Основы SQL

Создание таблиц и связей

```
CREATE TABLE titles
(id int(10) not null);
```
- создать таблицу titles

```
DROP TABLE titles;
```
- удалить таблицу titles

При создании таблицы

для каждого столбца можно указать:

1) Имя столбца
2) Тип данных
3) Нулевое или ненулевое значение
4) Первичный ключ
5) Значение по умолчанию
6) Ограничения
7) Внешний ключ

Имя столбца

Имя столбца должно состоять из одного слова и желательно на английском языке

Тип данных

Целые числа

- **TINYINT** Может хранить числа от -128 до 127
- **SMALLINT** Диапазон от -32 768 до 32 767
- **MEDIUMINT** Диапазон от -8 388 608 до 8 388 607
- **INT** Диапазон от -2 147 483 648 до 2 147 483 647
- **BIGINT** Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

Дробные числа

- **DECIMALS** - количество знаков после десятичной точки, которые будут учитываться.
- **UNSIGNED** - задает беззнаковые числа.
- **FLOAT** Число с плавающей точкой небольшой точности.
- **DOUBLE** Число с плавающей точкой двойной точности.
- **REAL** Синоним для DOUBLE.
- **DECIMAL** Дробное число, хранящееся в виде строки.
- **NUMERIC** Синоним для DECIMAL.

Строки

- **VARCHAR** Может хранить не более 255 символов.
- **TEXT** Может хранить не более 65 535 символов.
- **MEDIUMTEXT** Может хранить не более 16 777 215 символов.
- **LONGTEXT** Может хранить не более 4 294 967 295 символов.

Бинарные данные

- **TINYBLOB** Может хранить не более 255 символов.
- **BLOB** Может хранить не более 65 535 символов.
- **MEDIUMBLOB** Может хранить не более 16 777 215 символов.
- **LONGBLOB** Может хранить не более 4 294 967 295 символов.
- **BLOD**-данные не перекодируются автоматически, если при работе с установленным соединением включена возможность перекодирования текста "на лету".

Дата и время

- **DATE** Дата в формате ГГГГ-ММ-ДД
- **TIME** Время в формате ЧЧ:ММ:СС
- **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 ГГГГММДДЧЧММСС, что сильно умаляет преимущества его использования в **PHP**

Нулевое или ненулевое значение

Если столбец может принимать нулевые значения, то пишется `NULL`, в противном случае пишется `NOT NULL`. По умолчанию (если ничего не писать) устанавливается значение `NULL`.

Первичный ключ

Для того, чтобы указать, что столбец является первичным ключом после пишется `primary key`. Если ключ составной, то primary key пишется для каждого столбца, входящего в его состав.

Значение по умолчанию

Для задания значения по умолчанию используется оператор default, после которого ставится значение, которое будет значением по умолчанию.

Авто инкремент

Для автоматического увеличения значения аргумента используется оператор `auto_increment`.

