**62.​ Языки описания данных. Язык SQL. Построение запросов на извлечение данных. Объединение, группировка, сортировка данных в запросах. Использование критериев отбора. Вложенные запросы. Построение запросов на занесение, удаление и обновление данных. Построение запросов на создание и удаление таблиц.**

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.

Язык SQL был разработан фирмой IBM в конце 70-х годов. Первый международный стандарт языка был принят международной стандартизирующей организацией ISO в 1989 г., а новый (более полный) - в 1992 г. В настоящее время все производители реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL92.

Единственной структурой представления данных в реляционной БД является двумерная таблица. Любая таблица может рассматриваться как одна из форм представления теоретико-множественного понятия **отношение** (relation), отсюда название модели данных - реляционная.   
В реляционной модели данных таблица обладает следующими основными свойствами:

1. идентифицуруется уникальным именем;
2. имеет конечное ненулевое количество столбцов;
3. имеет конечное (возможно, нулевое) число строк;
4. столбцы таблицы идентифицируются своими уникальными именами и номерами;
5. содержимое всех ячеек столбца принадлежит одному типу данных (т.е. столбцы однородны), содержимым ячейки столбца не может быть таблица;
6. строки таблицы не имеют какой-либо упорядоченности и идентифицируются только своим содержимым (т.е. понятие номер строки не определено);
7. в общем случае ячейки таблицы могут оставаться пустыми (т.е. не содержать какого-либо значения), такое их состояние обозначается как NULL.

На содержимое таблиц можно накладывать **ограничения** в виде:

1. требования уникальности содержимого каждой ячейки какого-либо столбца и/или совокупности ячеек в строке, относящихся к нескольким столбцам;
2. запрета для какого-либо столбца (столбцов) иметь пустые (NULL) ячейки.

**Ключом** таблицы называется столбец или комбинация столбцов, содержимое ячеек которого(ых) используется для прямого доступа к строкам таблицы. Различают ключи первичный и вторичные. Первичный ключ уникален и однозначно идентифицирует строку таблицы. Столбец строки, определенный в качестве первичного ключа, не может содержать пустое (NULL) значение в какой-либо своей ячейке. Вторичный ключ определяет местоположение, в общем случае, не одной строки таблицы, а нескольких подобных (в любом случае ускоряя доступ к ним, хотя не в такой степени как ключ первичный).   
Ключи используются внутренними механизмами СУБД для оптимизации затрат на доступ к строкам таблиц (путем, например, их физического упорядочения по значениям ключей или построения двоичного дерева поиска).

Основными **операциями** над таблицами являются следующие.

1. **Проекция** - построение новой таблицы из исходной путем включения в нее избранных столбцов исходной таблицы.
2. **Ограничение** - построение новой таблицы из исходной путем включения в нее тех строк исходной таблицы, которые отвечают некоторому критерию в виде логического условия (ограничения).
3. **Объединение** - построение новой таблицы из 2-ух или более исходных путем включения в нее всех строк исходных таблиц (при условии, конечно, что они подобны).
4. **Декартово произведение** - построение новой таблицы из 2-ух или более исходных путем включения в нее строк, образованных всеми возможными вариантами конкатенации (слияния) строк исходных таблиц. Количество строк новой таблицы определяется как произведение количеств строк всех исходных таблиц.

Пречисленные выше 4 операции создают базис, на основе которого может быть построено большинство (но не все) практически полезных запросов на извлечение информации из реляционной БД.   
*Примечание*. Набор операций будет полным, если дополнить его операциями пересечения и вычитания.

Кроме перечисленных выше в языке SQL реализованы операции модификации содержимого строк таблицы и пополнения таблицы новыми строками (что теоретически может рассматриваться как операция объединения), а также операции управления таблицами.   
Рассмотренные выше операции над таблицами реляционной БД обладая функциональной полнотой, будучи реализованы на практике в своем чистом каноническом виде, как правило, крайне неэкономичны (в первую очередь это относится к комбинации операций ограничения и декартового произведения). Разработчики реальных реляционных СУБД прибегают ко всевозможным приемам и ухищрениям для минизации вычислительных затрат при выполнении этих операций. Общим способом, нашедшим отражение в языке SQL, повышения эффективности выполнения запросов в реляционных СУБД  являются импользование ключей индексов.

