1. СУБД MySQL
2. Язык обработки данных SQL

Сервер данных MySQL и его возможности

MySQL — это популярный сервер данных, применяемый при создании Web-сайтов.

- MySQL — весьма быстрый и нетребовательный к ресурсам компьютера сервер данных.
- Возможностей MySQL вполне хватает для создания Web-сайтов.
- MySQL распространяется бесплатно, более того — его исходные тексты открыты для изучения и доработки.
- MySQL прекрасно работает в связке с PHP, технологии создания активных серверных Web-страниц.

MySQL поддерживает запросы SQL, одновременный доступ нескольких пользователей к базам данных, индексы, права, множество типов данных и пр.

Можно давать отдельные права на выполнение разных видов запросов SQL.

Атрибут прав SELECT дает пользователю возможность извлекать данные из таблицы.

Права на добавление записей дает атрибут INSERT, на изменение — UPDATE, а на удаление — DELETE.

Также можно дать права на создание, изменение и удаление таблиц и индексов, выполнение служебных операций и пр.

Можно также задать **интернет-адрес** компьютера, с которого данный пользователь может подключаться к серверу.

Фактически интернет-адрес компьютера в MySQL является частью имени пользователя, которое в этом случае записывается вот так:

<имя пользователя>@<интернет-адрес компьютера>

то есть как адрес электронной почты.

Например:

root@localhost

Пользователь root имеет право подключаться к серверу данных только с локального компьютера (localhost).

remote_user@dev.domain.ru

Пользователь remote_user может подключиться к серверу только с компьютера dev.domain.ru и ни с какого другого (даже локального).

Если нужно дать пользователю возможность подключаться с любого компьютера, нужно будет вместо интернет-адреса подставить шаблон %, задающий любой интернет-адрес.

Например:

travelling_user@%

Пользователь travelling_user может подключаться к серверу с любого компьютера — и локального, и удаленного.

Шаблон % можно использовать и вместо имени пользователя; тогда он будет задавать любого пользователя.

Так, если написать

localhost

то с локального компьютера к серверу сможет подключиться любой пользователь (с любым именем, даже если оно явно не записано в списке пользователей).

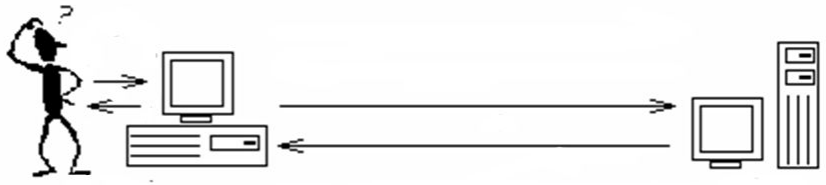
А если записать

%@%

то к серверу сможет подключиться любой пользователь с любого компьютера (такому пользователю будет нужно дать минимальные права).

Схема взаимодействия клиента данных с сервером :

1. Клиент данных формирует запрос на языке SQL.
2. Клиент данных передает сформированный запрос клиентской части сервера данных, установленной на клиентском компьютере.
3. Клиентская часть "упаковывает" принятый запрос в сетевые пакеты и передает его серверу данных.
4. Сервер данных принимает запрос, расшифровывает его, выполняет и отсылает результат обратно.
5. Клиентская часть сервера данных принимает результат, "распаковывает" его и возвращает клиенту данных.
6. Клиент данных принимает результат и выводит его на экран либо предпринимает какие-то действия (например, сообщает пользователю об ошибке).



