第04天 MySQL

**【学习目标】理解、了解、应用、记忆**

1. **【理解】数据库概述**
2. 【理解】阐述对数据库概念的理解
3. 【理解】阐述常见数据库有哪些,开源数据库有哪些
4. **【应用】MySQL和**Navicat**安装与使用**
5. 【理解】独立完成MySQL的安装预配置
6. 【理解】阐述MySQL启动/停止方式及登录MySQL登录名的命令
7. 【理解】独立安装并配置Navicat
8. **【应用】SQL语句**
9. 【理解】阐述SQL语句中的分类及SQL语句中的数据类型
10. 【应用】独立编写SQL语句完成创建数据库、显示数据库、删除数据库等操作
11. 【应用】独立编写SQL语句完成创建表、修改表、删除表等操作。
12. 【应用】独立编写SQL语句使用INSERT语句给数据库中插入数据
13. 【应用】独立编写SQL使用UPDATE语句修改表中数据
14. 【应用】独立编写sql使用DELETE删除表中数据
15. 【应用】阐述DOS命令出现乱码的现象、原因及解决办法
16. 【应用】独立编写SQL语句完成相关查询操作

# 数据库的概述

## 数据库的概述

### 概述

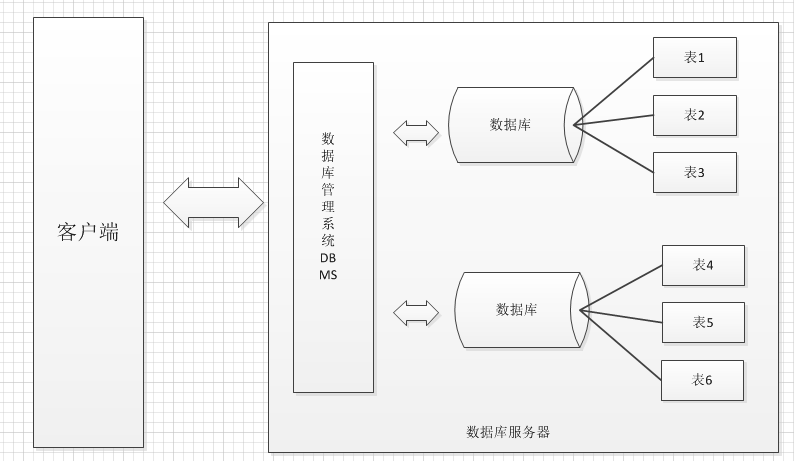
* 什么是数据库

数据库就是存储数据的仓库，其本质是一个文件系统，数据按照特定的格式将数据存储起来，用户可以对数据库中的数据进行增加，修改，删除及查询操作。

* 什么是数据库管理系统

数据库管理系统（DataBase Management System，DBMS）：指一种操作和管理数据库的大型软件，用于建立、使用和维护数据库，对数据库进行统一管理和控制，以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中表内的数据。

* 数据库与数据库管理系统的关系



### 数据库表

数据库中以表为组织单位存储数据。

表类似我们的Java类，每个字段都有对应的数据类型。

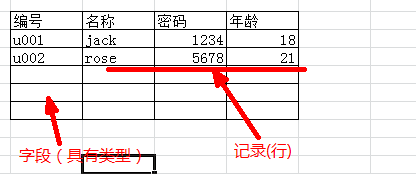


那么用我们熟悉的java程序来与关系型数据对比，就会发现以下对应关系。

类----------表

类中属性----------表中字段

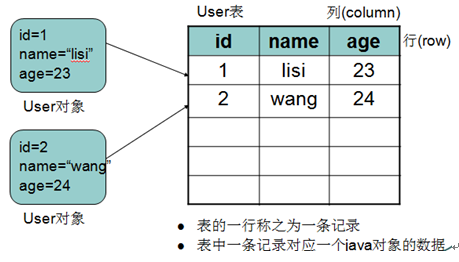
对象----------记录



### 表数据

根据表字段所规定的数据类型，我们可以向其中填入一条条的数据，而表中的每条数据类似类的实例对象。表中的一行一行的信息我们称之为记录。

* 表记录与java类对象的对应关系



## 常见数据库

常见的数据库管理系统

MYSQL ：开源免费的数据库，小型的数据库.已经被Oracle收购了.MySQL6.x版本也开始收费。

Oracle ：收费的大型数据库，Oracle公司的产品。Oracle收购SUN公司，收购MYSQL。

DB2 ：IBM公司的数据库产品,收费的。常应用在银行系统中.

