базами данных и языка SEQUEL (Structured English Query Language).

Исходное название SEQUEL только частично отражало суть этого языка. Несмотря на то что язык был ориентирован главным образом на удобную и понятную пользователям формулировку запросов к реляционной базе данных, он уже являлся полноценным языком реляционной базы данных, содержащим, помимо операторов формулирования запросов и манипулирования базой данных, следующие элементы:

- средства определения схемы базы данных и манипулирования ей;
- средства определения ограничений целостности и триггеров;
- средства создания представлений базы данных;
- возможности определения структур физического уровня, поддерживающих эффективное выполнение запросов;
- средства авторизации доступа к отношениям и их полям;
- средства поддержки точек сохранения транзакции и откатов.

В конце 70-х годов модифицированный вариант языка SEQUEL, получивший название SQL, был выпущен корпорацией Oracle в качестве языка коммерческой системы управления базами данных. В 1983 г. компания IBM выпустила SQL в качестве языка управления СУБД DB2.

Американский национальный институт стандартов (ANSI) принял язык SQL в качестве стандарта в 1986 г. С тех пор этот стандарт пересматривался два раза — в 1989 г. были внесены некоторые незначительные изменения, а в 1992 г. Стандарт SQL был довольно существенно расширен.

Основные типы команд

- ▶ DDL (Data Definition Language) — язык определения данных. Команды данной группы используются для создания и изменения структуры объектов базы данных (например, для создания и удаления таблиц);
- ▶ DML (Data Manipulation Language) — язык манипулирования данными. Команды DML используются для манипулирования информацией, содержащейся в объектах базы данных;

- ▶ DCL (Data Control Language) — язык управления данными. Соответствующие команды предназначены для управления доступом к информации, хранящейся в базе данных;
- ▶ DQL (Data Query Language) — язык. Это наиболее часто используемые команды, предназначенные для формирования запросов к базе данных (*запрос* — это обращение к базе данных для получения соответствующей информации);
- ▶ команды администрирования базы данных предназначены для осуществления контроля за выполняемыми действиями и анализа производимых операций;
- ▶ команды управления транзакциями.

ИСПОЛЬЗОВАНИЕ SQL

Существуют три формы SQL: интерактивный (*Interactive*), статический (*Static*) и динамический (*Dynamic*). Функционируют они одинаково, но используются по-разному.

- *Интерактивный SQL* применяется для непосредственной работы с БД -пользователь вводит SQL-оператор, он сразу же выполняется и пользователь видит результат выполнения (или код ошибки).
- *Статический SQL* содержит SQL-операторы, жестко закодированные в теле исполняемого приложения. Наиболее распространен встроенный SQL (*Embedded SQL*), где SQL-код включается в исходный текст (базовой) программы, написанной на другом языке (например, C или Pascal); при использовании встроенного SQL результаты выполнения операторов SQL перенаправляются в переменные, которыми оперирует базовая программа. К настоящему времени SQL встроен в языки Ada, Cobol, Fortran C, Pascal, PL/1, Java, Mumps (теперь M).
- *Динамический SQL* также является частью приложения, но конкретный SQL-код генерируется во время выполнения (Run Time), а не вводится заранее.

Выбор данных из таблиц. Команда SELECT

Выборка данных из БД является наиболее распространенной операцией SQL. Обращение к базе данных называется запросом и для его реализации необходимо использовать команду SELECT. Базовая команда (предписание) SELECT состоит из двух частей, носящих названия *клауз* (clause):

SELECT некоторые данные (имя (имена) колонки)

FROM таблица или некоторые таблицы (имя (имена) таблицы) ;

Клауза SELECT всегда вводится первой, а за ней следует клауза FROM.

Вывод запрошенной информации в данном случае (спецификация вывода не определена) осуществляется на экран дисплея.

Рассмотрим выборку данных в таблицах DEPT и EMP, используя некоторые простые запросы SQL.

Простейший пример - вывести все строки и колонки таблицы DEPT.

Команды SQL, обеспечивающие этот запросы, следующие (строки после последовательности дефисов являются комментарием, не входят в исходный текст и не должны вводиться в диалоге):

```
SELECT DEPTNO,DNAME,LOC --- вводит пользователь !  
FROM DEPT; ---то же...
```

Команда CREATE TABLE

Перед тем как выбрать данные из базы данных, их нужно в нее ввести, а перед этим нужно создать таблицу, в которой эти данные будут храниться.

Ниже приведен пример команды SQL для создания таблицы DEPT :

```
CREATE TABLE DEPT (DEPTNO NUMBER (2),  
DNAME CHAR (14),  
LOC CHAR (13));
```

В команде CREATE TABLE сначала сообщается, как назвать таблицу (DEPT). Далее задаются имена колонок (полей) таблицы (DEPTNO, DNAME, LOC) и тип данных, которые каждая колонка содержит. В создании данной таблицы, например, определяется, что колонка DEPTNO содержит только цифровые данные (NUMBER), а колонки DNAME и LOC любые символьные данные (CHAR) - буквы, числа или знаки пунктуации. Наконец, следует задать максимальную длину любого значения, которое можно хранить в колонках. Например, в команде CREATE TABLE, приведенной выше, задано, что длина имени места расположения (LOC) не должна быть длиннее 13 символов.

