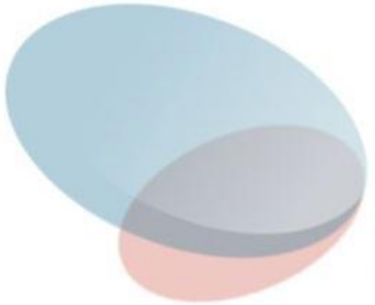


СУБД
Лекция 1
Основы языка SQL

Основы языка SQL

История

Язык SQL (Structured Query Language - структурированный язык запросов) представляет собой **стандартный высокоуровневый язык описания данных и манипулирования ими в** системах управления базами данных (**СУБД**), **построенных на основе реляционной модели данных**.

В **начале 1970-х** годов компанией IBM была разработана экспериментальная реляционная СУБД **IBM System R**, для которой затем был создан специальный **язык SEQUEL**, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как **Structured English QUEry Language** — «структурированный английский язык запросов». Позже язык SEQUEL был переименован в **SQL**.

Основы языка SQL

Стандартизация

Целью стандартизации является переносимость приложений между различными СУБД.

Год Название Иное название Изменения

1986 SQL-86 SQL-87 Первый вариант стандарта, принятый институтом ANSI (Американский национальный институт стандартов) и одобренный ISO (Международная организация по стандартизации) в 1987 году.

1989 SQL-89 FIPS 127-1 (Federal Information Processing Standards, Федеральные Стандарты Обработки Информации) Немного доработанный вариант предыдущего стандарта.

1992 SQL-92 SQL2, FIPS 127-2 Значительные изменения (ISO 9075); уровень Entry Level (начальный (англ. entry), средний (англ. intermediate), полный (англ. full)) стандарта SQL-92 был принят как стандарт FIPS 127-2.

1999 SQL:1999 SQL3 Добавлена поддержка регулярных выражений рекурсивных запросов, поддержка триггеров, базовые процедурные расширения, нескалярные типы данных и некоторые объектно-ориентированные возможности.

Основы языка SQL

(Регуля́рные выраже́ния (англ. regular expressions)— это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. По сути это строка-образец (англ. pattern, по-русски её часто называют «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска.)

2003 SQL:2003 Введены расширения для работы с XML-данными (XML — текстовый формат, предназначенный для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например, XHTML)), оконные функции (применяемые для работы с OLAP-базами данных) (OLAP (англ. online analytical processing, аналитическая обработка в реальном времени) — технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу) , генераторы последовательностей и основанные на них типы данных.

2006 SQL:2006 Функциональность работы с XML-данными значительно расширена. Появилась возможность совместно использовать в запросах SQL и XQuery (XQuery — язык запросов, разработанный для обработки данных в формате XML).

Основы языка SQL

2008 SQL:2008 Улучшены возможности оконных функций («окно» - набор строк, характеризуемых равенством значений списка выражений), устранены некоторые неоднозначности стандарта SQL:2003

В настоящее время действует стандарт, принятый в 2003 году (SQL:2003) с небольшими модификациями, внесёнными позже.

Несмотря на наличие международного стандарта ANSI SQL-92, многие компании, занимающиеся разработкой СУБД (например, **Oracle, Sybase, Microsoft, MySQL AB**), вносят изменения в язык SQL, применяемый в разрабатываемой СУБД, тем самым отступая от стандарта. Таким образом, **появляются специфичные для каждой конкретной СУБД диалекты языка SQL**.

Основы языка SQL

SQL не является языком программирования (то есть не предоставляет средств для автоматизации операций с данными).

Вводимые разными производителями **расширения** касались в первую очередь процедурных расширений. Это **хранимые процедуры** (stored procedures) и **процедурные языки-** «надстройки».

Практически в каждой СУБД применяется свой процедурный язык. Стандарт для процедурных расширений представлен спецификацией SQL/PSM.