SQLServer：MicroSoft 公司收费的中型的数据库。C#、.net等语言常使用。

SyBase ：已经淡出历史舞台。提供了一个非常专业数据建模的工具PowerDesigner。

SQLite : 嵌入式的小型数据库，应用在手机端。

常用数据库：**MYSQL**，Oracle．

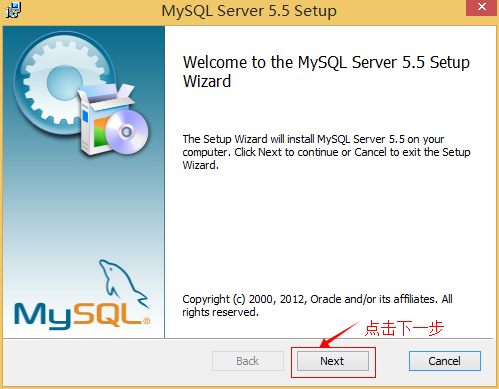
这里使用MySQL数据库。MySQL中可以有多个数据库，数据库是真正存储数据的地方。

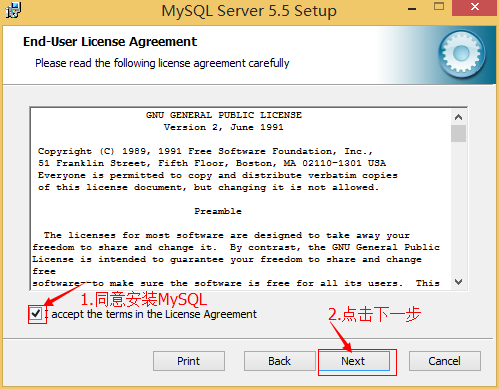
# MySQL和Navicat的安装与使用

## MySQL安装与配置

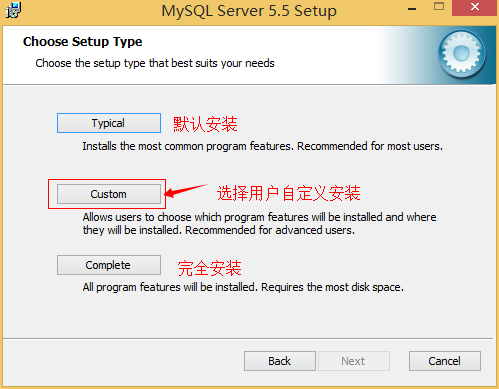
### MYSQL的安装

１、打开下载的mysql安装文件mysql-5.5.27-winx64.msi，双击运行。

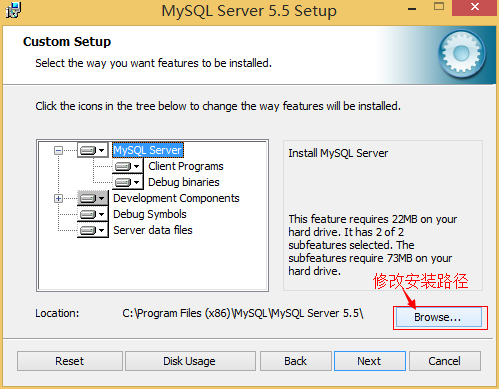




２、选择安装类型，有“Typical（默认）”、“Complete（完全）”、“Custom（用户自定义）”三个选项，选择“Custom”，按“next”键继续。