Для создания таблицы EMP требуется следующая инструкция SQL:

```
CREATE TABLE EMP (EMPNO NUMBER (4) NOT NULL,  
ENAME CHAR (10),  
JOB CHAR (9),  
MGR NUMBER(4),
```

HIREDATE DATE, --- тип DATE

SAL NUMBER(7,2),

COMM NUMBER(7,2),

DEPTNO NUMBER(2));

Таблица создается как файл с заданной конструкцией CREATE.

Уничтожение таблиц

Для уничтожения таблицы служит предписание DROP, две нижеследующие команды уничтожают таблицы EMP и DEPT:

DROP TABLE EMP;

DROP TABLE DEPT;

Инструкция DROP также применяется для уничтожения индексов таблицы.

Введение строк в таблицу

INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');

INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');

INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');

INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');

Эти четыре инструкции SQL как раз и заполняют приведенную выше таблицу DEPT.

Выбор заданных строк

Из последнего примера видно, как клауза SELECT позволяет выбрать из таблицы заданные колонки. Но для выбора заданных строк нужно ввести в команду SELECT клаузу WHERE:

SELECT *

Клауза WHERE заставляет искать данные в таблице и выводить только те строки, которые удовлетворяют условиям поиска. В примере выше будут возвращены только те строки, где номер отдела работника был равен 30

(WHERE DEPTNO = 30).

Сложные (комбинированные) условия поиска

Иногда необходимо задать несколько условий поиска в клаузе WHERE.

Предположим, например, что необходимо иметь список менеджеров компании с окладом более 2800 долларов:

SELECT ENAME, JOB, SAL

FROM EMP

WHERE JOB = 'MANAGER'

AND SAL > 2800; --- логическая связка AND

Несколько условий поиска объединены здесь словом ключевым AND (И) (JOB='MANAGER' AND SAL>2800). Соединитель AND означает, что данные должны удовлетворять обоим перечисленным условиям поиска. Можно соединять по AND любое число условий.

Альтернативные условия поиска

В дополнение к возможности выбирать строки, соответствующие *всем* условиям, можно выбирать и строки, соответствующие *любому* из нескольких условий:

```
SELECT ENAME, JOB, SAL
```

```
FROM EMP
```

```
WHERE JOB = 'MANAGER'
```

В этом примере соединяются условия поиска словом OR (ИЛИ) (JOB='MANAGER' OR SAL > 2800). Логическая связка OR значит, что если данные удовлетворяют одному из нескольких условий, то они будут выбраны.

Отрицательные условия поиска

Можно выбирать строки, *не* удовлетворяющие данному условию. Например, несложно выбрать всех менеджеров, которые *не работают* в отделе 30:

```
SELECT ENAME, JOB, DEPTNO
```

```
FROM EMP
```

```
WHERE JOB = 'MANAGER'
```

```
AND DEPTNO != 30; --- пример НЕ РАБНО
```

Можно комбинировать AND, OR и NOT в одном запросе, чтобы выбрать нужную информацию; для достижения нужного результата следует применять скобки.

Поиск в диапазоне

Оператор BETWEEN позволяет выбирать строки в заданном диапазоне (включая границы одного).

Например, перечислим всех работников, оклад которых находится между 1200 и 1400 долларами:

```
SELECT ENAME, SAL
```

```
FROM EMP
```

```
WHERE SAL BETWEEN 1200 AND 1400;
```

Поиск значений в списке

Предписание IN дает возможность выбрать строки, содержащие заданные значения. Перечислим все отделы, номера которых 10 или 30:

```
SELECT *  
FROM DEPT  
WHERE DEPTNO IN (10,30); --- только 10 или 30
```

Отметим, что следует заключить список значений в скобки - (10,30). Для этого запроса возможно было применить логическую связку OR для получения того же самого результата (WHERE DEPTNO=10 OR DEPTNO=30).

Последовательности сопоставления символов

Можно также выбрать строки, соответствующие образцу символов или цифр, который задается после клаузы LIKE. Например, перечислим всех работников, имеющих в имени символ 'R' в третьей от начала позиции:

```
SELECT ENAME  
FROM EMP  
WHERE ENAME LIKE '__R%';
```

В это примере используется оператор LIKE для указания выбрать все строки из таблицы EMP, в которых третий символ в фамилии работника суть R, то есть соответствует (LIKE) образцу, который задан как (__R%). Каждый символ подчеркивания (а их два) говорит об одной позиции любого символа, а знак процента (%) задает любую строку без символов или с любым их количеством.

Упорядочение строк по запросу

Во всех предыдущих примерах строки выводились на экран в порядке, заданном существующей таблицей. Возможно управлять порядком вывода строк на экран путем ввода клаузы ORDER BY (ПО) в конец команды SELECT. Например, если желательно вывести список работников отдела 30, упорядоченный по окладам, следует воспользоваться следующим запросом:

```
SELECT SAL,JOB,ENAME  
FROM EMP  
WHERE DEPTNO = 30  
ORDER BY SAL; --- пример клаузы ORDER BY
```

Клауза ORDER BY ведет сортировку строк по увеличивающемуся (меньший оклад сначала) порядку. Но упорядочение не ограничивается только одной последовательностью в одной колонке. Например, несложно перечислить всех работников по порядку работы и внутри него (порядка работы) по окладу:

```
SELECT JOB,SAL,ENAME
```

```
FROM EMP
```

```
ORDER BY JOB, SAL DESC; --- в порядке уменьшения SAL
```