1. Mysql需要 在MySQLServer8.0文件夹中自己编写配置文件my.ini

**（在cmd的mysql命令中，不区分大小写，但是需要用“;”结束命令）**

1. **本机用户名：**MySQL3306

**端口**：3306

**IP地址：**192.168.1.105**（会变化）**

**密码：**1230re0321re

1. 安装mysql命令到系统：

cmd管理员模式，进入mysql server的bin目录，

命令：mysql –install

1. 查看系统用户目录信息:

SELECT DISTINCT CONCAT('User: ''',user,'''@''',host,''';') AS query FROM mysql.user;

或者：

select host,user,plugin from user;

**（注：**plugin---查看密码加密方式）

1. **常用命令：**

**对mysql服务器的操作命令(cmd使用管理员模式运行)**

初始化数据库：mysqld --initialize-insecure --user=mysql

开启服务器：net start mysql （若无法启动，则到服务中手动开启）

关闭服务器：net stop mysql

**登陆的操作命令**

windows登陆本地用户：mysql –u root –p

本地远程登陆：mysql -h **服务器IP** -P 3306**(端口)** -u root**(用户名)** -p

创建用户（由root权限----host=%）：

create user ‘chris’identified by '1230re0321re';

删除用户：Delete FROM user Where User='test' and Host='localhost';

修改用户名：update user set user="新用户名" where user="旧用户名";

flush privileges;

查看当前登陆的用户名：select user();

**对数据库的操作命令**

显示本地mysql所有的数据库：show databases;

选择使用的数据库：use mysql

显示当前数据库中的表格：show tables;

**对表格中的数据操作命令**

显示当前表格中的数据（显示两行）：select \* from bookinfo limit 2;

删除某张表中的所有数据：delete from 表格名字;

查找表格中有多少条数据：select count(\*) from 表格名字;

条件查找：select count(\*) from 表格名字 where （条件，eg：id>300）;

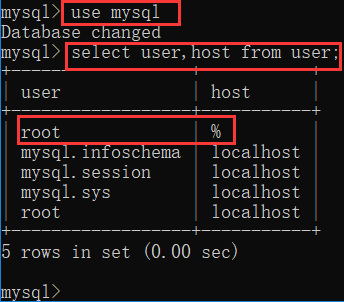
1. **本地远程登陆：（cmd使用管理员权限打开）——root用户远程登陆**

登陆本数据库的root用户（注意root用户的权限应为root，即：host=%）

注：**%：**代表由远程权限

**localhost：**代表本地

(**电脑的IP地址是会变化的**，有时候需要使用**ipconfig查看本机IP**地址，再使用对应的IP地址登陆MySQL)

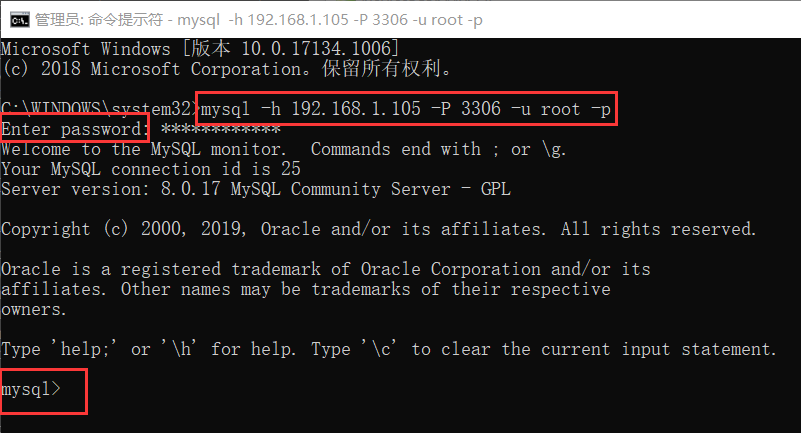


**命令**：本地远程登陆：mysql -h **服务器ip地址** -P 3306 -u **root** -p

**eg**：本地远程登陆：mysql -h **192.168.1.105** -P 3306 -u **root** -p

（注：**本机IP获取**：cmd中用ipconfig，得到网卡的ipv4地址就是。）

**登陆成功的样子：**



1. **本地远程登陆：（自己创建的用户）**

root登陆mysql之后：

**命令：**

**（注：mysql8.0将创建用户、权限设定的命令分离了，所以需要步骤1-2）**

1. **创建用户：**

create user '用户名'@'访问主机' identified by '密码';