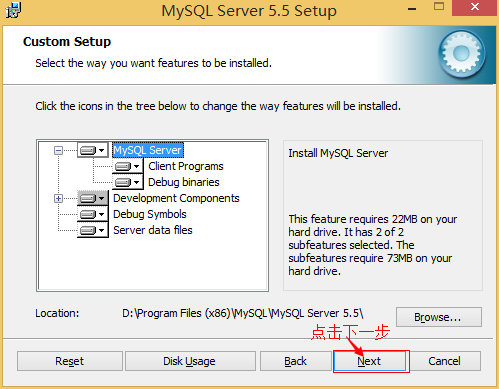
３、点选“Browse”，手动指定安装目录。

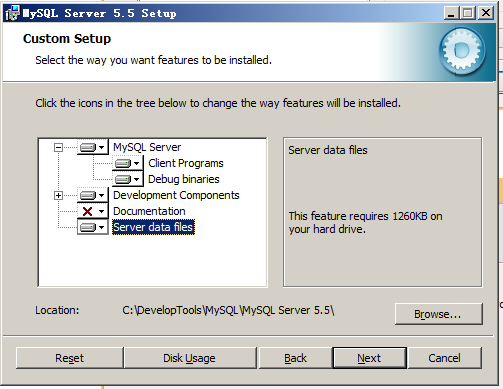


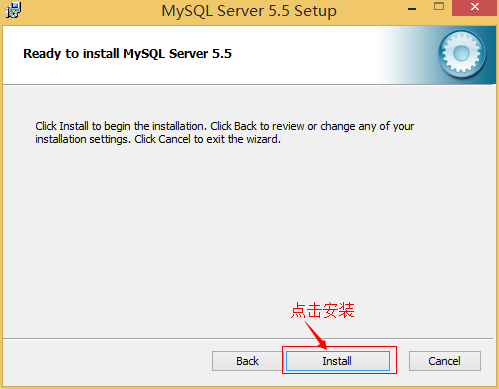
４、填上安装目录，我的是“F:\Server\MySQL\MySQL Server 5.0”，也建议不要放在与操作系统同一分区，这样可以防止系统备份还原的时候，数据被清空。按“OK”继续。

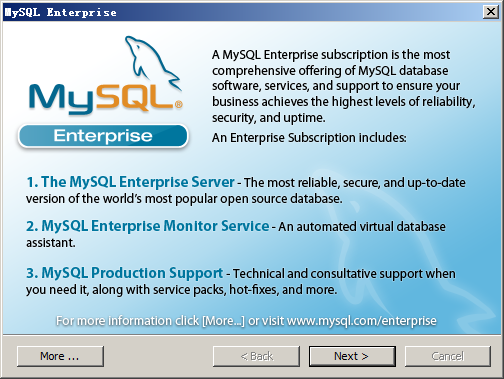


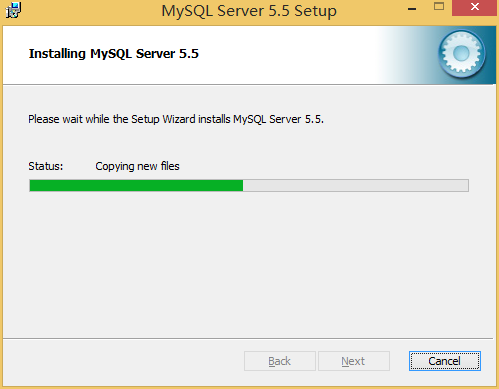
确认一下先前的设置，如果有误，按“Back”返回重做。按“Install”开始安装。