**Индексом** называется скрытая от пользователя вспомогательная управляющая структура, обеспечивающая прямой метод доступа к строкам таблицы, позволяющий исключить последовательный просмотр всех строк таблицы для обнаружения отвечающих некоторому критерию поиска. Индексы автоматически создаются для всех ключей таблицы + могут быть созданы вручную.

Популярные СУБД: Oracle, MS SQL, MySql, PostreSQL, SQLite

Далее все примеры кода приведены для MySQL

**Основы синтаксиса языка SQL**

Программа на языке SQL представляет собой простую линейную последовательность операторов языка SQL. Язык SQL в своем чистом виде операторов управления порядком выполнения запросов к БД (типа циклов, ветвлений, переходов) не имеет (однако они часто присутствуют в реализациях языка конкретными СУБД – например в Transact-SQL).

**Типы данных языка SQL**

К основным типам данных можно отнести:

INT, SMALLINT, FLOAT, CHAR, VARCHAR, BLOB, DATE

Отличие типов данных CHAR и VARCHAR заключается в том, что для хранения в таблице строк символов типа CHAR используется точно *size* байт (хотя содержание хранимых строк может быть значительно короче), в то время как для строк типа VARCHAR незанятые символами строк (пустые) байты в таблице не хранятся.   
Подчеркнем, что величины *len* и *dec* (в отличие от *size*) не влияют на размер хранения данных в таблице, а только форматируют вывод данных из таблицы.

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

**Манипулирование таблицами**

Для создания, изменения и удаления таблиц в SQL БД используются операторы CREATE TABLE, ALTER TABLE и DROP TABLE.

**Создание таблицы**

Создание таблицы в БД реализуется оператором CREATE TABLE, имеющим следующий синтаксис

**CREATE TABLE *имя\_табл* (*с\_спецификация*, ...);**

где *с\_спецификация* имеет разнообразный синтаксис. Здесь же рассматриваются наиболее часто используемые ее формы.

1. Описание столбца таблицы

***имя\_столбца тип\_данных* [NULL]**

где *имя\_столбца* -  имя столбца таблицы, а *тип\_данных* - спецификация одного из типов данных, рассмотренных в разделе [Типы данных языка SQL](http://www.opennet.ru/docs/RUS/sql/#DataTypes). Необязательное ключевое слово NULL означает, что ячейкам данного столбца разрешено быть пустыми (т.е. не содержать какого-либо значения).

1. Описание столбца таблицы

***имя\_столбца тип\_данных* NOT NULL [DEFAULT *по\_умолч*] [PRIMARY KEY]**

где конструкция NOT NULL запрещает иметь в таблице пустые ячейки в данном столбце. Конструкция PRIMARY KEY указывает, что содержимое столбца будет играть роль первичного ключа для создаваемой таблицы. Конструкция DEFAULT *по\_умолч* переопределяет имеющееся для столбцов каждого типа данных значение по умолчанию (например, 0 для числовых типов), используемое при добавлении в таблицу оператором [INSERT INTO](http://www.opennet.ru/docs/RUS/sql/#Insert) строк, не содержащих значений в этом столбце.

1. Описание первичного ключа

**PRIMARY KEY *имя\_ключа* (*имя\_столбца*, ...)**

Эта спецификация позволяет задать первичный ключ для таблицы в виде композиции содержимого нескольких столбцов.

1. Описание вторичного ключа

**KEY *имя\_ключа* (*имя\_столбца*, ...)**

**Модификация таблицы**

Модификация существующей таблицы в БД реализуется оператором ALTER TABLE, имеющим следующий синтаксис

**ALTER TABLE *имя\_табл м\_специкация* [,*м\_спецификация* ...]**

где *м\_спецификация* имеет различные формы. Ниже рассматриваюся наиболее часто используемые.

1. Добавление нового столбца

**ADD COLUMN *с\_спецификация***

где *с\_спецификация* - описание добавляемого столбца в том виде, как оно используется  для создания таблицы оператором [CREATE TABLE](http://www.opennet.ru/docs/RUS/sql/#CreateTable).