Язык обработки данных

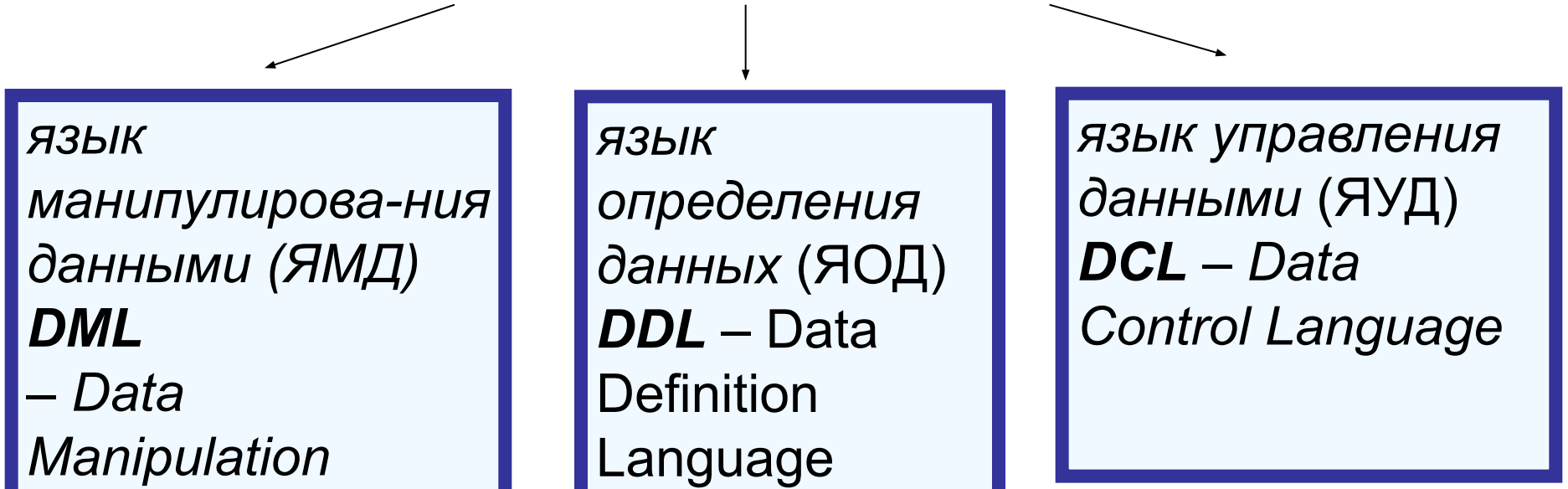
SQL

SQL - информационно-логический язык, предназначенный для описания хранимых данных, для извлечения хранимых данных и для модификации данных.

Запросы SQL можно разделить на три группы:

- **Запросы управления данными** (запросы выборки данных, добавления, изменения и удаления записей).
- **Запросы определения данных** (запросы создания, изменения и удаления баз данных, таблиц, индексов, связей и пр.).
- **Служебные запросы**. Выполняют различные технические задачи: сбор статистики использования баз данных, резервное копирование и пр.

Компоненты SQL



Создание БД

Create - позволяет создавать базы данных и таблицы

CREATE mydb;

- создание пустой БД mydb

Создание таблиц

Базовые таблицы создаются с помощью предложения **CREATE TABLE**:

CREATE TABLE имя_таблицы
(описание_поля_1,
[описание_поля_2]
[,…]);

Описание поля таблицы

имя_поля тип_данных [ограничения]

где

имя_поля – имя поля (столбца) таблицы;

тип_данных – спецификация одного из типов данных

Ограничения:

- **NOT NULL** – запрещает пустые ячейки в данном поле
- **DEFAULT по_умолч** – определяет значение по умолчанию
- **UNIQUE** – значение в поле должно быть уникальным
- **PRIMARY KEY** – указывает, что поле является первичным ключом
- **UNSIGNED** - запрещает числовым полям принимать отрицательные значения
- **AUTO_INCREMENT** - превращает обычное целочисленное поле в поле счетчика
- и др.

Некоторые типы данных, поддерживаемые MySQL

Тип данных	Обозначение в MySQL	Примечание
Строковый	VARCHAR	Строки длиной от 1 до 255 символов. Длина строки задается при создании поля
Целочисленный	SMALLINT	Целые числа от -32768 до 32767 или от 0 до 65535
	MEDIUMINT	Целые числа от -8388608 до 8388607 или от 0 до 16777215
	INT	Целые числа от -2147483648 до 2147483647 или от 0 до 4294967295
	BIGINT	Целые числа от -9223372036854775808 до 9223372036854775807 или от 0 до 18446744073709551615
С плавающей точкой	FLOAT	Число с плавающей точкой от -3,402823466*E38 до -1,175494351 *10E-38, 0 и от 1,1 75494351*10'38 до 3,402823466*1038
	DOUBLE	Число с плавающей точкой от -1,7976931348623157*10308 до -2,2250738585072014*10'308, 0 и от 2,2250738585072014*10'308 до 1,7976931348623157*10308

Некоторые типы данных, поддерживаемые MySQL (продолжение)

Логический	BOOL	Логическая величина "истина" (true) или "ложь" (false)
Дата	DATE	Значения даты от 01.01.1000 до 31.12.9999
	DATETIME	Объединенные значения даты от 01.01.1000 00:00:00 до 31.12.9999 23:59:59
Memo	BLOB	Строки переменной длины. Могут вмещать до 65535 символов
	MEDIUMBLOB	Строки переменной длины. Могут вмещать до 16777215 символов
	LONGBLOB	Строки переменной длины. Могут вмещать до 4294967295 символов

