xSQL(Structured Query Language)

Язык SQL был впервые разработан фирмой IBM и затем одобрен многими компаниями в качестве стандарта языка управления реляционными базами данных. Этот язык ориентирован на операции с данными, представленными в виде логической взаимосвязанной совокупности таблиц. Использование концепции операций, ориентированных на табличное представление данных, позволило создать компактный язык SQL с небольшим набором команд. Это дало возможность достаточно легко определять, выводить и отправлять информацию в БД, упростив программирование сложных запросов. Таблица в языке SQL представляет собой совокупность строк и столбцов. Кроме обычных таблиц, язык SQL позволяет создавать особый тип таблиц - выборки, представляющие собой подмножество строк и колонок из одной или нескольких таблиц. Часто выборку называют виртуальной таблицей, поскольку она фактически не содержит данных, а только позволяет их выводить. Данные в выборке отражают их реальное изменение в соответствующих таблицах, и, наоборот, изменение данных в выборке приводит к изменению этих данных в соответствующих таблицах.

Особенность команд языка SQL состоит в том, что они ориентированы в большей степени на конечный результат обработки этих данных, чем на процедуры этой обработки. Система определяет оптимальный путь для вывода данных. Такой подход принципиально отличается от других языков управления данными, требующих знаний структуры БД и последовательности операций, ведущих к выводу данных.

Полный набор языка SQL включает около 30 основных команд, обеспечивающих все операции, необходимые для определения таблиц, выборок, создания запросов и вывода, изменения, удаления и ввода информации.

Команды языка SQL легко использовать, так как в них осуществляются ссылки на одни и те же элементы - строки и колонки таблицы или выборки. Для создания прикладных программ в операционных средах язык SQL обычно используется вместе с языками программирования высокого уровня типа FORTRAN, COBOL, C. Эффективное выполнение команд языка SQL реализуется благодаря хранению специальной ссылочной информации на каждую таблицу и выборку. Эта информация содержится в файлах, называемых каталогами таблиц, которые формируются при создании БД командой SQL **CREATE DATABASE**.

Язык SQL оперирует с понятием БД, которая содержит всю информацию, необходимую для обработки данных в прикладных программах.

В состав БД SQL входят следующие компоненты:

- таблицы - основные структуры данных в БД;
- выборки - тип виртуальной таблицы, которая обеспечивает обработку, ввод и вывод определенных строк и колонок из одной или более таблиц;
- синонимы - альтернативные имена таблиц и выборок;

- индексы - файлы, присоединенные к таблицам для обеспечения более гибкого поиска данных и поддержки целостности БД;
- каталоги - множество таблиц в каждой БД, которые описывают БД и ее содержание;

Перед созданием и использованием таблиц SQL необходимо создать или активировать существующую БД. Каждая БД хранится в определенном каталоге вместе с таблицами и связанными файлами, включая каталоги таблиц. В каталоге таблиц содержится информация о базовых таблицах, выборках, индексах и синонимах, а также данных по защите информации и по операциям, выполненными над таблицами и другими файлами в этой БД.

Команды создания баз данных и таблиц SQL

В языке SQL имеется пять команд, связанных с созданием и использованием БД: **CREATE DATABASE, SHOW DATABASE, START DATABASE, STOP DATABASE, DROP DATABASE.**

Создание БД определяется следующей командой:

CREATE DATABASE [путь] имя БД,

которая открывает в текущем каталоге или в каталоге, заданном в команде, подкаталог с именем БД. В этом подкаталоге формируются подкаталоги таблиц, в которые помещаются данные о всех существующих в данном каталоге таблицах SQL.

Таким образом, БД SQL - это специально созданный подкаталог, в котором размещается каталог таблиц, созданных в этом подкаталоге, таблицы и другие связанные с ними файлы.

Имя БД должно быть уникальным. Если при открытии БД уже существует подкаталог с таким же именем, то подкаталог определяется как БД SQL и в нем размещаются все компоненты БД. В каждый текущий момент может быть открыта только одна БД. Поэтому если требуется открыть другую БД, надо предварительно закрыть текущую.

Закрытие БД выполняется командой **STOP DATABASE**. Активация(открытие) имеющейся БД осуществляется командой **START DATABASE[<имя БД>]**. При это БД, ранее открытые и не закрытие в момент активации другой БД, автоматически закрываются. После этого можно выполнять любые команды SQL по определению таблиц, поиску данных и тд. Для удаления БД используется команда DROP DATABASE <имя БД>. Эта команда удаляет заданную БД, а так же все ее компоненты(каталоги таблиц и т.д.). Удалить можно только закрытые БД.