1. Удаление первичного ключа для таблицы

**DROP PRIMARY KEY**

1. Изменение/удаление значения по умолчанию

**ALTER COLUMN *имя\_столбца* SET *по\_умолч***

или

**ALTER COLUMN *имя\_столбца* DROP DEFAULT**

**Удаление таблицы**

Удаление одной или сразу нескольких таблиц из БД реализуется оператором DROP TABLE, имеющим следующий простой синтаксис

**DROP TABLE *имя\_табл,* ...**

Подчеркнем, что оператор DROP TABLE удаляет не только все содержимое таблицы, но и само описание таблицы из БД. Если требуется удалить только содержимое таблицы, то необходимо использовать оператор [DELETE FROM](http://www.opennet.ru/docs/RUS/sql/#DeleteFrom).

**Добавление строк в таблицу**

Для добавления строк в таблицу SQL базы данных используется оператор INSERT INTO. Основные его синтаксические формы описываются ниже.

1. Добавление строки перечислением значений всех ее ячеек

**INSERT INTO *имя\_табл* VALUES (*знач*, ...);**

где *знач* - константное значение ячейки строки. Значения ячеек в списке должны соответствовать порядку перечисления спецификаций столбцов таблицы в операторе [CREATE TABLE](http://www.opennet.ru/docs/RUS/sql/#CreateTable). Допустимо в качестве *знач* указывать ключевое слово NULL, что означает отсутствие значения для соответствующей ячейки строки.   
Перед добавлением новой строки в таблицу СУБД проверяет допустимость перечисленных значений, используя описание столбцов таблицы из оператора CREATE TABLE.

1. Добавление строки с использованием списка имен столбцов

**INSERT INTO *имя\_табл* (*имя\_столбца*, ...) VALUES (*знач*, ...);**

Здесь списки имен столбцов и значений ячеек добавляемой строки должны быть согласованы, хотя нет никаких требований к их порядку. Допустимо опускать в списках информацию о некоторых ячейках строки, при этом

* + ячейки, соответствующие столбцам со спецификацией NULL в операторе CREATE TABLE, будут пустыми;
  + ячейки, соответствующие столбцам со спецификацией NOT NULL в операторе CREATE TABLE, заполняются значениями по умолчанию.

1. Добавление строк по результатам запроса к БД

**INSERT INTO *имя\_табл* [(*имя\_столбца*, ...)] SELECT ...**

Такой оператор дает возможность добавить в таблицу 0, 1 или сразу несколько новых строк, полученных в результате запроса к базе данных, реализуемого оператором [SELECT](http://www.opennet.ru/docs/RUS/sql/#Select).

**Выборка данных из таблиц**

Для извлечения данных, содержащихся в таблицах SQL БД, используется оператор SELECT, имеющий в общем случае сложный и многовариантный синтаксис. В данном учебном пособии рассматриваются только несложные и наиболее часто используемые примеры конструкций оператора SELECT.   
Упрощенно оператор SELECT выглядит следующим образом:

**SELECT [ALL | DISTINCT] *в\_выражение*, ...**

**FROM *имя\_табл* [*син\_табл*], ...**

**[WHERE *сложн\_условие*]**

**[GROUP BY *полн\_имя\_столбца*|*ном\_столбца*, ...]**

**[ORDER BY *полн\_имя\_столбца*|*ном\_столбца* [ASC|DESC], ...]**

**[HAVING *сложн\_условие*];**

Результатом работы оператора является выводимая на стандартный вывод (экран дисплея) вновь построенная таблица, для которой

* количество и смысл (семантика) столбцов определяется списком элементов *в\_выражение*;
* содержимое строк определяется содержимым исходных таблиц из списка FROM и критерием выборки, задаваемым *сложн\_условие*.

При описании синтаксиса оператора SELECT использованы следующие обозначения:

* *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.

**Описание столбцов результирующей таблицы**

**1.** Специальным (и часто используемым) видом *в\_выражение* является символ \*, имеющий смысл все столбцы таблиц из списка FROM.

**2.** Простым (и также часто используемым) случаем *в\_выражение* является полное имя столбца одной из таблиц списка FROM.

**3.** В общем случае *в\_выражение* может представлять собой сложное скобочное выражение над содержимым столбцов таблицы, использующее арифметические, строковые, логические операции и функции.

**4.** В общем случае *в\_выражение* допускает использование агрегативных (называемых также групповыми) функций, принимающих в качестве своего единственного аргумента значения всех ячеек указанного столбца результирующей таблицы.