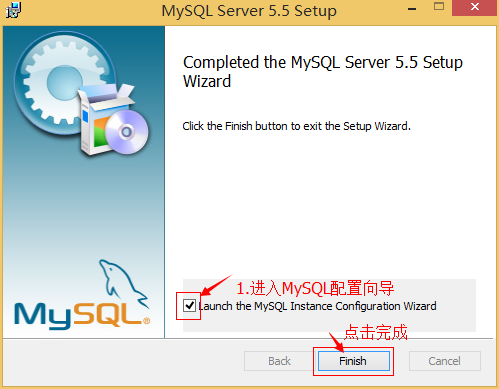






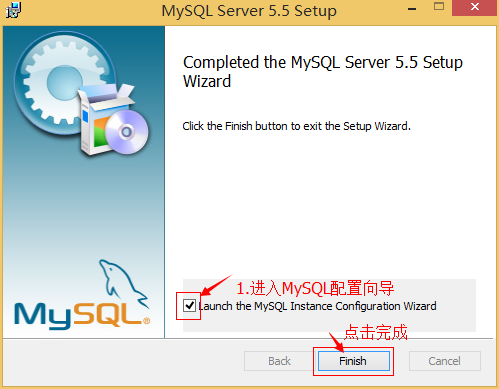


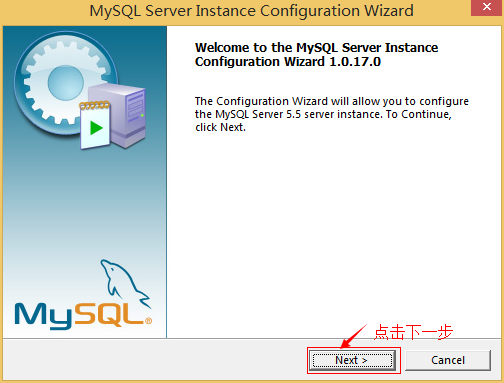
５、正在安装中，请稍候，直到出现下面的界面, 则完成MYSQL的安装



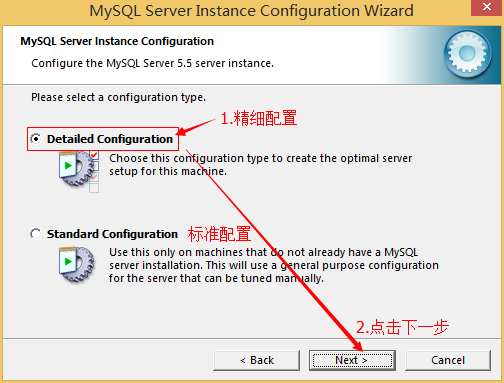
### MYSQL的配置

１、安装完成了，出现如下界面将进入mysql配置向导。

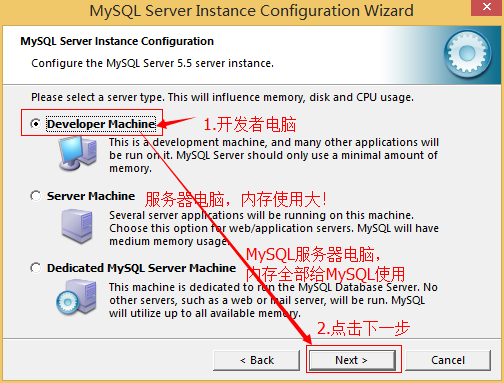




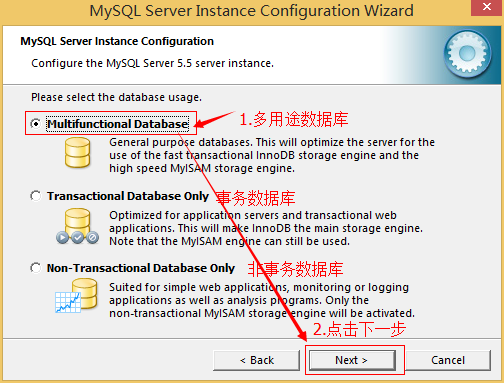
２、选择配置方式，“Detailed Configuration（手动精确配置）”、“Standard Configuration（标准配置）”，我们选择“Detailed Configuration”，方便熟悉配置过程。