Описание таблицы items

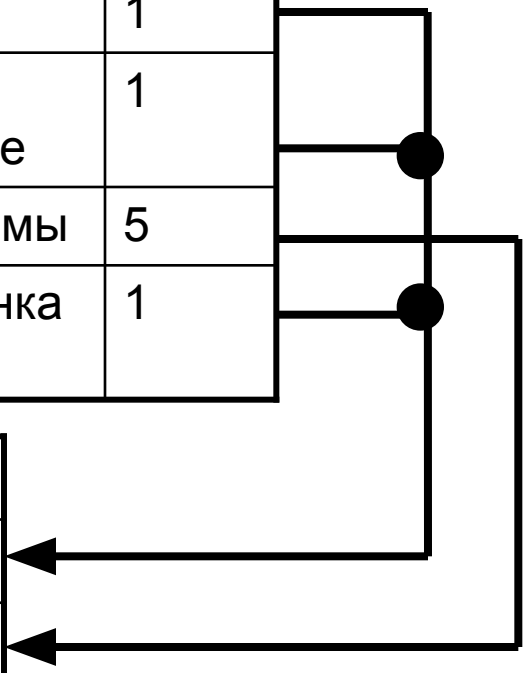
Имя поля	Описание поля	Тип данных поля	Описание типа данных
id	Идентификатор записи	SMALLINT	Целые числа от 0 до 65535
		UNSIGNED	Статей и файлов у нас будет немного, так что этого должно хватить
author	Автор статьи или файла	VARCHAR(30)	Строка фиксированной длины 30 символов
name	Название статьи или файла	VARCHAR(60)	Строка фиксированной длины 60 символов
added	Дата добавления статьи или файла в список	DATE	Дата
href	Интернет-адрес файла или статьи	VARCHAR(255)	Строка фиксированной длины 255 символов — именно такова максимальная длина интернет-адреса
catid	Ссылка на категорию (значение поля id соответствующей записи таблицы categories)	SMALLINT	Целые числа от 0 до 65535
		UNSIGNED	

Описание таблицы categories

Имя поля	Описание поля	Тип данных поля	Описание типа данных
id	Идентификатор записи	INT UNSIGNED	Целые числа от 0 до 65535.Категорий у нас будет немного,так что этого должно хватить
name	Название категории	VARCHAR(20)	Строка фиксированной длины 20 символов
file	Если "истина", то запись обозначает категорию файлов, в противном случае — категорию статей	BOOL	Логические величины

Таблица items

date	author	name	catid
12.08.2009	Кирсанов, Д.	Цвет в веб-дизайне	1
29.07.2009	Семенов И.	К вопросу об эффективности поиска конкретики в Интернете	1
30.10.2004	Олифер, Н.	Сетевые операционные системы	5
10.02.2010	Андреев А.	Исследование активности рынка интернет-ссылок в Рунете	1



name	file	catid
Интернет	да	1
Система	нет	5
Офис	да	2
Программирование	да	3
Интернет	нет	4

Таблица categories

```
CREATE TABLE categories
( id smallint UNSIGNED
  AUTO_INCREMENT ,
  name varchar( 15 ) NOT NULL ,
  file bool NOT NULL ,
  PRIMARY KEY ( id ) )
```

```
CREATE TABLE items
(id smallint UNSIGNED AUTO_INCREMENT,
autor varchar(40) NOT NULL,
name varchar(80) NOT NULL,
added date NOT NULL,
href varchar(255),
catid smallint NOT NULL,
PRIMARY KEY (id))
```

Удаление таблиц и баз данных

DROP TABLE имя_таблицы;

Изменение записей таблицы

Добавление записи

INSERT INTO <имя таблицы>

(<имена полей, разделенные запятыми>)

VALUES (<значения полей, разделенные запятыми>);

Пример:

```
INSERT INTO items (name, author)
  VALUES ("Цвет в веб-дизайне",
  "Кирсанов, Д.");
```

Изменение записи

```
UPDATE <имя таблицы>
SET
   <имя 1-го поля>=<новое значение 1-го поля>,
   <имя 2-го поля>=<новое значение 2-го поля>
   ...
WHERE <критерий фильтрации для нахождения изменяемой записи>;
```

Например

```
UPDATE categories
SET name="Internet"
WHERE id=1;
```

Удаление записи

DELETE FROM <имя таблицы>
WHERE <критерий фильтрации, необходимый для нахождения удаляемой записи>;

