Базы данных и СУБД. Введение в SQL. Основные понятия. Язык запросов SQL

SQL (*Structured Query Language* - "язык структурированных запросов") - универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. Реляционная база данных - база данных, основанная на реляционной модели данных. Реляционная модель данных (РМД) - логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики как теории множеств и логика первого порядка.является непроцедурным языком и не содержит операторов управления, организации подпрограмм, ввода-вывода и т.п. В связи с этим SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД (например, FoxPro СУБД Visual FoxPro, ObjectPAL СУБД Paradox, Visual Basic for Applications СУБД Access).

В современных СУБД с интерактивным интерфейсом можно создавать запросы, используя другие средства, например QBE. Однако применение SQL зачастую позволяет повысить эффективность обработки данных в базе. Например, при подготовке запроса в среде Access можно перейти из окна Конструктора запросов (формулировки запроса по образцу на языке QBE) в окно с эквивалентным оператором SQL. Подготовку нового запроса путем редактирования уже имеющегося в ряде случае проще выполнить путем изменения оператора SQL. В различных СУБД состав операторов SQL может несколько отличаться.

Язык SQL не обладает функциями полноценного языка разработки, а ориентирован на доступ к данным, поэтому его включают в состав средств разработки программ. В этом случае его называют встроенным SQL. Стандарт языка SQL поддерживают современные реализации следующих языков программирования: PL/1, Ada, C, COBOL, Fortran, MUMPS и Pascal.

В специализированных системах разработки приложений типа клиент-сервер (данную архитектуру мы рассмотрим позже) среда программирования, кроме того, обычно дополнена коммуникационными средствами (установление и разъединение соединений с серверами БД, обнаружение и обработка возникающих в сети ошибок и. т.д.), средствами разработки пользовательских интерфейсов, средствами проектирования и отладки.

Различают два основных метода использования встроенного SQL: статический и динамический.

При статическом использовании языка (статический SQL) в тексте программы имеются фиксированные по структуре вызовы функций языка SQL, включаемые в выполняемый модуль в процессе компиляции. Параметры запросов (обычно представляют константные значения, с которыми сравниваются значения полей в таблицах), являющиеся переменными языка программирования, позволяют добиться некоторой гибкости статических запросов.

При динамическом использовании языка (динамический SQL) предполагается динамическое построение запроса в форме текстовой строки. Данная строка используется как параметр для функции выполнения SQL-запросов, которая выполняет синтаксический анализ строки запроса и формирует на его основе последовательность команд БД. Динамический метод обычно применяется в случаях, когда в приложении заранее неизвестен вид SQL-вызова.

В результате выборки данных из одной или нескольких, таблиц может быть получено множество записей, называемое представлением. Представление по существу является таблицей, формируемой в результате выполнения запроса, которая существует "виртуально" только до завершения выполнения программы.

Для удобства работы с представлениями в язык SQL введено понятие курсора. Курсор представляет собой своеобразный указатель на набор записей в представлении, обеспечивающий в каждый момент доступ лишь к некоторой небольшой части строк представления.

С помощью операторов перемещения курсора по записям можно получить доступ ко всем строкам таблицы.

Основные операторы языка SQL

Функции любой *СУБД* включают:

1. создание, удаление, изменение *базы данных* (БД);
2. добавление, изменение, удаление, назначение прав пользователя;
3. внесение, удаление и изменение данных в БД (таблиц и записей);
4. выборку данных из БД.

К первым двум функциям имеют доступ только администраторы *СУБД* или привилегированные пользователи. Рассмотрим, как решаются последние две задачи (на самом деле это семь задач).

Прежде чем что-либо делать с данными, нужно создать таблицы, в которых эти данные будут храниться, научиться изменять структуру этих таблиц и удалять их, если потребуется. Для этого в языке *SQL* существуют операторы *CREATE TABLE* , *ALTER TABLE* и *DROP TABLE* .

Оператор CREATE TABLE

Оператор ***CREATE TABLE*** создает таблицу с заданным именем в текущей *базе данных* . Правила для допустимых имен таблицы приведены в документации. Если нет активной текущей *базы данных* или указанная таблица уже существует, то возникает ошибка выполнения команды.

В версии *MySQL* 3.22 и более поздних *имя таблицы* может быть указано как имя_базы_данных.имя_таблицы. Эта форма записи работает независимо от того, является ли указанная *база данных* текущей.

В версии *MySQL* 3.23 при создании таблицы можно использовать ключевое слово *TEMPORARY*. Временная таблица автоматически удаляется по завершении соединения, а ее имя действительно только в течение данного соединения. Это означает, что в двух разных соединениях могут использоваться временные таблицы с одинаковыми именами без конфликта друг с другом или с существующей таблицей с тем же именем (существующая таблица скрыта, пока не удалена временная таблица). В версии *MySQL* 4.0.2 для создания временных таблиц необходимо иметь привилегии CREATE *TEMPORARY* TABLES.