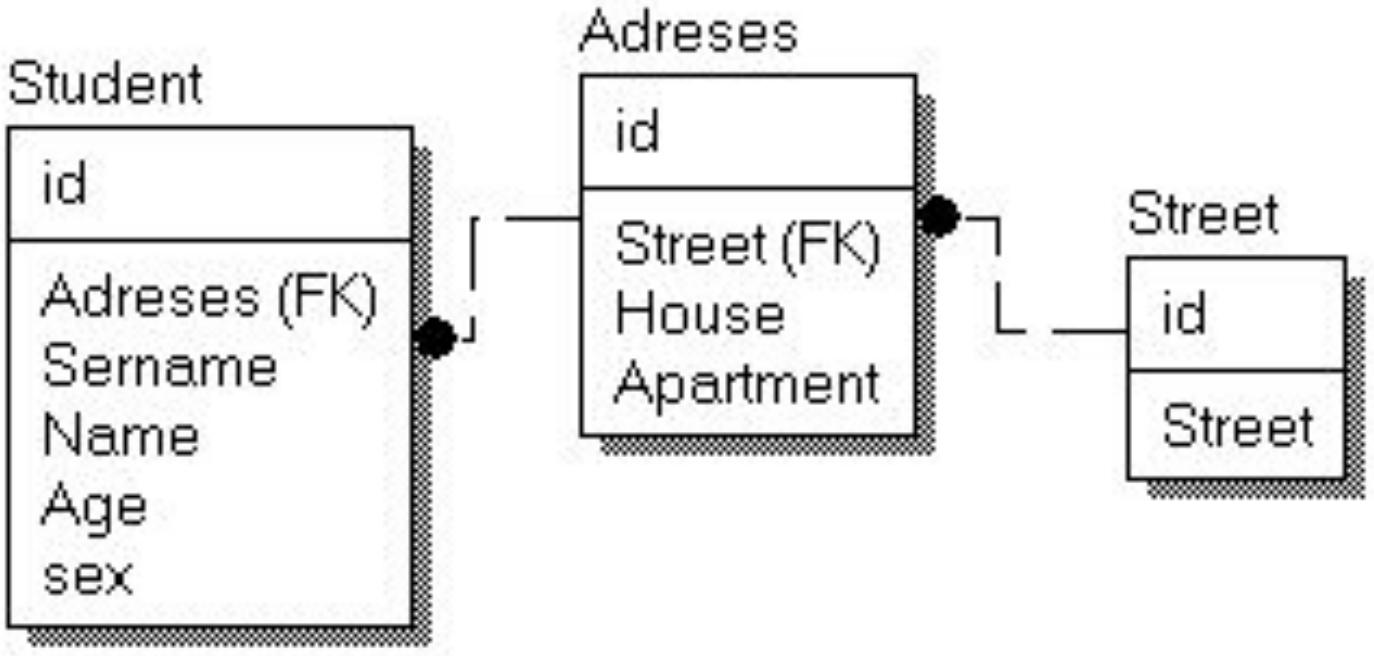
Ограничения

Для задания ограничений на значения столбца используется оператор check().
В скобках записываются ограничения в виде как записываются ограничения в where при запросе select, т.е. можно указывать in, like, between и пр.
Для задания ограничений сразу нескольких столбцов они записываются в конце запроса CREATE TABLE

Внешний ключ

Для того, чтобы указать, что столбец является внешним ключом пишется references и имя таблицы, на которую ссылается внешний ключ, а затем в круглых скобках имя столбца в этой таблице, на которую он ссылается (обычно первичный ключ таблицы).

Если внешний ключ создается после описания атрибутов, а также создается составной первичный ключ, то пишется foreign key после чего записывается имя столбца-первичного ключа, а после этого references и имя таблицы, на которую ссылается внешний ключ



Фамилия должна быть по умолчанию Ivanov

Возраст – от 17 до 60

Пол – только мужской или женский

```
CREATE TABLE Street
(id int(11) not null primary key
auto_increment,
street varchar(25) not null);

CREATE TABLE Adreses
(id int(11) not null primary key
auto_increment,
street_id int(11) not null,
house integer,
apartment int(11),
foreign key Adreses(street_id)
    references Street(id));
```

CHECK(MAX_SALARY<=25000)

```
CREATE TABLE Student
(id int(11) not null primary key
auto_increment,
adreses_id int(11) not null,
sername varchar(25) default 'Иванов',
name varchar(25),
age int(11) check(Age>=17 and
Age<=60),
sex varchar(5),
foreign key Student(adreses_id)
    references Adreses(id));
```

INSERT

INSERT INTO имя_таблицы
[(столбец1 [, столбец2][1,…,n])]
VALUES
(константа1 [, константа2][1,…,n]);

Схема запроса на вставку данных

Указание столбцов необходимо для:

1) добавления данных в той последовательности, в какой перечислены столбцы;
2) добавления строк с пустыми полями.

INSERT

```
INSERT INTO Street VALUES (1, 'пр. Ленина');
INSERT INTO Street (street) VALUES ('ул. Красногвардейцев');
INSERT INTO Street (street) VALUES ('ул. Иркутский тракт’);

INSERT INTO Adreses VALUES (1, 2, 10, 53);
INSERT INTO Adreses VALUES (2, 1, 2, 8);
INSERT INTO Adreses (id, street_id, house) VALUES (3, 3, 5);
INSERT INTO Adreses (street_id, house) VALUES (2, 10);

INSERT INTO Student VALUES (1, 3, 'Пупкин' , 'Иван', 30, 'Man');
INSERT INTO Student (adreses_id, sername, age) VALUES (2,
'Суворов', 60);
INSERT INTO Student (adreses_id, sername, name, age) VALUES
(1, 'Суворов' , 'Александр', 56);
INSERT INTO Student (adreses_id, name, age, sex) VALUES (4,
'Эмануил', 45, 'Man');
```

SELECT

SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;
DISTINCT – отбрасываются дублирующие записи и выполняется сортировка;
FROM – определяются имена используемых таблиц;
WHERE – выполняется фильтрация строк объекта в соответствии с заданными условиями;
ORDER BY – определяется упорядоченность результатов выполнения операторов.
GROUP BY – образуются группы строк, имеющие одно и то же значение в указанном столбце;
HAVING – фильтруются группы строк объекта в соответствии с указанным условием;

SELECT

SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;

```
SELECT sername FROM Student;
SELECT * FROM Student
SELECT sername, name, age FROM Student
```

DISTINCT

DISTINCT – отбрасываются дублирующие записи и выполняется сортировка;

```
SELECT distinct sername FROM Student
SELECT distinct sername, name Desc FROM Student
```

WHERE

Существует пять основных типов условий поиска (или предикатов):

1) сравнение,
2) диапазон,
3) принадлежность множеству,
4) соответствие шаблону,
5) значение NULL.