Основы языка SQL

Процедурные расширения для популярных СУБД :

СУБД	Краткое название	Расшифровка
InterBase/Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL (англ.)	SQL Procedural Language (расширяет SQL/PSM); также в DB2 хранимые процедуры могут писаться на обычных языках программирования: Си, Java и т. д.
MS SQL Server/Sybase ASE	Transact-SQL	Transact-SQL
MySQL	SQL/PSM	SQL/Persistent Stored Module
Oracle	PL/SQL	Procedural Language/SQL (основан на языке Ada)
PostgreSQL	PL/pgSQL	Procedural Language/ PostgreSQL Structured Query Language (очень похож на Oracle PL/SQL)

Основы языка SQL

Реляционные СУБД

В настоящее время наибольшее распространение получили **реляционные SQL СУБД двух групп**:

- мощные крупные **коммерческие СУБД**, ориентированные на хранение огромных объемов информации (от гигабайт);
- мобильные компактные **свободно распространяемые** (в том числе и в исходных кодах) СУБД, использование которых оправдано и для БД объемом всего лишь в десятки килобайт.

Основы языка SQL

Наиболее известными **СУБД первой группы** являются:

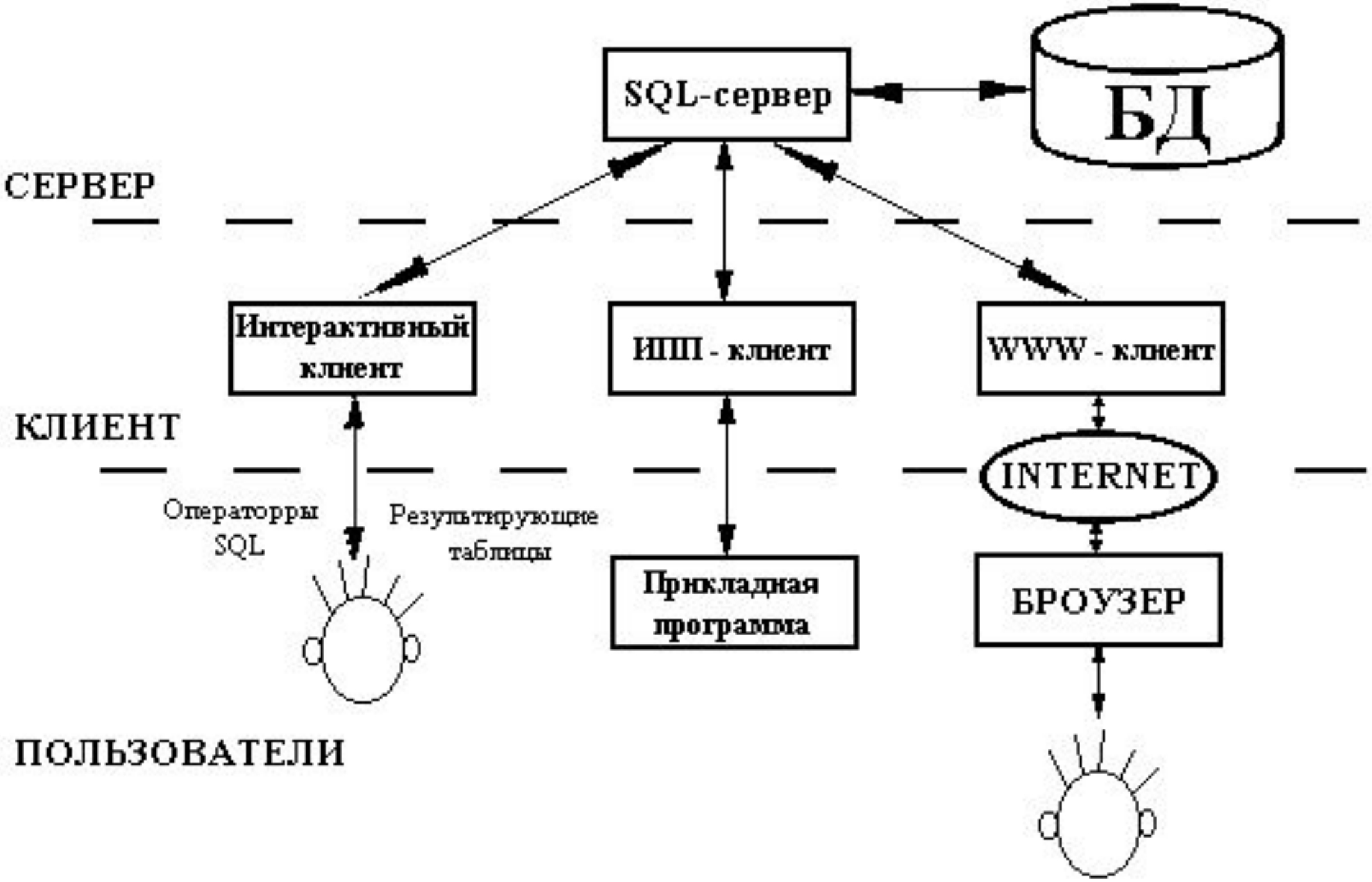
- Sybase SQLserver фирмы Sybase, Inc.;
- Oracle фирмы Oracle Corporation;
- Ingres фирмы Computer Associates International;
- Informix фирмы Informix Corporation.