В версии *MySQL* 3.23 и более поздних можно использовать ключевые слова IF NOT EXISTS для того, чтобы не возникала ошибка, если указанная таблица уже существует. Следует учитывать, что при этом идентичность структур этих таблиц не проверяется.

Каждая таблица представлена набором определенных файлов в директории *базы данных*.

Синтаксис

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
    имя_таблицы [(определение_столбца,...)]
     [опции_таблицы] [select_выражение]
```

В выражении определение_столбца перечисляют, какие столбцы должны быть созданы в таблице. Каждый столбец таблицы может быть пустым (NULL), иметь значение по умолчанию, являться ключом или автоинкрементом. Кроме того, для каждого столбца обязательно указывается тип данных, которые будут в нем храниться. Если не указывается ни NULL, ни NOT NULL, то столбец интерпретируется так, как будто указано NULL. Если поле помечают как автоинкремент (AUTO_INCREMENT), то его значение автоматически увеличивается на единицу каждый раз, когда происходит добавление данных в таблицу и в это поле записывается пустое значение (NULL, т.е. ничего не записывается) или 0. Автоинкремент в таблице может быть только один, и при этом он обязательно должен быть проиндексирован. Последовательность AUTO_INCREMENT начинается с 1. Наличие автоинкремента является одной из особенностей *MySQL*. Формально описание столбца (определение_столбца) выглядит так:

```
имя_столбца тип [NOT NULL | NULL]
  [DEFAULT значение_по_умолчанию]
  [AUTO_INCREMENT][PRIMARY KEY]
  [reference_definition]
```

Тип столбца (тип в выражении определение_столбца) может быть одним из следующих:

- целый: INT[(length)] [*UNSIGNED*] [ZEROFILL]
- действительный: REAL[(length,decimals)] [*UNSIGNED*] [ZEROFILL]
- символьный: CHAR(length) [BINARY] и VARCHAR(length) [BINARY]
- дата и время: DATE и TIME
- для работы с большими объектами: *BLOB*
- текстовый: TEXT
- перечислимое множество: *ENUM*(value1,value2,value3,...) и SET(value1,value2,value3,...)

Полный список типов смотрите в документации *MySQL*.

Вместо перечисления столбцов и их свойств в определении_столбца можно задавать списки ключевых и индексных полей, ограничения и проверки:

```
PRIMARY KEY (имя_индексируемого_столбца, ...)
```

или

```
KEY [имя_индекса] (имя_индексируемого_столбца,...)
```

или

```
INDEX [имя_индекса] (имя_индексируемого_столбца,...)
```

или

```
UNIQUE [INDEX] [имя_индекса]
  (имя_индексируемого_столбца,...)
```

или

```
FULLTEXT [INDEX] [имя_индекса]
  (имя_индексируемого_столбца,...)
```

или

[CONSTRAINT symbol]
FOREIGN KEY [имя_индекса]
(имя_индексируемого_столбца,...)
[reference_definition]

или

CHECK (expr)

При задании всех этих элементов указывается список полей (столбцов), которые будут входить в индекс, ключ или ограничение, имя_индексируемого_столбца записывается следующим образом:

имя_столбца [(длина_индекса)]

FOREIGN KEY, CHECK и REFERENCES на самом деле ничего не делают в *MySQL*. Они добавлены только для совместимости с другими SQL-серверами. Поэтому на них мы останавливаться не будем.

Кроме всего перечисленного, при создании таблицы можно указать некоторые ее свойства (опции_таблицы), например такие:

- тип таблицы: TYPE = {BDB | *HEAP* | *ISAM* | *InnoDB* | MERGE | MRG_MYISAM | *MYISAM* }
- начальное значение счетчика автоинкремента: AUTO_INCREMENT = число
- средняя длина строк в таблице: AVG_ROW_LENGTH = число
- комментарии к таблице (строка из 60 символов): COMMENT = "строка"
- максимальное и минимальное предполагаемое число строк: MAX_ROWS = число и MIN_ROWS = число

И последний (опять же опциональный) элемент команды CREATE – это выражение *SELECT* (select_выражение). Синтаксис такой:

[IGNORE | REPLACE] SELECT ...
(любое корректное выражение SELECT)

Если при создании таблицы в команде CREATE указывается выражение *SELECT* , то все поля, полученные выборкой, добавляются в создаваемую таблицу.

Пример Создадим таблицу Persons

```
mysql>CREATE TABLE Persons
   (id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50), last_name
   VARCHAR(100), death_date INT,
    description TEXT, photo INT,
    citizenship CHAR(50) DEFAULT 'Russia');
```