Задание

Повторить теоретический материал, кратко ответить на вопросы:

Создание базы данных.

Создание, удаление таблицы и изменение определения таблицы.

Индексы.

Правила выполнения однотабличных запросов.

Многотабличные запросы (соединения).

С уважением, Хвастова Светлана Ивановна

!!! Если возникнут вопросы обращаться по телефону 0721389311.

Электронная почта: xvsviv@rambler.ru

Лекция на тему: «MySQL»

Цель: закрепить знания по SQL.

Типы данных SQL

Типы данных SQL разделяются на три группы:
- строковые;
- с плавающей точкой (дробные числа);
- целые числа, дата и время.

Типы данных SQL строковые

Типы данных SQL	**Описание**
CHAR(size)	Строки фиксированной длиной (могут содержать буквы, цифры и специальные символы). Фиксированный размер указан в скобках. Можно записать до 255 символов
VARCHAR(size)	Может хранить не более 255 символов.
TINYTEXT	Может хранить не более 255 символов.
TEXT	Может хранить не более 65 535 символов.
BLOB	Может хранить не более 65 535 символов.
MEDIUMTEXT	Может хранить не более 16 777 215 символов.
MEDIUMBLOB	Может хранить не более 16 777 215 символов.
LONGTEXT	Может хранить не более 4 294 967 295 символов.
LONGBLOB	Может хранить не более 4 294 967 295 символов.
ENUM(x,y,z,etc.)	Позволяет вводить список допустимых значений. Можно ввести до 65535 значений в **SQL Тип данных ENUM** список. Если при вставке значения не будет присутствовать в списке *ENUM*, то мы получим пустое значение. Ввести возможные значения можно в таком формате: ENUM ('X', 'Y', 'Z')
SET	**SQL Тип данных SET** напоминает *ENUM* за исключением того, что *SET* может содержать до 64 значений.

Типы данных SQL с плавающей точкой (дробные числа) и целые числа

Типы данных SQL	**Описание**
TINYINT(size)	Может хранить числа от -128 до 127
SMALLINT(size)	Диапазон от -32 768 до 32 767
MEDIUMINT(size)	Диапазон от -8 388 608 до 8 388 607
INT(size)	Диапазон от -2 147 483 648 до 2 147 483 647
BIGINT(size)	Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
FLOAT(size,d)	Число с плавающей точкой небольшой точности.
DOUBLE(size,d)	Число с плавающей точкой двойной точности.
DECIMAL(size,d)	Дробное число, хранящееся в виде строки.

Типы данных SQL — Дата и время

Типы данных SQL	**Описание**
DATE()	Дата в формате ГГГГ-ММ-ДД
DATETIME()	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP()	Дата и время в формате timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIME()	Время в формате ЧЧ:ММ:СС
YEAR()	Год в двух значной или в четырехзначном формате.

Типы данных MySQL

Типы данных MySQL разделяются на следующие типы:

Числовые типы данных

Типы данных	**Байт**	**От**	**До**
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

Типы данных даты и времени

Типы данных	**Значение «Ноль»**
DATETIME	’0000-00-00 00:00:00′
DATE	’0000-00-00′
TIMESTAMP	00000000000000 (длина зависит от количества выводимых символов)
TIME	’00:00:00′
YEAR	0000

Символьные Типы данных

Типы данных	**Макс. размер**	**Байт**
TINYTEXT или TINYBLOB	2^8-1	255
TEXT или BLOB	2^16-1 (64K-1)	65535

MEDIUMTEXT или MEDIUMBLOB	2^24-1 (16M-1)	16777215
LONGBLOB	2^32-1 (4G-1)	4294967295

SQL ALTER TABLE

Команда **ALTER TABLE** используется для **добавления**, **удаления** или **модификации** колонки в уже существующей **таблице**.

Команда ALTER TABLE

Команда **ALTER TABLE** изменяет определение таблицы одним из следующих способов:

добавляет столбец

добавляет ограничение целостности

переопределяет столбец (тип данных, размер, умалчиваемое значение)

удаляет столбец

модифицирует характеристики памяти или иные параметры

включает, выключает или удаляет ограничение целостности или триггер.

Условие: Таблица должна быть в схеме пользователя, или пользователь должен иметь системную привилегию **ALTER ANY TABLE**.

Добавляя столбец с ограничением **NOT NULL**, разработчик или администратор **БД** должны учесть ряд обстоятельств. Сначала нужно создать столбец без ограничения, а затем ввести значения во все его строки. После того как все значения столбца станут не **NULL**-значениями, к нему можно применить ограничение **NOT NULL**. Если столбец с ограничением **NOT NULL** пытается добавить пользователь, возвращается сообщение об ошибке, говорящее о том, что либо таблица должна быть пустой, либо в столбце должны содержаться значения для каждой существующей строки (напомним, что после наложения на столбец ограничения **NOT NULL** в нем не могут присутствовать **NULL**-значения ни в одной из существующих строк.