К наиболее популярным **СУБД второй группы** относятся:

- PostgreSQL организации PostgreSQL;
- microSQL фирмы Hughes Technologies Pty. Ltd.;
- mySQL фирмы T.C.X DataKonsult AB.

Все перечисленные выше СУБД построены по принципу "**клиент-сервер**", как это показано на рисунке ниже.

Основы языка SQL



Основы языка SQL

SQL-сервер реализует хранение данных и манипулирование ими. Он принимает запросы на языке SQL от своих клиентов, выполняет их и возвращает результаты (чаще всего в виде вновь построенных таблиц) клиентам. Для общения с клиентами используется специальный протокол (как правило, реализованный в виде протокола прикладного уровня стека сетевых протоколов TCP/IP).

Клиентскую часть СУБД составляют клиенты **трех основных типов.**

Интерактивные клиенты, обеспечивающие пользователю-человеку возможность общения с SQL-сервером непосредственно с помощью языка SQL.

Основы языка SQL

ИПП-клиенты, обеспечивающие интерфейс прикладного программирования (ИПП) прикладным программам, использующим средства SQL-сервера. Такой ИПП может быть средством общения прикладной программы с SQL-сервером на языке SQL или набором стандартных функций доступа к реляционной SQL БД без формирования символьных строк запросов (например, стандартный интерфейс ODBC). (ODBC (англ. Open Database Connectivity) — это программный интерфейс (API) доступа к базам данных, разработанный фирмой Microsoft, в сотрудничестве с Simba Technologies на основе спецификаций Call Level Interface (CLI). Стандарт CLI призван унифицировать программное взаимодействие с СУБД, сделать его независимым от поставщика СУБД и программно-аппаратной платформы.)

WWW-клиенты, встраиваемые в World Wide Web-сервера и обеспечивающие доступ к информационным возможностям SQL-сервера пользователям сети Internet по протоколу HTTP (протоколу передачи гипертекстовых документов).

Основы языка SQL

Основы синтаксиса языка SQL

Язык SQL представляет собой совокупность

- операторов;
- инструкций;
- и вычисляемых функций.

"**Программа**" на языке SQL - простая **линейная последовательность операторов языка SQL**. Язык SQL в своем "чистом" виде **операторов управления порядком выполнения запросов** к БД (типа циклов, ветвлений, переходов) **не имеет**.

Операторы языка SQL строятся с применением:

- **зарезервированных** ключевых **слов**;
- идентификаторов (**имен**) таблиц и столбцов таблиц;
- логических, арифметических и строковых **выражений**, используемых для формирования критериев поиска информации в БД и для вычисления значений ячеек результирующих таблиц;
- **идентификаторов** (имен) операций и функций, используемых в выражениях.