**eg:** create user 'test'@'%' identified by '1230re0321re';

1. **赋予权限：（账户可以使用的数据库）**

grant 权限列表 on 数据库 to '用户名'@'访问主机' ;

**eg:** grant all privileges on \*.\* to 'test'@'%' ;

(修改权限时在后面加with grant option)

**（步骤3-4：修改密码加密方式）**

**（远程登陆的账号其密码加密方式为：mysql\_native\_password）**

1. alter user '用户名'@'%' identified by '密码' password expire never;

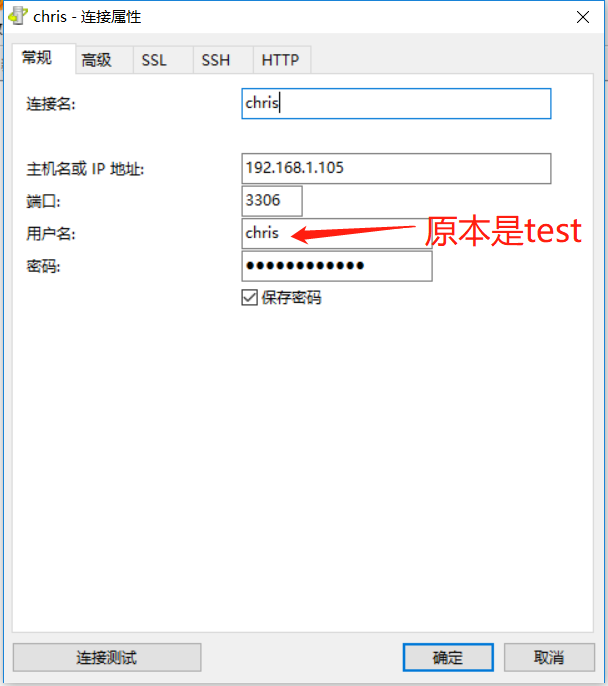
**eg：**alter user 'test'@'%' identified by '1230re0321re' password expire never;

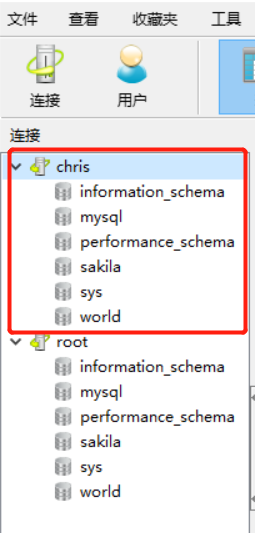
1. alter user '用户名'@'%' identified by with mysql\_native\_password by '密码';

**eg：**alter user 'test'@'%' identified with mysql\_native\_password by '1230re0321re';

1. flush privileges; （修改内容，立即生效-----刷新权限）

**登陆成功的样子：**





1. **windows10启用端口3306的教程：**

<https://blog.csdn.net/yj19961126/article/details/81224107>

1. **关于MySQL的重要说明：**

* **MySQL中表格名字、数据库名，都不区分大小写**
* **表格中的参数变量名，区分大小写**
* 在进行模糊查询的时候，可以使用like，like需要结合%使用。
* '%a' ：以a结尾的数据 （%：表示匹配任意个字符）
* 'a%' ：以a开头的数据
* '%a%' ：含有a的数据
* '\_a\_' ：三位且中间字母是a的 （\_：表示匹配一个字符）
* '\_a' ：两位且结尾字母是a的
* 'a\_' ：两位且开头字母是a的