Изменяя типы данных существующих столбцов или добавляя столбцы в таблицу базы данных, нужно соблюдать ряд условий. Общепринято, что

увеличение – это хорошо, а уменьшение, как правило, — не очень. Допустимые увеличения:

Увеличение размера столбца **CHAR** или **VARCHAR2**

Увеличение размера столбца **NUMBER**

Добавление новых столбцов в таблицу

Уменьшение различных характеристик таблицы, в том числе некоторых типов данных столбцов и реального числа столбцов таблицы, требует особых действий. Часто перед внесением изменения нужно убедиться в том, что в соответствующем столбце или столбцах все значения являются **NULL**-значениями. Для выполнения подобных операций над столбцами таблицы, содержащими данные, разработчик должен найти или создать какую-то область для временного хранения этих данных. Например, создать таблицу с помощью команды **CREATE TABLE AS SELECT**, в которой извлекаются данные из первичного ключа и изменяемого столбца или столбцов. Допустимые изменения:

Уменьшение размера столбца **NUMBER** (только при пустом столбце для всех строк)

Уменьшение размера столбца **CHAR** или **VARCHAR2** (только при пустом столбце для всех строк)

Изменение типа данных столбца (только при пустом столбце для всех строк)

ALTER TABLE Пример 1

Добавление столбца в таблицу:

ALTER TABLE t1 **ADD** (pole1 char(10));

ALTER TABLE Пример 2

Изменение размера столбца таблицы:

ALTER TABLE t1 **MODIFY** (pole1 char(20));

ALTER TABLE Пример 3

Удаление столбца таблицы:

ALTER TABLE t1 **DROP COLUMN** pole1;

С помощью команды ALTER TABLE можно изменить имя таблицы без реального переноса физической информации в БД:

ALTER TABLE t1 **RENAME TO** t2;

Аналогичную операцию можно выполнить с помощью команды RENAME:

RENAME t1 **TO** t2;

Ограничения целостности столбцов и таблиц БД можно изменять, а также запрещать, разрешать и удалять. Это дает разработчику возможность создавать, модифицировать и удалять бизнес-правила, ограничивающие данные. Рассмотрим добавление ограничений в БД. Простота или сложность этого процесса зависит от определенных обстоятельств. Если вместе с БД создать ограничение нельзя, проще всего добавить его перед вводом данных:

ALTER TABLE Пример 4

Модификация структуры таблицы

ALTER TABLE t1 **MODIFY** (pole1 **NOT NULL**);

CREATE TABLE t2

(pole1 CHAR(10) **PRIMARY KEY**);

ALTER TABLE t1 **ADD**

(**CONSTRAINT** fk_t1 **FOREIGN KEY** (pole1)

REFERENCES t2 (pole1));

ALTER TABLE t1 **ADD** (**UNIQUE** (p_name));

ALTER TABLE t1 **ADD** (p_size CHAR(4) **CHECK**

(p_size IN (‘P’,’S’,’M’,’L’,’XL’,’XXL’,’XXXL’)));

В первой из приведенных выше команд для добавления ограничения NOT NULL для столбца используется конструкция MODIFY, а для добавления всех табличных ограничений целостности других типов – конструкция ADD. Столбец, для которого добавляется ограничение, должен уже существовать в таблице БД; в противном случае создать ограничение не удастся.

ALTER TABLE Пример 5

Для добавления ограничений целостности можно не указывать имя создаваемого ограничения с помощью ключевого слова **CONSTRAINT**. В этом случае команда будет выглядеть следующим образом:

ALTER TABLE t1 ADD FOREIGN KEY (pole1) REFERENCES t2 (pole1);

Существует ряд условий создания ограничений:

Первичные ключи: в столбцах не могут содержаться NULL-значения, и все значения должны быть уникальны.

Внешние ключи: в тех столбцах других таблиц, на которые производятся ссылки, должны содержаться значения, соответствующие всем значениям ссылающихся столбцов, либо значения этих последних должны быть NULL-значениями.

Ограничения UNIQUE: все значения столбцов должны быть уникальными или NULL-значениями.

Ограничения CHECK: новое ограничение будет применяться только по отношению к данным, добавляемым или модифицируемым после его создания.

NOT NULL: NULL-значения в столбцах запрещены.

Ограничения можно разрешать и запрещать. Разрешенное ограничение выполняет свои функции, реализуя бизнес-правила по отношению к вводимым в таблицу данным, а запрещенное ограничение переводится в разряд недействующих, как если бы оно было удалено, и его правила не реализуются.

ALTER TABLE Пример 6

Запрещение ограничений:

ALTER TABLE t1 **DISABLE PRIMARY KEY**;
ALTER TABLE t1 **DISABLE UNIQUE** (p_name);

ALTER TABLE Пример 7

В некоторых случаях запрещение первичного ключа, от которого зависят внешние ключи, может вызвать определенные сложности, например:

ALTER TABLE t2 **DISABLE PRIMARY KEY;**

Error at line 1: **Cannot disable constraint …. – dependencies exist** (невозможно запретить ограничение – существуют зависимости)

Для удаления первичного ключа при наличии зависящих от него внешних ключей в команде ALTER TABLE DISABLE <ограничения> обязательна конструкция CASCADE:

ALTER TABLE t2 **DISABLE PRIMARY KEY CASCADE;**

ALTER TABLE Пример 8

Запрещенное ограничение разрешается следующим образом:

ALTER TABLE t1 **ENABLE PRIMARY KEY**;

ALTER TABLE t1 **ENABLE UNIQUE** (p_name);

Разрешить можно только те ограничения, которые были установлены ранее, а в данный момент запрещены.