Основы языка SQL

В языке SQL не делается различия между прописными (большими) и строчными (маленькими) буквами, т.е., например, строки "SELECT", "Select", "select" представляют собой одно и то же ключевое слово.

Но, согласно общепринятому стилю программирования, операторы (и другие зарезервированные слова) в SQL всегда следует писать прописными буквами.

Все ключевые слова, имена функций и, как правило, имена таблиц и столбцов записываются латинскими буквами.

Для конструирования имен таблиц и их столбцов допустимо использовать буквы, цифры и знак "_" (подчеркивание), но первым символом имени обязательно должна быть буква.

Запрещено использование ключевых слов и имен функций в качестве идентификаторов таблиц и имен столбцов.

Основы языка SQL

Оператор начинается с ключевого слова-глагола (например, "CREATE" - создать, "UPDATE" - обновить, "SELECT" - выбрать и т.п.) и заканчивается знаком ";" (точка с запятой).

Оператор записывается в свободном формате и может занимать несколько строк. Допустимыми разделителями лексических единиц в операторе являются:

- один или несколько пробелов,
- один или несколько символов табуляции,
- один или несколько символов "новая строка".

Комментарии при использовании в различных СУБД в текстах "программ" на языке SQL могут помечаться следующими способами:

- от двойного минуса ("--") до конца строки;
- от символа "#" до конца строки;
- между последовательностями "/*" и "*/" (стиль комментариев языка СИ).

Основы языка SQL

Типы данных языка SQL

Типы данных, используемые в языке SQL для хранения информации в столбцах таблиц БД, весьма разнообразны. Но производители конкретных реляционных СУБД "улучшают" множество типов данных, регламентируемых стандартом, реализуя свои собственные версии и расширения.

Следующие типы данных являются стандартными.

- INT[(len)] - **целое** число длиной **4 байта**, представляемое при выводе максимально len цифрами;
- SMALLINT[(len)] - **целое** число длиной **2 байта**, представляемое при выводе максимально len цифрами;
- FLOAT[(len,dec)] - **действительное** число, представляемое при выводе максимально len символами с dec цифрами после десятичной точки;
- CHAR(size) - **строка символов** фиксированной длины размером size символов;

Основы языка SQL

Типы данных языка SQL

- VARCHAR(size) - **строка символов** переменной длины максимальным размером до size символов;
- BLOB (Binary Large OBject) - **массив** произвольных (двоичных) **байтов** (максимальный размер зависит от реализации, обычно это 65535 байт); этот тип данных может использоваться, например, для хранения изображений;
- DATE - астрономическая **дата**;
- TIME - астрономическое **время**.

Символьные константы (типа CHAR и VARCHAR) записываются как последовательности символов, заключенные в одиночные апострофы.

Основы языка SQL

- TINYINT[(length)] [UNSIGNED] [ZEROFILL]
- SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
- MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
- INT[(length)] [UNSIGNED] [ZEROFILL]
- INTEGER[(length)] [UNSIGNED] [ZEROFILL]
- BIGINT[(length)] [UNSIGNED] [ZEROFILL]
- REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
- DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
- FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
- DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
- NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
- CHAR(length) [BINARY]
- VARCHAR(length) [BINARY]
- DATE
- TIME
- TIMESTAMP
- DATETIME
- TINYBLOB
- BLOB
- MEDIUMBLOB
- LONGBLOB
- TINYTEXT
- TEXT
- MEDIUMTEXT
- LONGTEXT
- ENUM(value1,value2,value3,...)
- SET(value1,value2,value3,...)

Основы языка SQL

Десятичные константы (типа FLOAT) могут записываться в "научной" нотации как последовательности следующих компонент:

- знак числа;
- десятичное число с точкой;
- символ "e";
- знак ("+" или "-") показателя степени;
- целое число, играющее роль показателя степени числа 10.

Например, десятичное число **-0,123** может быть записано как **-12.3e-2**.