**eg：**

String sql = SELECT \* from table WHERE author LIKE '%chris'

* **SQL注入：**

SQL注入，就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

我们永远不要信任用户的输入，我们必须认定用户输入的数据都是不安全的，需要**对用户输入的数据进行过滤处理**。

**eg：**

过滤用户输入的名字，规定只能使用数字、字母、下划线，长度在8-20

if (preg\_match("/^\w{8,20}$/", $\_GET['username'], $**matches**))

{

$result = mysqli\_query($conn, "SELECT \* FROM users

WHERE username=$**matches**[0]");

}

else

{

echo "username 输入异常";

}



# 常用的SQL语句

1. **创建表：**

/\*

创建商品信息表

\*/

**create table** **`product\_info`** ( **--注意：表格名，属性名用的``，中文用’’**

**`product\_id`** varchar(32) not null **AUTO\_INCREMENT**, --**自增**

**`product\_price`** decimal(8,2) not null comment **'单价'**,

**`product\_stock`** int not null comment '库存', **--值：不能为空**

**`create\_time`** **timestamp** not null **default** current\_timestamp comment '创建时间',  **--时间戳、默认值**

**`update\_time`** **timestamp** not null **default** current\_timestamp **on update** current\_timestamp comment '修改时间',

**`type`** mediumint(1) **NOT NULL**,

**primary key** (**`product\_id`**) **--创建主键：id**

**KEY** **`index\_type`** (**`type`**) **--创建索引检索名：index\_type**

) **comment** '商品表'; **--表属性中注释名：商品表**

1. **primary key、unique key、foreign key、index的区别：**

（除了index之外，**其他的key都具有约束、索引的作用，只要是key都会在对应的key上建立一个index**）

1. **primary key**：主键，一张表只能有一个，且不能为null。唯一标识数据库表中的每条记录；主键必须包含唯一的值。
2. **unique key**：唯一键，建立唯一索引（index），一张表可以有多个索引。（和primary key类似）
3. **foreign key**：外部键，即：在该表中存放一列其他表的主键id。当需要进行关联表查询时，可以用这个外部键进行查找——不使用级联表的原因：级联表在数据量大时，容易导致数据库宕机，因为在进行增、删、改时级联表需要进行多表操作；
4. **index**：索引，辅助查询，创建index时会在innodb表空间中创建一个类似目录的结构存储（对数据库表中一列或多列的值进行排序—的一种结构）。用来快速找出在一个列上用一特定值的行。若没有索引，MySQL不得不首先以第一条记录开始并然后读完整个表直到它找出相关的行。

**eg：创建index索引**

create table t(id int,**unique index** inx\_tx\_id (id)); **index当作key使用**

SQL基本语法

1. **SQL语法分类：**

* DCL：数据库控制语言，eg：授权grant、取消授权：revoke
* DML：data manipulation language，数据操作语言；
* select：从数据库中选择数据
* update：更新数据
* delete：删除数据
* insert into：插入数据
* DDL：data definition language， 数据定义语言；
* create databse：创建数据库；
* alter database：修改数据；
* create table：创建表；
* alter table：修改表 ；
* drop table：删除整张表；
* create index：创建索引；
* drop index：删除索引；
* truncate table 表名：删除整张表格的数据，但保留数据结构（属性名）

1. **（DML）基本数据操作语言使用： \*表示全选**

* **select：从表格中选择数据**
* **格式：**SELECT 列名称 FROM 表名称
* **示例：**SELECT LastName,FirstName FROM Persons

* **distinct：去除重复数据**
* **格式：**SELECT DISTINCT 列名称 FROM 表名称
* **示例：**（取出Orders表格中Company列的不重复数据）