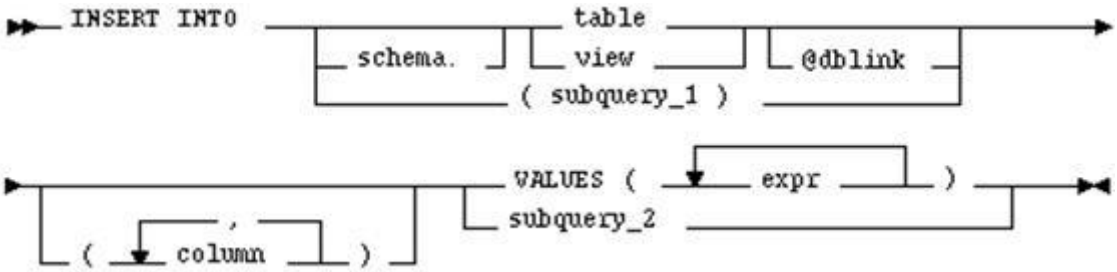
Ограничение, процесс создания которого завершился неудачей, не будет существовать в запрещенном виде, ожидая своего разрешения после устранения ошибки. Как правило, владелец таблицы или тот, кому предоставлены соответствующие права, может удалить ограничение:

ALTER TABLE t1 **DROP UNIQUE** (p_name);

SQL INSERT

Команда INSERT добавляет строки в таблицу или представление основной таблицы.

Синтаксис команды Sql INSERT



Синтаксис команды Insert

Основные ключевые слова и параметры команды INSERT

schema — идентификатор полномочий, обычно совпадающий с именем некоторого пользователя

table view — имя таблицы, в которую строки должны быть вставлены; если указано представление, то строки вставляются в основную таблицу представления

subquery_1 — подзапрос, который сервер обрабатывает тем же самым способом как представление

column — столбец таблицы или представления, в который для каждой вставленной строки вводится значение из фразы **VALUES** или подзапроса; если один из столбцов таблицы опускается из этого списка, значением столбца для вставленной строки является значение по умолчанию столбца, определенное при создании таблицы. Если полностью опускается список столбца, предложение **VALUES** или запрос должен определить значения для всех столбцов в таблице

VALUES — определяет строку значений, которые будут вставлены в таблицу или представление; значение должно быть определено в предложении **VALUES** для каждого столбца в списке столбцов

subquery_2 — подзапрос, который возвращает строки, вставляемые в таблицу; выборочный список этого подзапроса должен иметь такое же количество столбцов, как в списке столбцов утверждения **INSERT**

Утверждение **INSERT** с фразой **VALUES** добавляет одиночную строку к таблице. Эта строка содержит значения, определенные фразой **VALUES**. Утверждение **INSERT** с **подзапросом** вместо фразы **VALUES** добавляет к таблице все строки, возвращенные **подзапросом**. Сервер обрабатывает **подзапрос** и вставляет каждую возвращенную строку в таблицу. Если подзапрос не выбирает никакие строки, сервер не вставляет никакие строки в таблицу.

Подзапрос может обратиться к любой таблице или представлению, включая целевую таблицу утверждения **INSERT**. Сервер назначает значения полям в новых строках, основанных на внутренней позиции столбцов в таблице и

порядке значений фразы **VALUES** или в списке выбора запроса. Если какие-либо столбцы пропущены в списке столбцов, сервер назначает им значения по умолчанию, определенные при создании таблицы. Если любой из этих столбцов имеет **NOT NULL** ограничение то сервер возвращает ошибку, указывающую, что ограничение было нарушено и отменяет утверждение **INSERT**.

При выдаче утверждения **INSERT** включается любой **INSERT** — триггер, определенный на **таблице**.

INSERT MySQL

Для **вставки** новых строк в базу данных **MySQL** используется **команда INSERT**, примеры команды **INSERT** приведены ниже: **INSERT INTO Пример 1.**

Вставка новой строки в таблицу table_name.

INSERT INTO table_name VALUES (’1′,’165′,’0′,’name’);

INSERT INTO Пример 2.

Вставка новой строки в таблицу table_name с указанием вставки данных в нужные нам колонки.

INSERT INTO table_name VALUES (’1′,’165′,’0′,’name’);

В базе данных **MySQL** имеется возможность вставлять множество новых строк, используя одну команду **INSERT**.

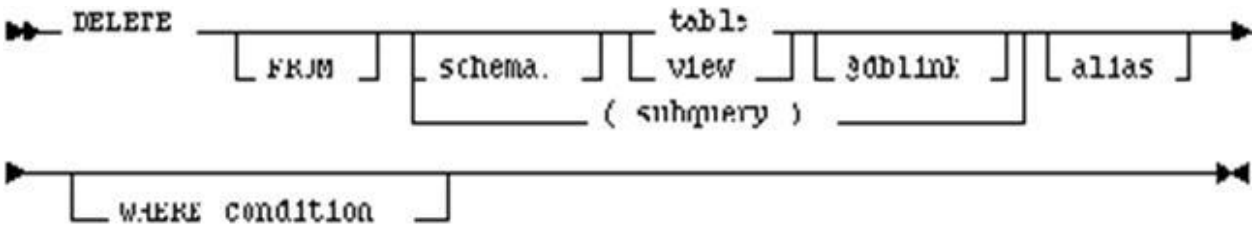
INSERT INTO Пример 3.