Для просмотра всех созданных БД используется команда **SHOW DATABASE**, выводящая список имен БД в текущем каталоге.

Таблицы SQL

Таблицы представляют собой основной компонент БД SQL, в котором определяется структура базы данных. Структура таблицы задается командой **CREATE TABLE** и может быть впоследствии изменена(добавлением колонки) командой **ALTER TABLE**. Команды **INSERT, UPDATE и DELETE** позволяют добавлять, изменять или удалять данные из таблицы.

Для создания новой таблицы SQL и определения типов данных используется команда

CREATE TABLE <имя таблицы>

(<имя колонки> < тип данных>,

<имя колонки>, <тип данных>...);

Пример:

CREATE TABLE Upr

(Tab_no CHAR(4),

Fam CHAR(15),

Dat_rab DATE,

Adres CHAR(30),

Oklad DECIMAL(4,0),

Prem DECIMAL(4,0)

);

Типы данных, разрешенные для использования в таблицах SQL

SMALLINT - целые числа с 6 значащими цифрами, включая знак. Диапазон от -99 999 до 999 999

INTEGER - целые числа с 11 значащими цифрами, включая знак. Диапазон от -9 999 999 999 до 99 999 999 999.

DECIMAL(X,Y) - числа с фиксированной десятичной точкой с x(от 1 до 19) позициями(включая знак числа) и y(от 0 до 19) дробными позициями справа от точки.

NUMERIC(X,Y) - числа с фиксированной десятичной точкой.

FLOAT(X,Y) - числа с плавающей точкой.

CHAR(n) - строка из n(до 254) символов. В колонку с этим типом данных могут вводиться данные из символьных колонок, переменных и строк.

DATE - даты в формате, определенном командами **SET DATE** и **SET CENTURY**. По умолчанию mm/dd/yy.

LOGICAL - логические значение True или False.

Ввод, обновление и удаление данных, модификация таблиц и создание выборок

Для ввода новых строк данных используются команды INSERT и LOAD DATA. Ввод конкретных значений в заданные колонки выполняется командой:

INSERT INTO <имя таблицы> [(< список колонок >)] VALUES (< список значеий >);

Пример:

INSERT INTO Upr VALUES ("0112", " Петров", {20/10/94}, 300);

Если список колонок отсутствует, то значения вводятся по порядку во все колонки

Для вывода отобранных строк из существующей таблицы используется предложение **SELECT**:

INSERT INTO <имя таблицы> [(< список колонок >)] <предложение SELECT>;

Предложение **SELECT** представляет собой команду для извлечения данных из заданных колонок таблицы в соответствии с заданным условием:

SELECT < колонки > FROM < таблицы > [WHERE < условие >] ...;

Пример:

INSERT INTO Upr1

SELECT * FROM Upr2 WHERE Tab_no > 0112

Символ * означает, что в предложении **SELECT** выбираются все колонки таблицы.

Для изменения значений строк в таблице в соответствии с заданным условием используется команда

UPDATE < имя таблицы >

SET < имя колонки > = < выражение>

[WHERE< условие>];

Пример:

UPDATE Upr1

SET Prem = Prem * 1,3

WHERE Dat_rab < CTOD("01/01/93") ;

Удаление строк таблицы по заданному условию можно выполнить командой

DELETE FROM < имя таблицы > [WHERE < условие >];

Пример:

DELETE FROM Upr1 WHERE Fam = "Котов";

В языке SQL можно добавлять колонки в существующую таблицу, но нельзя из удалять или изменять. Для модернизации таблицы целесообразно создать новую таблицу и занести в нее строки из существующей таблицы. Добавление колонок в существующую таблицу выполняется командой

ALTER TABLE < имя таблицы > ADD (< имя колонки > < тип данных >)

Пример:

ALTER TABLE Upr1 ADD (Telefon CHAR(9))

Для создания выборки с заданным именем и именами колонок используется команда

CREATE VIEW < имя выборки > [(< список колонок выборки >)] AS < предложение SELECT > [WITH CHECK OPTION];

Пример:

CREATE VIEW Ord AS SELECT Tab_no, Fam, Oklad FROM Upr1 WHERE Address = "Ордынка";

Если список колонок выборки не указан, то имена выбираются из списка предложения **SELECT**, которое определяет строки и колонки из таблицы, включаемой в выборку.