SELECT DISTINCT Company FROM Orders

* **where**：筛选数据，添加条件，从表格中取出符合条件的数据
* **格式：**SELECT 列名称 FROM 表名称 WHERE 列 运算符 值
* **示例**：（从表格Persons中取出列值为Beijing的数据）

SELECT \* FROM Persons WHERE City='Beijing'

* **注意：**值 = 字符时，用'字符'

值 = 数字时，用数字表示

* **and、or：**逻辑链接：
* **示例：**从Persons表格中，获取FirstName = Thomas、William，且LastName = Carter的数据；

SELECT \* FROM Persons WHERE **(**FirstName='Thomas' OR FirstName='William'**)** AND LastName='Carter'

* **注意**：可以使用（）拼接条件

* **order by**：排序
* **格式：** SELECT 列 FROM 表 ORDER BY 需要排序的列
* **示例：** 从Orders表中取出数据，Company降序，OrderNumber升序

SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC, OrderNumber ASC

* **注意：**

DESC（descending）：降序； ASC（ascending）：升序（默认值）

**order by name； #默认就是升序**

* **inset to：再表格中插入新的行**
* **格式：**INSERT INTO 表名 (列1, 列2,...) VALUES (值1, 值2,....)
* **示例：**向Persons表格中的LastName、Address列插入数据

INSERT INTO Persons (LastName, Address) VALUES ('Wilson', 'Champs-Elysees')

* **update**：更新数据，更新某行某列的数据、或某行若干列的数据
* **格式：**UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值
* **示例：**更新Person表格，列名为LastName中，数据 = Wilson一行的Address、City列的数据；

UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing'

WHERE LastName = 'Wilson'

* **delete**：删除表格中的行
* **格式**：DELETE FROM 表名称 WHERE 列名称 = 值
* **示例**：删除Person表中，列名为LastName中，数据 = Wilson一行的数据

DELETE FROM Person WHERE LastName = 'Wilson'

* **注意**：删除所有行的SQL语句：

DELETE FROM table\_name 或 DELETE \* FROM table\_name

* **having：**用于where后面拼接聚合函数，因为where后面无法使用这写函数。常见的聚合函数：

count(\*)、avg、max、min、sum等

**eg：求出t\_user表中重复的名字**

select name from t\_user group by name **having** count(1) > 1

* **group by**

根据某些列对查找的结果进行分组。

**本质上：**先排序后分组，基于order by，遵循索引的最左前缀原则。如若在where中能处理的问题，就不要使用having去限定条件。

**eg：**求出t\_user表中，按照年龄分组的数据

select age from t\_user **group by** age

**或者，（这两组的是等价的）**

select **distinct** age from t\_user

**eg：获取姓重复数据的id**

select id from t\_user where name **in** (select name from t\_user **group by** name **having** **count(1)** > 1)

1. **SQL语句嵌套执行：**

（1）**嵌套查询**：一个查询语句(select-from-where)查询语句块可以嵌套在另外一个查询块的where子句中。其中外层查询也称为父查询，主查询。内层查询也称子查询。最多嵌套32层子查询。

（2）**作用**：实现多表、多条件查询。

* **简单嵌套查询**

select name,age from t\_person where **age >** (

select **age** from t\_person where name = '孙权'

)；

* **in嵌套查询：**

从子句中获取查询的结果，并且在主句中采用类似于for循环的方式遍历子句返回的结果。

select name from t\_person where **country\_id in** (

select **country\_id** from t\_country where country\_name = '魏国'

);

* **some嵌套查询：**

从子句中返回的结果，主句中根据这些结果进行查询，只要有符合条件的就返回。

select name from t\_person where **country\_id** = **some**(

select **country\_id** from t\_country where country\_name = '魏国'

);

* **all嵌套查询：**

当主句中查询结果全部大于子句中返回结果时，才会返回这些查询的结果。

select name from t\_person where **country\_id** > **all**(

select **country\_id** from t\_country where country\_name = '魏国'

);