Вставка несколько строк в таблицу table_name.

INSERT INTO table_name (tbl_id, chislo, chislotwo, name) VALUES (’1′,’159′,’34′,’name1′), (’2′,’14′,’61′,’name2′), (’3′,’356′,’8′,’name3′);

SQL DELETE

Команда DELETE удаляет строки из таблицы или представления основной таблицы базы данных, например, в **MySQL, Oracle**.



Синтаксис команды DELETE

Команда DELETE. Основные ключевые слова и параметры команды DELETE

schema — идентификатор полномочий, обычно совпадающий с именем некоторого пользователя

table view — имя таблицы, из которой удаляются строки; если определяется представление, сервер удаляет строки из основной таблицы представления

subquery — **подзапрос**, с помощью которого выбираются строки для удаления; сервер выполняет **подзапрос** и использует строки его результата как таблицу фразы FROM

WHERE — удаляет только строки, которые удовлетворяют условию; условие может ссылаться на таблицу и содержать **подзапрос**.

При выдаче утверждения **DELETE** включается любой **DELETE**-триггер, определенный на таблице.

Команда DELETE Пример №1

Удаление всех строк из таблицы:

DELETE FROM temp_assign;

В данном примере **команда DELETE** удаляет все строки без исключения.

Команда DELETE Пример №2.

Удаляет из таблицы всех продавцов, у которых комиссионные меньше 100 у.е. в месяц:

DELETE FROM emp WHERE JOB = ‘SALESMAN’ AND COMM < 100;

В данном примере **команда DELETE** удаляет все строки, которые попадают под условие JOB = ‘SALESMAN’ AND COMM < 100;

Команда DELETE Пример №3

Предыдущий пример можно записать по-другому:

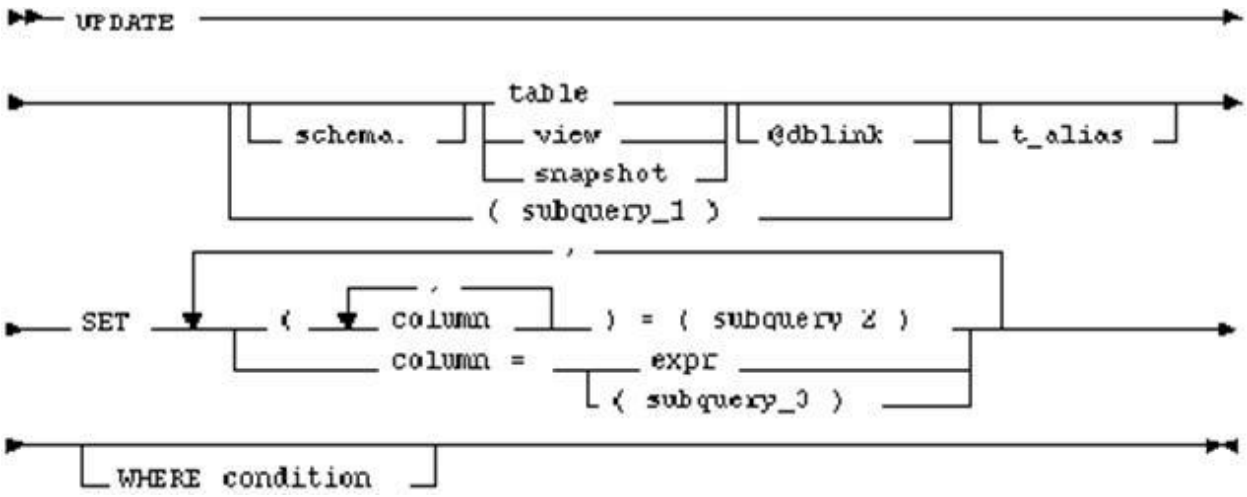
DELETE FROM (select * from emp) WHERE job = ‘SALESMAN’ AND comm < 100;

Для удаления всех записей в MySQL можно использовать следующую команду **DELETE**:

DELETE * FROM table_nam;

SQL UPDATE

Команда UPDATE — производит изменения в уже существующей записи или во множестве записей в таблице **SQL**. Изменяет существующие значения в таблице или в основной таблице представления.



Синтаксис команды UPDATE

Команда UPDATE. Основные ключевые слова и параметры команды UPDATE

schema — идентификатор полномочий, обычно совпадающий с именем некоторого пользователя

table view — имя таблицы **SQL**, в которой изменяются данные; если определяется представление, данные изменяются в основной таблице **SQL** представления

subquery_1 — подзапрос, который сервер обрабатывает тем же самым способом как представление

column — столбец таблицы **SQL** или представления **SQL**, значение которого изменяется; если столбец таблицы из предложения **SET** опускается, значение столбца остается неизменяемым

expr — новое значение, назначаемое соответствующему столбцу; это выражение может содержать главные переменные и необязательные индикаторные переменные

subquery_2 — новое значение, назначаемое соответствующему столбцу

subquery_3 — новое значение, назначаемое соответствующему столбцу

WHERE — определяет диапазон изменяемых строк теми, для которых определенное условие является **TRUE**; если опускается эта фраза, модифицируются все строки в таблице или представлении. При выдаче утверждения **UPDATE** включается любой **UPDATE-триггер**, определенный на таблице.