WHERE

1) сравнение - сравниваются результаты вычисления одного выражения с результатами вычисления другого

Операторы сравнения:
= равенство;
< меньше;
> больше;
<= меньше или равно;
>= больше или равно;
<> не равно.

```
SELECT *
FROM Student
WHERE age>50;
```

```
SELECT *
FROM Student
WHERE age<=45;
```

WHERE

Более сложные запросы могут быть построены с помощью логических операторов AND, OR или NOT, а также скобок, используемых для определения порядка вычисления выражения.

```
SELECT * FROM Student
WHERE age>=30 AND age<=55;
```

```
SELECT * FROM Adreses
WHERE house<4 OR house >8;
```

WHERE

2) диапазон - проверяется, попадает ли результат вычисления выражения в заданный диапазон значений

Оператор BETWEEN используется для поиска значения внутри некоторого интервала

```
SELECT * FROM Student
WHERE age
BETWEEN 40 AND 55;

SELECT * FROM Student
WHERE age
NOT BETWEEN 40 AND 55;
```

WHERE

3) принадлежность множеству - проверяется, принадлежит ли результат вычислений выражения заданному множеству значений.

Оператор IN используется для сравнения некоторого значения со списком заданных значений

```
SELECT * FROM Street
WHERE street
IN ("пр. Ленина",
"ул. Иркутский тракт");
```

```
SELECT * FROM Student
WHERE name
IN (“Иван", "Александр");
```

WHERE

4) соответствие шаблону - проверяется, отвечает ли некоторое строковое значение заданному шаблону.

С помощью оператора LIKE можно выполнять сравнение выражения с заданным шаблоном, в котором допускается использование символов-заменителей:

% любое количество символов.

_ один символ строки.

WHERE

```
SELECT * FROM Student
WHERE sername Like "Су%";

SELECT * FROM Student
WHERE name Like "%ан";

SELECT * FROM Student
WHERE name Like "%ан%";

SELECT * FROM Student
WHERE age Like "_0";
```

WHERE

5) Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

Оператор IS NULL используется для сравнения текущего значения со значением NULL:

```
SELECT * FROM Student
WHERE name IS NULL;

SELECT * FROM Student
WHERE sex IS NOT NULL;

SELECT * FROM Adreses
WHERE apartment IS NULL;
```

ORDER BY

ORDER BY сортирует данные выходного набора в заданной последовательности. Сортировка по возрастанию задается ключевым словом **ASC**. Сортировка в обратной последовательности задается ключевым словом **DESC**.

```
SELECT *
FROM Student
ORDER BY sername DESC, name ASC;
```

Агрегирующие функции

Count (Выражение) - определяет количество записей в выходном наборе SQL-запроса;
Min/Max (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;
Avg (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.
Sum (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.

```
SELECT COUNT(*) AS COUNT
FROM Street;

SELECT MAX(Age) AS MaxAge
FROM Student;

SELECT AVG(Age) AS
Средний_возраст
FROM Student;
```

GROUP BY

GROUP BY группирует одинаковые строки

```
SELECT Sername FROM Student
GROUP BY Sername;

SELECT Sername, Count(Age) AS Количество
FROM Student
GROUP BY Sername;

SELECT Sername, AVG(Age) AS СрВозраст
FROM Student
GROUP BY Sername;
```

HAVING

HAVING аналогичен WHERE, но:

1) HAVING используется именно для группировки (вместе с GROUP BY);
2) WHERE выполняется до группировки, HAVING – после;
3) в HAVING можно использовать агрегирующие функции, в WHERE – нельзя;

HAVING

```
SELECT Sername,
Max(Age) AS МаксимальныйВозраст,
Min(Age) AS МинимальныйВозраст,
Count(Age) AS Количество
FROM Student
GROUP BY Sername
HAVING Count(Age)>1;
```

UPDATE

```
UPDATE имя_таблицы
SET имя_столбца = выражение
[WHERE условие];
```

Схема запроса на изменение данных

```
UPDATE Student
SET sex = 'Man';
```

```
UPDATE Student
SET name = 'Владимир'
WHERE sername = 'Суворов' AND age = 60;
```

DELETE

DELETE FROM имя_таблицы
WHERE условие;

Схема запроса на удаление данных

DELETE FROM Student
DELETE FROM Adreses
WHERE house = 10;

DROP

DROP TABLE имя_таблицы;

Схема запроса на удаление таблицы

DROP TABLE Student;
DROP TABLE Adreses;