Отличие типов данных CHAR и VARCHAR: для хранения в таблице строк символов типа CHAR используется точно size байт (хотя содержимое может быть значительно короче), а для строк типа VARCHAR незанятые символами строк ("пустые") байты в таблице не хранятся.

Величины len и dec (в отличие от size) не влияют на размер хранения данных в таблице, а только **форматируют вывод** данных из таблицы.

Основы языка SQL

Тип данных BLOB поддерживается непосредственно не всеми СУБД, однако каждая из них предлагает его аналог (например, BINARY или IMAGE).

Рекомендация. Разрабатывая мобильное приложение (рассчитанное на работу в среде различных СУБД), старайтесь **без необходимости избегать использования необязательных возможностей** в описании типов данных.

Основы языка SQL

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории, например, вместо "отношений" используются "таблицы", вместо "кортежей" - "строки", вместо "атрибутов" - "колонки" или "столбцы".

Стандарт языка SQL, хотя и основан на реляционной теории, но во многих местах отходит он нее. **Например**, отношение в реляционной модели данных не допускает наличия одинаковых кортежей, а таблицы в терминологии SQL могут иметь одинаковые строки. Имеются и другие отличия.

Язык SQL является реляционно полным. Это означает, что любой оператор реляционной алгебры может быть выражен подходящим оператором SQL.

Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что **превратился в инструмент программиста**.

Основы языка SQL

В язык SQL в качестве составных частей входят:

- Язык манипулирования данными (Data Manipulation Language, DML)
- Язык определения данных (Data Definition Language, DDL)
- Язык управления данными (Data Control Language, DCL)
- Язык управления транзакциями (Transaction Control Language, TCL)

Основы языка SQL

Операторы определения данных (Data Definition Language, DDL):

- CREATE создать объект БД (саму базу DATABASE , таблицу TABLE, представление VIEW , индекс INDEX, триггер TRIGGER, процедуру PROCEDURE и т. д.);
- ALTER изменить объект;
- DROP удалить объект;

операторы манипулирования данными (Data Manipulation Language, DML):

- SELECT выбрать данные, удовлетворяющие заданным условиям;
- INSERT вставить (добавить) новые данные;
- UPDATE изменить существующие данные;
- DELETE удалить данные;

Основы языка SQL

операторы определения доступа к данным (Data Control Language, DCL):

- GRANT предоставить пользователю (группе) права на определенные операции с объектом;
- REVOKE забрать ранее выданные права;
- DENY задать запрет, имеющий приоритет над разрешением;

операторы управления транзакциями (Transaction Control Language, TCL):

- COMMIT применяет транзакцию;
- ROLLBACK откатывает все изменения, сделанные в контексте текущей транзакции;
- SAVEPOINT делит транзакцию на более мелкие участки.

Основы языка SQL

С точки зрения прикладного интерфейса существуют **две** формы **SQL**:

- Интерактивный SQL
- Встроенный или вложенный SQL.

Интерактивный SQL используется для непосредственного (оперативного) выполнения запросов пользователя к базе данных. При завершении ввода команды в этой форме SQL она поступает на выполнение, и вы сможете увидеть результат немедленно (если он вообще получится).

Встроенный SQL состоит из команд SQL, помещённых внутри программ, которые обычно написаны на другом языке (типа Си или Паскаля). Это делает эти программы более мощными и эффективными.

Основы языка SQL

Динамический SQL

Для упрощения создания интерактивных SQL-ориентированных систем **во встроенный SQL были включены операторы, позволяющие во время выполнения транзакции откомпилировать и выполнить любой оператор SQL.**

Оператор PREPARE вызывает динамическую компиляцию оператора SQL, текст которого содержится в указанной переменной символьной строке включающей программы. Текст может быть помещен в переменную при выполнении программы любым допустимым способом, например, введен с терминала.

Оператор DESCRIBE служит для получения информации об указанном операторе SQL, ранее подготовленном с помощью оператора PREPARE.

Для выполнения ранее подготовленного оператора SQL, не являющегося оператором выборки, служит **оператор EXECUTE** и т.д.