Пример:

```
DELETE FROM categories
WHERE id=3;
```

Простейший запрос выборки данных

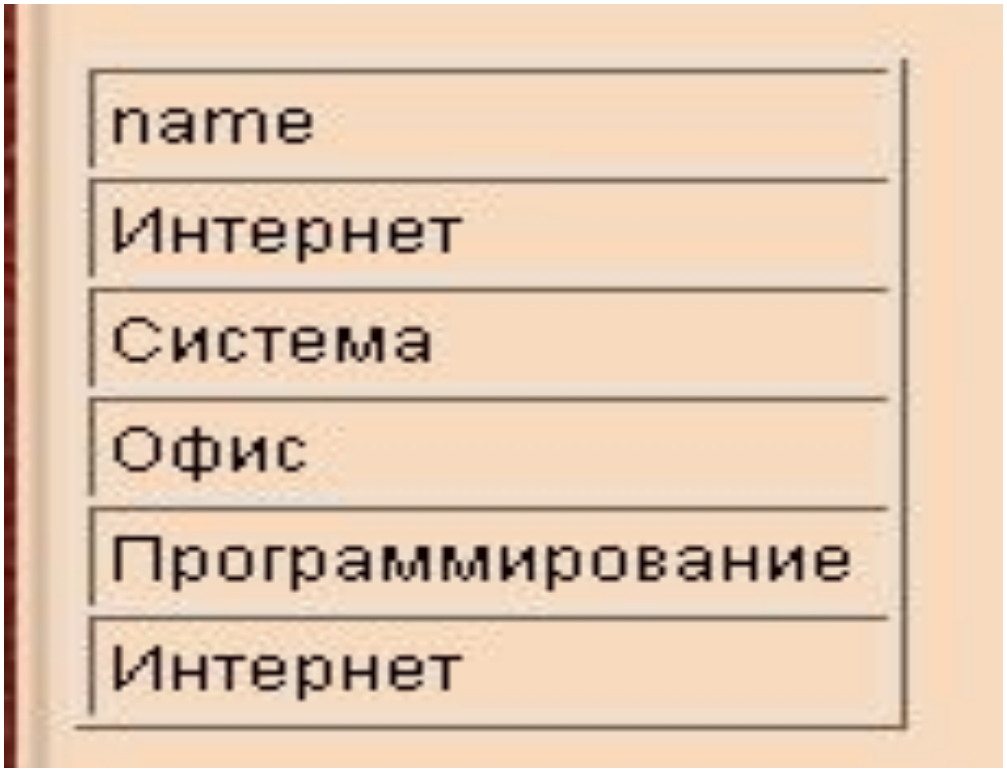
SELECT [DISTINCT] * | <список полей, разделенных запятыми>
FROM <имя таблицы>;

Если вместо списка полей подставить знак звездочка (*), будут выбраны все поля.

Примеры

```
SELECT * FROM items;

SELECT name FROM categories;
```



Список значений поля name таблицы categories, возвращенных запросом SQL

SELECT name FROM categories;

Если указано ключевое слово **DISTINCT**, то возвращаются только уникальные строки. Например, запрос:

```
SELECT DISTINCT name FROM categories;
```

вернет результат

name
Интернет
Система
Офис
Программирование

Вместо двух строк «Интернет» получили одну.

Сортировка данных

Для задания порядка сортировки служит дополнительные ключевые слова **ORDER BY**, которые ставятся в конец запроса:

. . . **ORDER BY <список критериев сортировки через запятую>**

Критерии сортировки имеют такой вид:

<имя поля, по которому ведется сортировка> [DESC]

Поля, по которым должна вестись сортировка записей, перечисляются через запятую после ключевого слова **ORDER BY**, которое, в свою очередь, ставится в конце запроса перед знаком точки с запятой.

Правила сортировки:

1. Сначала записи сортируются по полю, указанному первым в списке.
2. Если для некоторых записей значения этого поля одинаковы, то записи далее сортируются по полю, указанному вторым в списке.
3. Если для каких-то записей значения и этого поля одинаковы, то они будут отсортированы по полю, указанному третьим в списке.
4. И т.д.

По умолчанию записи сортируются, так, чтобы значения поля выстроились по возрастанию.

Если нужно отсортировать их по убыванию значений данного поля, нужно после имени этого поля поставить ключевое слово **DESC**.