* **extis嵌套查询：**

主句中如果存在子句返回的结果，**就直接返回主句查询的第一个结果**（主句不会查询所有与子句返回结果相同的数据）

select \* from t\_person where **exists**(

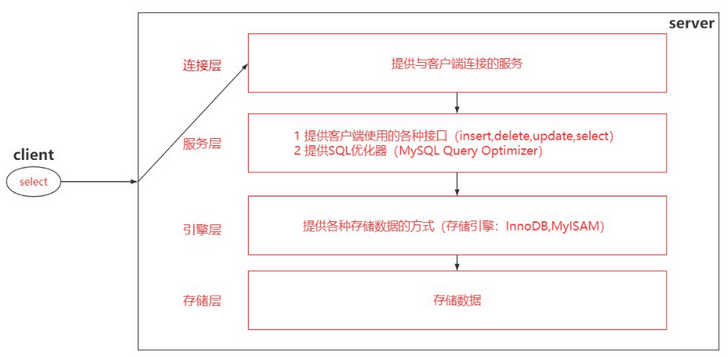
select \* from t\_erson where person\_Id = 100

);



数据库引擎

1. **MySQL逻辑分层：**

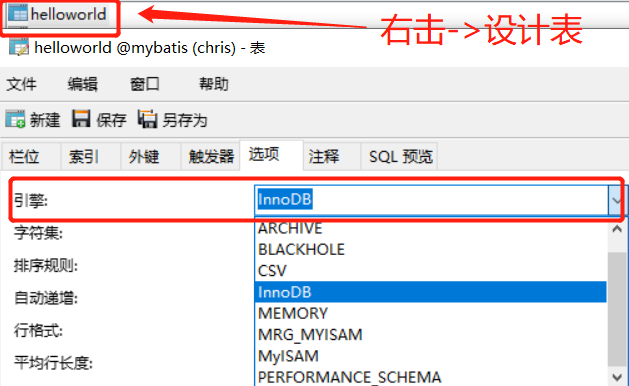


1. **引擎：**

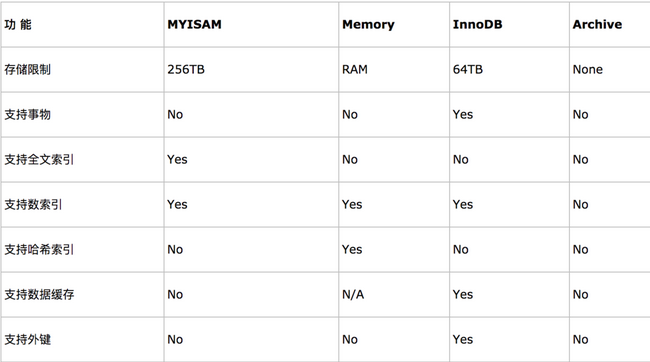
数据库管理系统（DBMS）使用数据搜索引擎进行创建、查找、更新、删除数据（CRUD），不同的引擎提供不同的存储机制、索引技巧、锁定水平等。**MySQL的核心就是存储引擎。**

数据库引擎是用于存储、处理和保护数据的核心服务。当你访问数据库时，不管是手工访问，还是程序访问，都不是直接读写数据库文件，而是通过数据库引擎去访问数据库文件。

**注意：**数据引擎是指表格中的引擎；（例：使用navicat修改表格引擎）



1. **不同存储引擎的对比：**



**如何选择引擎：**

1. **MYISAM**：只需要插入、查询操作，不支持事务，MYISAM效率高；
2. **Memory**：只需要临时存放数据，数据量不大，不需要较高的数据安全性，MySQL使用该引擎作为临时表，存放查询的中间结果；
3. **InnoDB**（MySQL的默认引擎）：需要提交、回滚、崩溃恢复的事务安全、并发控制；（支持事务）
4. **Archive**：只需要插入、查找 操作，支持高并发的插入操作，但不需要事务安全。（例如：数据归档、记录日志信息）