**Описание критерия выборки содержимого строк результирующей матрицы**

В качестве критерия выбора информации из таблиц списка FROM оператора SELECT выступает *сложн\_условие*, записываемое после ключевого слова WHERE и имеющее следующий вид:

***прост\_условие***

**или**

***прост\_условие* AND *сложн\_условие***

**или**

***прост\_условие* OR *сложн\_условие***

Типичными вариантами *прост\_условие* являются следующие.

* Сравнение

***полн\_имя\_столбца @ полн\_имя\_столбца\_или\_константа***

где *@* - один из операторов сравнения: (больше), (меньше), = ( не меньше), = (не больше), = (равно), (не равно), а *полное\_имя\_столбца* - имя столбца, конкретизированное при необходимости именем или синонимом имени таблицы, как это было описано [выше](http://www.opennet.ru/docs/RUS/sql/#FullName).

* Сопоставление с образцом

***полн\_имя\_столбца* [NOT] LIKE *образец***

где *образец* имеет вид, описанный в [таблице 3](http://www.opennet.ru/docs/RUS/sql/#Table3).

* Проверка на пустое значение в ячейке стодбца

***полн\_имя\_столбца* IS [NOT] NULL**

При конструировании *сложн\_условие* допустимо использование  круглых скобок для управления порядком вычисления условий.

**Упорядочивание и группирование строк результирующей таблицы**

Для обеспечения структурированности в расположении строк результирующей таблицы в операторе SELECT используются конструкции GROUP BY и ORDER BY.

* Упорядочение строк достигается перечислением полных имен столбцов, по которым в возрастающем (ASC) или убывающем (DESC) порядке сортируются строки результирующей таблицы. При этом строки упорядочиваются в первую очередь по столбцу, указанному первым в списке ORDER BY. Затем, если среди значений ячеек первого столбца есть повторяющиеся, производится упорядочение по второму столбцу и так далее.

* Оператор SELECT может обеспечить вычисление [агрегативных функций](http://www.opennet.ru/docs/RUS/sql/#Agregate) для групп строк результирующей таблицы. Для этого используется список полных имен столбцов в конструкции GROUP BY. Первое полное имя столбца в списке GROUP BY используется для разбиения строк результирующей таблицы на первичные группы, первичные группы разделяются на подгруппы вторым в списке полным именем столбца и так далее.

Оператор SELECT выводит значения агрегативных функций для самых малых подгрупп.

**Выборка из нескольких таблиц**

В общем случае оператор SELECT языка SQL дает возможность выборки информации сразу из нескольких таблиц, перечисленных в списке FROM. На концептуальном уровне рассмотрения (уровне реляционной модели данных) такая выборка включает в себя два основных этапа:

* + построение промежуточной таблицы, представляющей собой декартово произведение таблиц из списка FROM (т.е. таблицы, строки которой представляют собой все возможные сочетания строк исходных таблиц);
  + копирование в результирующую таблицу всех строк промежуточной,

**Манипулирование строками таблиц**

Для удаления и изменения строк таблиц SQL БД применяются операторы DELETE и UPDATE.

**Удаление строк**

Удаление строк таблицы реализуется оператором DELETE FROM, имеющим следующий синтаксис

**DELETE FROM *имя\_табл* [WHERE *сложн\_условие*]**

где *сложн\_условие* имеет описанный выше [синтаксис](http://www.opennet.ru/docs/RUS/sql/#SelectWhere). В результате выполнения оператора из таблицы удаляются все строки, удовлетворяющие критерию *сложн\_условие*. Если в операторе DELETE FROM конструкция WHERE опущена, то удаляются все строки таблицы.

**Модификация строк**

Изменение содержимого строк таблицы реализуется оператором UPDATE, имеющим следующий синтаксис

**UPDATE *имя\_табл* SET *имя\_столбца*=*выражение*, ...**

**[WHERE *сложн\_условие*]**

где *выражение* - выражение (в простейшем случае - константа), согласующееся по результату с типом данных столбца. В *выражение* допустимо использование значений ячеек любых столбцов таблицы, рассмотренных ранее операций и функций (но не агрегативных), а также прежнего содержимого модифицуруемой ячейки. Обновлению подлежат столбцы строк, отвечающих критерию *сложн\_условие*. Если конструкция WHERE в операторе отсутствует, то обновляются все строки таблицы.