Основы языка SQL

Операторы DDL

1. Создание базы данных

CREATE DATABASE [IF NOT EXISTS] db_name;

2. Создание таблицы

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
[(create_definition,...)] [table_options] [select_statement]

tbl_name - имя таблицы,

TEMPORARY - параметр используется для создания временной таблицы,

create_definition - определяет внутреннюю структуру создаваемой таблицы (названия и типы полей, ключи, индексы и т.д.):

- col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT] [PRIMARY KEY] [reference_definition],
- PRIMARY KEY (index_col_name,...),
- KEY [index_name] (index_col_name,...),
- INDEX [index_name] (index_col_name,...),
- UNIQUE [INDEX] [index_name] (index_col_name,...),
- [CONSTRAINT symbol] FOREIGN KEY [index_name] (index_col_name,...) [reference_definition],
- CHECK (expr).

Основы языка SQL

```
CREATE TABLE имя_табл (
имя_столбца тип_данных [NOT NULL ] [DEFAULT по_умолч]
        [PRIMARY KEY],
…
PRIMARY KEY имя_ключа (имя_столбца, ...),
KEY имя_ключа (имя_столбца, ...),
[CONSTRAINT имя_ограничения] FOREIGN KEY (имя_поля,…)
        REFERENCES имя_родительской_табл (имя_поля,…)
                [ON DELETE CASCADE],
CHECK (имя_поля IN (знач,…) | имя_поля > знач |…)
);
```

Основы языка SQL

Пример

```
1) CREATE TABLE P(PNUM INT PRIMARY KEY,
PNAME  VARCHAR(15) NOT NULL,
PSTATUS SMALLINT(3) NOT NULL);
2) CREATE TABLE D(DNUM INT PRIMARY KEY,
DNAME  VARCHAR(15) NOT NULL,
DSTATUS SMALLINT(3) NOT NULL);
3) CREATE TABLE PD(PNUM INT NOT NULL,
DNUM INT NOT NULL,
VOLUME INT NOT NULL,
PRIMARY KEY PNUM_DNUM(PNUM, DNUM),
FOREIGN KEY (PNUM) REFERENCES P (PNUM),
FOREIGN KEY (DNUM) REFERENCES D (DNUM)
);
```

Основы языка SQL

3. Удаление таблицы

DROP TABLE [IF EXISTS] tbl_name [, tbl_name,...];

4. Изменение таблицы

ALTER TABLE tbl_name alter_specification [, alter_specification ...];

alter_specification:

- ADD [COLUMN] (create_definition, create_definition,...)
- ADD INDEX [index_name] (index_col_name,...)
- ADD UNIQUE [index_name] (index_col_name,...)
- CHANGE [COLUMN] old_col_name create_definition [FIRST | AFTER column_name]
- DROP [COLUMN] col_name
- DROP INDEX index_name
- RENAME [TO] new_tbl_name
- …

Основы языка SQL

Примеры

1) ALTER TABLE P

ADD PTOWN VARCHAR(15),

ADD INDEX index_town (PTOWN);

2) ALTER TABLE D

ADD DPRICE INT NOT NULL

AFTER DNAME;

5. Удаление базы данных

DROP DATABASE [IF EXISTS] db_name;

Основы языка SQL

Операторы DML

1. Вставка новых строк в таблицу

```
INSERT  [INTO] tbl_name [(col_name,...)]
    VALUES (expression,...),(...),...
```

или

```
INSERT [INTO] tbl_name
    SET col_name=expression, col_name=expression, ...
```

или

```
 INSERT  [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

Основы языка SQL

Примеры

1) Вставка одной строки в таблицу

```
INSERT INTO  P (PNUM, PNAME, PSTATUS)
  VALUES (4, ‘Иванов’,3);
```

2) В таблицу TMP_TABLE вставляются данные о поставщиках из таблицы P, имеющих номера, большие 2