Опция **WITH CHECK OPTION** используется для выборок, которые могут обновляться, и устанавливает, что вводимые или обновляемые строки должны проверяться на соответствие с заданным условием. Например, эта опция позволяет запретить ввод и обновление строк выборки, которые не удовлетворяют условию, заданному в предложении **SELECT**(с ключевым словом **WHERE**).

Выборка позволяет реконструировать данные в БД, не прибегая к созданию новых таблиц или модификации старых. Как уже говорилось, выборки фактически не содержат данных и поэтому занимают значительно меньше места в памяти, чем таблицы. Можно создать несколько выборок по одной таблице, отражающих все изменения, сделанные в таблицах.

Удаление выборки осуществляется командой

DROP VIEW < имя таблицы >;

В язык SQL включены синонимы, которые замещают имя таблицы или выборки в команде. Синоним можно создать командой

CREATE SYNONYM < имя синонима > FOR < имя таблицы или выборки>;

Одной из важнейших характеристик любой БД является способность быстро находить информацию. Индексные файлы представляют собой одно из средств для быстрого поиска данных в БД SQL в соответствии с их значениями колонок. Индексы для БД SQL представляют собой указатели в множественном индексном файле .indx, который имеет то же имя, что и таблицы SQL. Для каждой БД можно создать до 47 указателей индексов. Для создания индекса существует следующая команда:

CREATE [UNIQUE] INDEX < имя индекса > ON < имя таблицы > (< имя колонки > [ASC/DESC] ...);

Указатели индексов создаются для заданных имен колонок. Имена колонок могут занимать не более 100 позиций в строке команды.

Опция **UNIQUE** определяет уникальный индекс, то есть в индекс включаются уникальные значения колонок. При вводе или обновлении данных в таблицу с уникальным индексом записываются только уникальные значения.

Опции **ASC/DESC** определяют порядок построения индекса соответственно по возрастанию или убыванию.

Индексы можно создать по всем полям, кроме логических. Удаление индексов осуществляется командой **DROP INDEX < имя индекса >;**

Запросы на извлечение данных языка SQL

Большинство запросов на извлечение данных из БД SQL строится на основе команды **SELECT**. Синтаксис этой команды позволяет создавать простые по конструкции предложения для реализации сложных запросов на извлечение, ввод и удаление данных:

SELECT < предложение >

[INTO < предложение >]

FROM < предложение >

[WHERE < предложение >]

[GROUP BY < предложение >]

[HAVING < предложение >]

[UNION SELECT < предложение >] ....

[ORDER BY< предложение > / FOR UPDATE OF < предложение >]

[SAVE TO TEMP < предложение >];

Простейшая форма этой команды:

SELECT < колонки> FROM < таблицы>;

Для высвечивания всех колонок таблицы нужно в команде вместе имен колонок поместить символ *:

SELECT * FROM < таблицы>;

Для того, чтобы вывести только уникальные строки таблицы, не повторяющиеся в заданных колонках, используется ключевое слово **DISTINCT**

SELECT DISTINCT < колонки> FROM < таблицы>;

Пример:

SELECT DISTINCT Det_no, Opisanie FROM Detal;

Для отбора строк по заданному условию используется предложение **WHERE**:

SELECT < колонки> FROM < таблицы> WHERE < >;

Пример:

SELECT Det_no, Opisanie, Adres FROM Detal WHERE Adres = "Минск";

В SQL входят специальные функции агрегирования, которые используются в командах **SELECT**:

COUNT() - определяет количество отобранных строк;

SUM() - суммирует значение числовых строк;

MIN() - находит минимальное значение символьных, числовых и колонок типа дата;

MAX() - находит максимальное значение символьных, числовых и колонок типа дата;

AVG() - вычисляет среднее значение каждой колонки.

Функции агрегирования могут быть включены в команду **SELECT** вместо имен колонок

Пример:

SELECT COUNT(*) FRO Upr1;

SELECT SUM(Oklad) FROM Upr1;

SELECT MAX(Oklad), MIN(Oklad), AVG(Oklad) FROM Upr1;

Символ * в функции **COUNT**() указывает на подсчет по любой колонке.

В предложении **WHERE** могут использоваться предикаты **BETWEEN, IN, LIKE**, которые позволяют записывать условие в более компактном виде.

Предикат **BETWEEN** определяет диапазон поисковых значений.

Предикат **IN** проверяет, находится ли значение колонки в указанном списке.

Предикат **LIKE** осуществляет поиск на включение подстроки заданной в переменной или константе. Используется метасимвол - %.

Пример:

SELECT Tab_no, Fam, Oklad FROM Upr1 WHERE Fam LIKE "Пет% " ;