Подзапросы. Если предложение **SET** содержит **подзапрос**, он возвращает точно одну строку для каждой модифицируемой строки. Каждое значение в результате подзапроса назначается соответствующим столбцам списка в круглых скобках. Если подзапрос не возвращает никакие строки, столбцу назначается **NULL**. **Подзапросы** могут выбирать данные из модифицируемой таблицы. Предложение **SET** может совмещать выражения и **подзапросы**.

Команда UPDATE Пример 1

Изменение для всех покупателей рейтинга на значение, равное 200:

UPDATE Customers **SET** rating = 200;

Команда UPDATE Пример 2

Замена значения столбца во всех строках таблицы, как правило, используется редко. Поэтому в команде **UPDATE**, как и в команде **DELETE**, можно использовать предикат. Для выполнения указанной замены значений столбца rating, для всех покупателей, которые обслуживаются продавцом Giovanni (snum = 1003), следует ввести:

UPDATE Customers **SET** rating = 200 WHERE snum = 1001;

Команда SQL UPDATE Пример 3

В предложении **SET** можно указать любое количество значений для столбцов, разделенных запятыми:

UPDATE emp **SET** job = ‘MANAGER’, sal = sal + 1000, deptno = 20 WHERE ename = ‘JONES’;

Команда UPDATE Пример 4

В предложении **SET** можно указать значение NULL без использования какого-либо специального синтаксиса (например, такого как IS NULL). Таким образом, если нужно установить все рейтинги покупателей из Лондона (city = ‘London’) равными NULL-значению, необходимо ввести:

UPDATE Customers **SET** rating = NULL WHERE city = ‘London’;

Команда UPDATE Пример 5

Поясняет использование следующих синтаксических конструкций команды **UPDATE**:

Обе формы предложения **SET** вместе в одном утверждении.

Подзапрос.

Предложение WHERE, ограничивающее диапазон модифицируемых строк.

UPDATE emp a **SET** deptno =

(**SELECT** deptno FROM dept WHERE loc = ‘BOSTON’), (sal, comm) = (**SELECT** 1.1*AVG(sal), 1.5*AVG(comm) FROM emp b WHERE a.deptno = b.deptno) WHERE deptno IN (**SELECT** deptno FROM dept WHERE loc = ‘DALLAS’ OR loc = ‘DETROIT’);

Вышеупомянутое утверждение **UPDATE** выполняет следующие операции:

Модифицирует только тех служащих, кто работают в Dallas или Detroit

Устанавливает значение колонки deptno для служащих из Бостона

Устанавливает жалованье каждого служащего в 1.1 раз больше среднего жалованья всего отдела

Устанавливает комиссионные каждого служащего в 1.5 раза больше средних комиссионных всего отдела

SQL SELECT

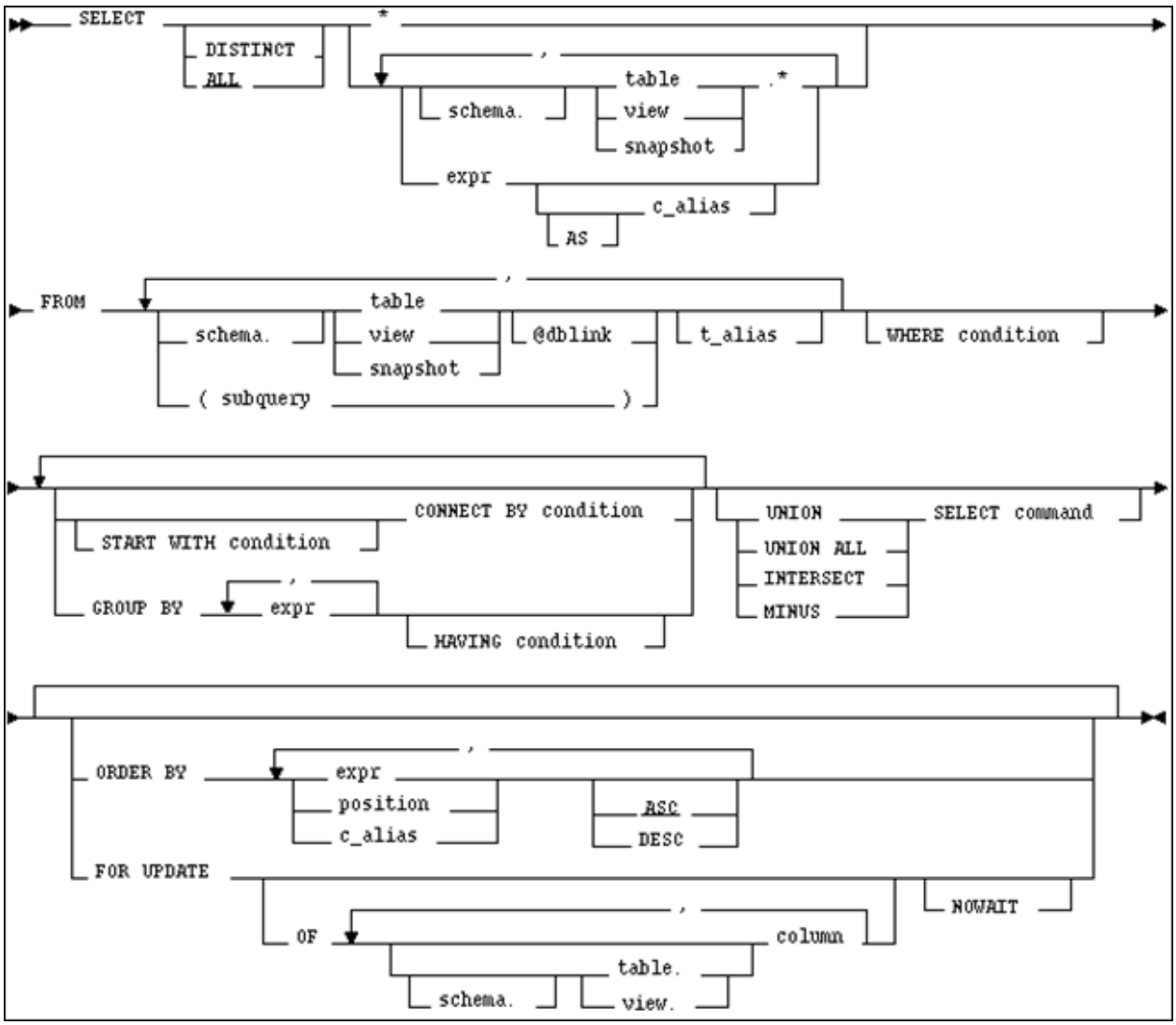
Команда SELECT производит выборку данных из таблиц по запросу.