```
INSERT INTO  TMP_TABLE (PNUM, PNAME)
  SELECT PNUM, PNAME
    FROM P
    WHERE P.PNUM>2;
```

Основы языка SQL

2. Удаление строк из таблицы

DELETE FROM table_name [WHERE where_definition]

[ORDER BY ...] [LIMIT rows]

LIMIT - задает максимальное количество строк (с начала таблицы), которые могут быть удалены за текущий запрос;

ORDER BY - задает имя поля, или имена полей через запятую, по которым происходит сортировка удаляемых записей.

Эта возможность актуальна при необходимости удалить только определенное количество записей, отсортированных по какому-либо свойству.

Основы языка SQL

Примеры

1) Удаление нескольких строк в таблице:

```
DELETE FROM P  WHERE PSTATUS = 1;
```

2) Удаление всех строк в таблице (таблица сохраняется):

```
DELETE FROM P;
```

3) Удаление одной детали с наименьшим статусом

```
DELETE FROM D
      ORDER BY DSTATUS LIMIT 1;
```

Основы языка SQL

3. Обновление записи в таблице

```
UPDATE tbl_name
    SET col_name1=expr1 [, col_name2=expr2, ...]
    [WHERE where_definition]
    [LIMIT rows];
```

SET - после этого ключевого слова должен идти список полей таблицы, которые будут обновлены и непосредственно сами новые значения полей в виде: имя_поля='значение‘;

WHERE - задает условие отбора записей, подлежащих изменению;

LIMIT - задает максимальное количество строк, которые могут быть изменены.

Основы языка SQL

Примеры

1) Обновление строк в таблице:

```
UPDATE P  SET PNAME = ‘Pushnikov’
  WHERE PNUM = 1;
UPDATE P  SET PTOWN = ‘TVER’
  WHERE P TOWN= ‘MOSCOW’;
```

2) Обновление всех строк в таблице

```
UPDATE  P  SET PSTATUS = PSTATUS +1;
```

Основы языка SQL

4. Поиск (выборка) записей

```
SELECT  [DISTINCT | ALL]
   expression,...
   [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
   [FROM table_references
      [WHERE where_definition]
      [GROUP BY {unsigned_integer | col_name | formula} [ASC |
        DESC], ...]
      [HAVING where_definition]
      [ORDER BY {unsigned_integer | col_name | formula} [ASC |
        DESC], ...]
      [LIMIT [offset,] rows]
```

Основы языка SQL

SELECT [ALL | DISTINCT] в_выражение, ...

FROM имя_табл [син_табл], ...

[WHERE сложн_условие]

[GROUP BY полн_имя_столбца|ном_столбца, ...]

[HAVING сложн_условие]

[ORDER BY полн_имя_столбца|ном_столбца [ASC|DESC], ...] ;

Результатом работы оператора является выводимая на стандартный вывод вновь построенная таблица, для которой

количество и смысл столбцов определяется списком элементов **в_выражение**;

содержимое строк определяется содержимым исходных таблиц из списка FROM и критерием выборки, задаваемым **сложн_условие**.

Основы языка SQL

син_табл - необязательный синоним имени таблицы, используемый для сокращения длины записи выражений и условий в операторе SELECT.

полн_имя_столбца - полное имя столбца в виде

[имя_табл|син_табл.]имя_столбца

Конкретизирующий таблицу префикс в имени столбца необходим только для различения столбцов, имеющих одинаковое имя в разных таблицах из списка FROM.

ном_столбца - номер столбца результирующей таблицы.

Замечание. Подчеркнутый вариант ALL, ASC используется по умолчанию.

Основы языка SQL

I. Отбор данных из одной таблицы

1) Выбрать **все данные из таблицы** поставщиков

```
SELECT *
  FROM P;
```

Замечание. **Дубликаты строк автоматически не отбрасываются**.

2) Выбрать все **строки** из таблицы поставщиков, **удовлетворяющие** некоторому **условию**