Примеры:

```
SELECT * FROM items ORDER BY
  author;

SELECT file, name FROM categories
  ORDER BY file, name DESC;
```

file	name
Нет	Система
Нет	Интернет
Да	Программирование
Да	Офис
Да	Интернет

Фильтрация данных

Для фильтрации используется ключевое слово **WHERE**.

Это слово ставится между ключевыми словами **FROM** и **ORDER BY**:

. . . WHERE <список критериев фильтрации через запятую> . . .

Сами критерии фильтрации имеют вид:

<имя поля> <оператор сравнения>

<заданное значение>

Оператор сравнения задает равенство или неравенство заданного значения и значения поля.

Например:

id = 3

имя поля

оператор сравнения

заданное значение

Доступные в стандарте SQL операторы сравнения

Оператор сравнения	Описание
=	Равно
<> или !=	Не равно
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно

```
SELECT * FROM items WHERE
  author="Кирсанов, Д.";

SELECT * FROM items WHERE
  author<>"Андреев, А.";
```

Строковые величины, являющиеся частью критериев в запросах SQL, должны заключаться в кавычки!

Логические операторы **OR** и **AND**

```
SELECT * FROM items WHERE
  author="Кирсанов, Д." OR
  author="Андреев, А.";
```

```
SELECT id FROM categories
  WHERE name="Интернет" AND
  file=true;
```

Логический оператор **NOT (НЕ)**

```
SELECT * FROM categories
  WHERE NOT id = 3;
```

Логические операторы можно комбинировать

```
SELECT * FROM items WHERE NOT
  (author="Кирсанов, Д." OR
author="Андреев, А.");
```

```
SELECT * FROM items WHERE NOT
  author="Кирсанов, Д." OR
author="Андреев, А.";
```

Задание связей между таблицами

Чтобы связать две таблицы и получить из них данные, используется ключевое слово **WHERE**.

```
SELECT items.author, items.name,
  categories.name
FROM items, categories
WHERE items.catid=categories.id;
```

```
SELECT items.author,
  items.name, categories.name
  FROM items, categories
WHERE
  items.catid=categories.id;
```

author	name	name
Кирсанов, Д.	Цвет в веб-дизайне	Интернет
Семенов И.	К вопросу об эффективности поиска конкретики в Интернете	Интернет
Олифер, Н.	Сетевые операционные системы	Система
Андреев А.	Исследование активности рынка интернет-ссылок в Рунете	Интернет

```
SELECT items.author, items.name,
  categories.name
FROM items, categories
WHERE items.catid=categories.id
  AND categories.file=false
ORDER BY categories.name,
  items.name;
```

Псевдонимы полей

Язык SQL предоставляет возможность дать полю другое имя (так называемый псевдоним). Псевдоним создается с помощью ключевого слова AS:

SELECT . . . *<имя поля>* AS *<псевдоним>*, . . .

```
SELECT items.author, items.name
  AS item_name, categories.name
  AS cat_name
FROM items, categories
WHERE items.catid=categories.id
  AND
  categories.file=false
ORDER BY categories.name,
  items.name;
```

Агрегатные функции SQL

Группировка — это объединение записей в группы по какому-либо критерию, называемому *критерием группировки*. Выполняется группировка с помощью ключевого слова **GROUP BY**, после которого записываются сами критерии группировки:

GROUP BY <имена полей, по которым будут группироваться записи, через запятую>

Ставится перед ключевыми словами **ORDER BY**.

Поля, по которым ведется группировка записей, должны быть первыми в списке полей ключевого слова **SELECT** и располагаться в том же порядке, в котором они перечислены после ключевого слова **GROUP BY**.

Поля, по которым ведется группировка записей, должны быть первыми в списке полей ключевого слова **ORDER BY** и, опять же, располагаться в том же порядке, в котором они перечислены после ключевого слова **GROUP BY**.

Нужно получить количество статей в каждой категории.

```
SELECT categories.name,
  COUNT(items.name) AS item_count
FROM items, categories
WHERE items.catid=categories.id
  AND
  categories.file=false
GROUP BY categories.name
ORDER BY categories.name;
```

Результат:

name	item_count
Интернет	3
Система	1

Агрегатные функции в языке SQL

Агрегатная функция	Описание
COUNT (<поле>)	Количество записей
SUM(<поле>)	Сумма значений поля всех записей группы
AVG(<поле>)	Среднее значение поля всех записей группы
MIN(<поле>)	Минимальное из значений поля во всех записях группы
MAX(<поле>)	Максимальное из значений поля во всех записях группы