Язык **SQL** допускает три типа синтаксических конструкций, начинающихся с ключевого слова **SELECT**:

оператор выборки (**select statement**)

спецификация курсора (**cursor specification**)

подзапрос (**subquery**).



Синтаксис команды SELECT

Основные ключевые слова и параметры команды SELECT в MySQL

DISTINCT — возвращает только одно значение для каждого набора одинаковых выбранных значений столбца

ALL — возвращает все выбранные строки, включая все повторяющиеся значения столбцов (принимается по умолчанию)

***** — выбирает все столбцы из всех таблиц или представлений, перечисленных после оператора **FROM**

schema — идентификатор полномочий, обычно совпадающий с именем некоторого пользователя

table.* view.* — выбирает все столбцы из указанной таблицы, представления

Expr — извлекает из таблицы (представления) некоторое определяемое выражение

table view — имя таблицы(представления), из которой происходит выборка данных.

subquery — подзапрос, который сервер обрабатывает тем же самым способом как представление.

WHERE — ограничивает множество строк выборкой тех записей, для которых условие является истинным; если это предложение опускается, сервер возвращает все строки из таблиц.

GROUP BY — группирует выбранные строки по группам строк с одинаковым значением указанных полей и возвращает одиночную строку итоговой информации для каждой группы.

HAVING — ограничивает выбираемые группы строк такими группами, для которых определяемое условие является истинным; если это предложение опускается, сервер возвращает строки всех групп.

UNION UNION ALL INTERSECT MINUS — объединяет строки, возвращенные двумя утверждениями **SELECT** с использованием операции пересечения множеств; для ссылки на столбец вводится псевдоним для его обозначения; предложение **FOR UPDATE** не может использоваться с этими операторами

ORDER BY — упорядочивает строки, возвращенные запросом.

Expr— значение выражения определяет правило упорядочивания строк.

ASC DESC — определяет порядок вывода данных (по возрастанию или по убыванию); значением по умолчанию является **ASC**.

FOR UPDATE — блокирует выбранные строки.

OF — блокирует выбираемые строки для специфической таблицы в **объединении**.

NOWAIT — возвращает управление пользователю, если команда **SELECT** пытается блокировать строку, которая уже блокирована другим

пользователем; если это предложение опускается, сервер ждет, пока строка не станет доступной и только тогда возвращает результаты команды **SELECT**.

Описание команды SELECT

Основой всех синтаксических конструкций, начинающихся с ключевого слова **SELECT**, является синтаксическая конструкция “табличное выражение”.

Семантика табличного выражения состоит в том, что на основе последовательного применения разделов **FROM**, **WHERE**, **GROUP BY** и **HAVING** из заданных в разделе **FROM** таблиц строится некоторая новая результирующая таблица, порядок следования строк которой не определен и среди строк которой могут находиться дубликаты (т.е. в общем случае таблица-результат табличного выражения является мультимножеством строк).

Наиболее общей является конструкция “спецификация курсора”. **Курсор** — это понятие **языка SQL**, позволяющее с помощью набора специальных операторов получить построчный доступ к результату запроса к **БД**. К табличным выражениям, участвующим в спецификации курсора, не предъявляются какие- либо ограничения. При определении спецификации курсора используются три дополнительных конструкции: спецификация запроса, выражение запросов и раздел **ORDER BY**.

В спецификации запроса задается список выборки (список арифметических выражений над значениями столбцов результата табличного выражения и констант). В результате применения списка выборки к результату табличного выражения производится построение новой таблицы, содержащей то же число строк, но вообще говоря другое число столбцов, содержащих результаты вычисления соответствующих арифметических выражений из списка выборки.