```
 SELECT *
  FROM P
  WHERE P.PNUM > 2;
```

Замечание. В разделе WHERE можно использовать сложные логические выражения, использующие поля таблиц, константы, операции сравнения (>, <, = и т.д.), скобки, знаки логических операций AND, OR и NOT.

Основы языка SQL

В разделе WHERE можно использовать предикаты сравнения, in, between, null, like и т.д.

Примеры условий отбора строк

1) (PD.PNUM, PD.DNUM) = (1, 25) эквивалентно

PD.PNUM = 1 AND PD.DNUM = 25

2) P.PNUM IN (1, 2, 3, 5)

3) P.PNUM [NOT] BETWEEN 1 AND 5

4) P.STATUS IS [NOT] NULL

5) P.PNAME [NOT] LIKE ‘%ov_’ [ESCAPE ‘c’]

Замечание. Предикат LIKE производит сравнение строки со строкой-шаблоном (“маской”). В **строке-шаблоне** разрешается использовать два трафаретных символа:

Символ подчеркивания "_" может использоваться вместо любого единичного символа в строке.

Символ процента "%" может заменять набор любых символов в строке (число символов в наборе может быть от 0 и более).

Основы языка SQL

Пусть **требуется найти строку, которая содержит ("%", "_") в качестве информационных символов**. Для этого с помощью **ключевого слова ESCAPE** нужно **определить** так называемый **escape-символ**, который, будучи поставленным перед символом "%" или "_", укажет, что этот символ является информационным.

Escape-символ не может быть символом "\" (обратная косая черта) и, вообще говоря, должен представлять собой символ, никогда не появляющийся в упоминаемом столбце как информационный символ.

Часто для этих целей используются символы "@" и "~".

Пример. Получить список поставщиков, в имени которых содержится "_" (знак подчеркивания).

```
SELECT pname
 FROM P
 WHERE pname LIKE ‘%@_%’ ESCAPE ‘@’;
```

Основы языка SQL

3)Выбрать некоторые колонки из исходной таблицы, **удалив из результата повторяющиеся строки** (ключевое слово DISTINCT):

```
SELECT DISTINCT P.NAME
  FROM P;
```

4) Использование **вычисляемых полей, скалярных выражений и переименований колонок** в запросах (ключевое слово AS…):

```
SELECT
    TOVAR.TNAME,
    TOVAR.KOL,
    TOVAR.PRICE,
    "=" AS EQU,
    TOVAR.KOL*TOVAR.PRICE AS SUMMA
  FROM TOVAR;
```

TNAME	KOL	PRICE	EQU	SUMMA
Болт	10	100	=	1000
Гайка	20	200	=	4000
Винт	30	300	=	9000

Основы языка SQL

5) Упорядочение результатов запроса (ключевое слово ORDER BY…):

```
SELECT PD.PNUM, PD.DNUM, PD.VOLUME
  FROM PD
  ORDER BY DNUM;
```

6) Упорядочение результатов запроса **по нескольким полям с возрастанием или убыванием** (ключевые слова ASC, DESC):

```
SELECT PD.PNUM,
    PD.DNUM,
    PD.VOLUME
  FROM PD
  ORDER BY
    DNUM ASC,
    VOLUME DESC;
```

PNUM	DNUM	VOLUME
3	1	1000
2	1	150
1	1	100
2	2	250
1	2	200
1	3	300

Основы языка SQL

7) Использование LIMIT [start,] length

- start - указывает, с какой позиции нужно выдавать найденные записи,
- length - количество записей.

Замечания

1. Если требуется выводить записи с начала таблицы, первый параметр можно опустить.

2. Записи нумеруются **с нуля**.

Пример. 1. Вывести из таблицы Поставщики 20 записей, начиная с 5-ой.

```
SELECT * FROM P LIMIT 5, 20;
```

2. Вывести из таблицы Поставщики **5 последних записей**.

```
SELECT * FROM P ORDER BY PNUM DESC LIMIT 0, 5;
```

или

```
SELECT * FROM P ORDER BY  PNUM DESC LIMIT 5;
```