Выражение запросов — это выражение, строящееся по указанным синтаксическим правилам на основе спецификаций запросов. Единственной операцией, которую разрешается использовать в выражениях запросов,

является операция **UNION** (объединение таблиц) с возможной разновидностью **UNION ALL**.

Оператор выборки — это отдельный оператор **языка SQL**, позволяющий получить результат запроса в прикладной программе без привлечения курсора. Поэтому оператор выборки имеет синтаксис, отличающийся от синтаксиса спецификации курсора, и при его выполнении возникают ограничения на результат табличного выражения. Фактически, и то, и другое диктуется спецификой оператора выборки как одиночного оператора **SQL**: при его выполнении результат должен быть помещен в переменные прикладной программы. Поэтому в операторе появляется раздел **INTO**, содержащий список переменных прикладной программы, и возникает то ограничение, что результирующая таблица должна содержать не более одной строки.

Подзапрос — **запрос**, который может входить в предикат условия выборки оператора **SQL**.

Для выполнения следующих **SQL** запросов **SELECT** нам необходимо прежде всего изучить структуру таблиц.

Имя таблицы	**Имя поля**	**Тип поля**	**Примечание**
FAKULTET	KOD_F	Integer	PRIMARY KEY
	NAZV_F	Char, 30	
SPEC	KOD_S	Integer	PRIMARY KEY
	KOD_F	Integer	
	NAZV_S	Char, 50	
STUDENT	KOD_STUD	Integer	PRIMARY KEY
	KOD_S	Integer	
	FAM	Char, 30	
	IM	Char, 15	
	OT	Char, 15	
	STIP	Decimal, 3	
	BALL	Decimal, 3	

SQL SELECT Пример №1

Выбрать студентов, получающих стипендию, равную 150.

SELECT fname FROM STUDENT WHERE STIP=150;

С помощью данного **SQL** запроса **SELECT** выбираются все значения из таблицы STUDENT, поле STIP которых строго равно 150.

SQL SELECT Пример №2

Выбрать студентов, имеющих балл от 82 до 90. Студенты должны быть отсортированы в порядке убывания балла.

SELECT FAM *FROM* STUDENT *WHERE* BALL *BETWEEN* 81 AND 91 *ORDER BY* BALL *DESC*;

Как видно из **SQL** примера, чтобы выбрать студентов, которые имеют балл от 82 до 90, мы используем условие *BETWEEN*. Чтобы отсортировать в убывающем порядке *DESC*.

SQL SELECT. Пример №3

Выбрать студентов, фамилии которых начинаются с буквы «А».

SELECT FAM *FROM* STUDENT *WHERE* FAM *LIKE* 'A%';

Для того, чтобы выбрать фамилии, начинающиеся с буквы «А», мы используем оператор **SQL** *LIKE* для поиска значений по образцу.

SQL SELECT Пример №4

Подсчитать средний балл на каждом факультете.

SELECT NAZV_F *As* Название, *ROUND*(*AVG*(BALL), 2) As СредБалл *FROM* FAKULTET, SPEC, STUDENT *WHERE* STUDENT.KOD_S=SPEC.KOD_S *AND* SPEC.KOD_F=FAKULTET.KOD_F *GROUP BY* NAZV_F;

Пример запроса **SQL SELECT** показывает нам использование функции **SQL** *AVG* для вычисления среднего значения, *ROUND* для округления значения, раздела *GROUP BY* для группировки столбцов.

SQL SELECT. Пример №5.

Подсчитать количество студентов, обучающихся на каждом факультете. Вывести в запросе название факультета, комментарий – «обучается», количество студентов, комментарий «человек».

SELECT NAZV_F||’ обучается ‘||*COUNT*(STUDENT.BALL)||’ человек’ *As* CountStudOnFakultet *FROM* FAKULTET, SPEC, STUDENT *WHERE* STUDENT.KOD_S=SPEC.KOD_S *AND* SPEC.KOD_F=FAKULTET.KOD_F *GROUP BY* NAZV_F;

SQL SELECT. Пример №6.

Упорядочить студентов по факультетам, специальностям, фамилиям.

SELECT NAZV_F, NAZV_S, FAM FROM FAKULTET, SPEC, STUDENT WHERE STUDENT.KOD_S=SPEC.KOD_S AND SPEC.KOD_F=FAKULTET.KOD_F *ORDER BY* NAZV_F, NAZV_S, FAM;

SQL SELECT. Пример №7.

Определить, кто учится на специальности, к которой относится студент «Асанов».

SELECT FAM FROM STUDENT WHERE STUDENT.KOD_S=(**SELECT** KOD_S FROM STUDENT WHERE FAM=’Асанов’);

В данном **SQL** примере мы используем подзапрос **SQL SELECT**, который возвращает код специальности, на которой учится студент по фамилии Асанов.

SQL SELECT. Пример №8.

Показать, какие специальности встречаются в таблице STUDENT. Дубликаты исключить. Вывести в запросе названия специальностей.

SELECT DISTINCT NAZV_S FROM SPEC, STUDENT WHERE STUDENT.KOD_S=SPEC.KOD_S;

Здесь мы с помощью **SQL** ограничения *DISTINCT* выводим только различные значения.

SQL SELECT. Пример №9.

Извлечь из базы данных все данные по сотрудникам, принятым на работу после 01.01.1980 г. в формате "Сотрудник < фамилия сотрудника и его инициалы> принят на работу < дата принятия на работу>".

SELECT CONCAT(CONCAT(CONCAT('Сотрудник ', sname), CONCAT(SUBSTR(fname, 0, 1), SUBSTR(otch, 0, 1))), CONCAT('принят на работу ', acceptdate)) FROM employees WHERE acceptdate > to_date('01.01.80','dd.mm.yyyy');

В данном **SQL SELECT**, используя **SQL** функцию *CONCAT* мы выводим все поля таблицы в одну строчку. **SQL** функция to_date возвращает привычное для СУБД значение даты.

SQL SELECT. Пример №10.

Извлечь из базы данных перечень должностей, которые имеют сотрудники следующих отделов: 'БИОТЕХНОЛОГИЙ', 'ИНЖЕНЕРНОЙ ЭКОЛОГИИ'. В запросе использовать названия отделов.

SELECT pname FROM posts, departments, employees WHERE posts.pid = employees.pid AND employees.did = departments.did AND (departments.dname = 'БИОТЕХНОЛОГИЙ' OR departments.dname = 'ИНЖЕНЕРНОЙ ЭКОЛОГИИ');

Пояснение: posts — таблица должностей, departments — таблица отделов, employees — таблица сотрудников, pname — название должности.

SQL SELECT. Пример №11.

Извлечь из базы данных значение максимального личного шифра и фамилию сотрудника с этим номером в формате "Максимальный личный шифр < значение шифра> имеет сотрудник < фамилия сотрудника и его инициалы>".

SELECT ‘Максимальный личный шифр ‘||eid||’ имеет сотрудник ‘||sname||’ ‘||SUBSTR(fname, 0, 1)||’. ‘||SUBSTR(otch, 0, 1)||’.’ As Максимальный_личный_шифр FROM employees WHERE eid = (**SELECT** *MAX*(eid) from employees);

Для вывода максимального личного шифра мы устанавливаем условие в WHERE так, чтобы шифр был равен полученному максимальному шифру из подзапроса **SELECT**, используя функцию MAX.

SQL SELECT. Пример №12.

Получить из базы данных значение числа записей в таблице данных о сотрудниках в формате “Таблица данных о сотрудниках содержит < число записей> записей”.

SELECT ‘Таблица данных о сотрудниках содержит ‘||*COUNT*(*)||’ записей’ FROM employees;

Используя **SQL** функцию COUNT, выводим количество записей таблицы employees.

SQL SELECT. Пример №13.

Получить единым запросом список отделов и должностей предприятия.

SELECT pname FROM posts UNION **SELECT** dname FROM departments;

С помощью UNION мы объединяем два запроса **SQL SELECT** и выводим их как один.

SQL SELECT. Пример №14.

Вывести 30 комментариев начиная с 5 комментария из таблицы replies, кроме комментариев автора ‘Вася’. Данные сортируются по дате добавления комментария в убывающем порядке.

SELECT * FROM replies WHERE author!=’Вася’ ORDER BY date DESC LIMIT 5, 30;

SQL SELECT. Пример №15.

*Получить из **SQL** таблицы news одну новость с пометкой «Важные новости».*

SELECT * FROM news WHERE status=’Важные новости’ LIMIT 1;

В данном **SQL** примере мы выбираем все столбцы поля, у которого столбец status равен ‘Важные новости’. **SQL** ограничение *LIMIT 1* означает, что выбираем только одну запись.

SQL SELECT. Пример №16.

Получить имя письма с идентификатором 1565.

SELECT name FROM mail_inbox WHERE id=’1565′;

SQL SELECT. Пример №17.

Получить название рекламного пакета с идентификатором 24.

SELECT title FROM ad_packages WHERE id=’24′;

SQL SELECT. Пример №18.

Вывести столбцы id, title, price, c_type с сортировкой по идентификатору в возрастающем порядке.

SELECT id, title, price, c_type FROM ad_packages ORDER BY id ASC;

SQL SELECT. Пример №19.

*Вывести все записи из **SQL** таблицы actions с идентификатором 1234567890.*

SELECT * actions WHERE uid=’1234567890′;

SQL SELECT. Пример №20.

SELECT * FROM buypts ORDER BY c_type DESC, price DESC;

SQL SELECT. Пример №21.

SELECT uid FROM refs WHERE rid=’19′;

SQL SELECT. Пример №22.

SELECT * FROM sellpts ORDER BY price ASC;

SQL SELECT. Пример №23.

SELECT * FROM useronline WHERE uid=’1′;

SQL SELECT. Пример №24.

SELECT * FROM mail_inbox WHERE uid=’4590′ AND status=’unread’;

SQL SELECT. Пример №25.

SELECT * FROM buyref WHERE rid!=’5′ ORDER BY dateStamp DESC;

SQL SELECT. Пример №26.

SELECT id FROM replies WHERE nid=’5′;

SQL SELECT. Пример №27.

SELECT id, dateStamp, title, text FROM news WHERE dateStamp=’1232342412′;

SQL SELECT. Пример №28.

SELECT id, dateStamp, author, text, remote_addr FROM replies WHERE nid=’45′ ORDER BY dateStamp ASC;