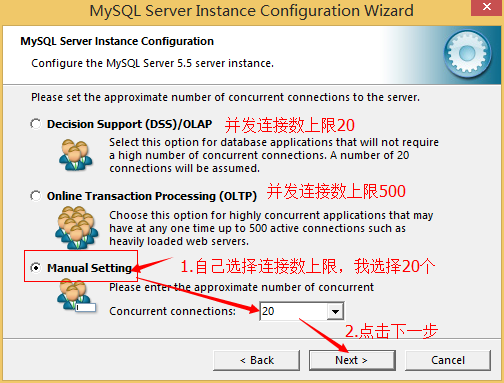
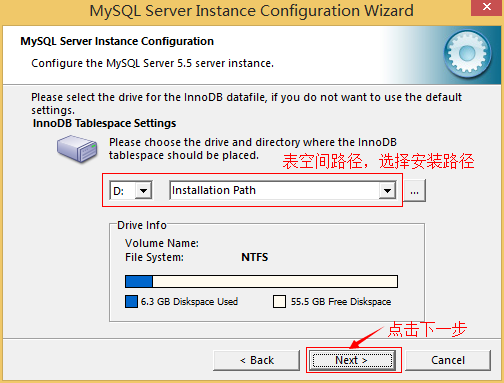
３、选择服务器类型，“Developer Machine（开发测试类，mysql占用很少资源）”、“Server Machine（服务器类型，mysql占用较多资源）”、“Dedicated MySQL Server Machine（专门的数据库服务器，mysql占用所有可用资源）”



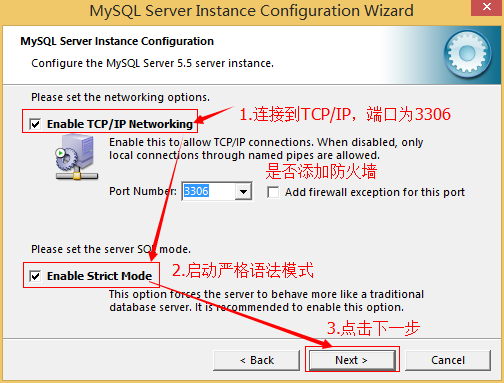
４、选择mysql数据库的大致用途，“Multifunctional Database（通用多功能型，好）”、“Transactional Database Only（服务器类型，专注于事务处理，一般）”、“Non-Transactional Database Only（非事务处理型，较简单，主要做一些监控、记数用，对MyISAM数据类型的支持仅限于non-transactional），按“Next”继续。



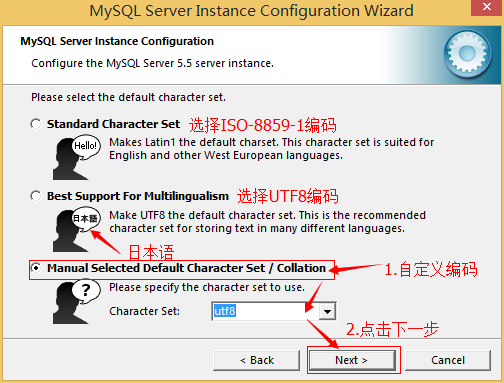
５、选择网站并发连接数，同时连接的数目，“Decision Support(DSS)/OLAP（20个左右）”、“Online Transaction Processing(OLTP)（500个左右）”、“Manual SETting（手动设置，自己输一个数）”。



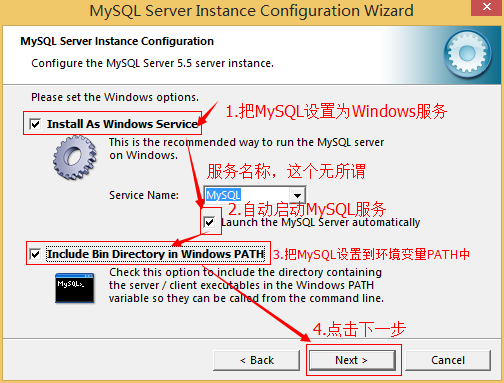
６、是否启用TCP/IP连接，设定端口，如果不启用，就只能在自己的机器上访问mysql数据库了，在这个页面上，您还可以选择“启用标准模式”（Enable Strict Mode），这样MySQL就不会允许细小的语法错误。如果是新手，建议您取消标准模式以减少麻烦。但熟悉MySQL以后，尽量使用标准模式，因为它可以降低有害数据进入数据库的可能性。按“Next”继续



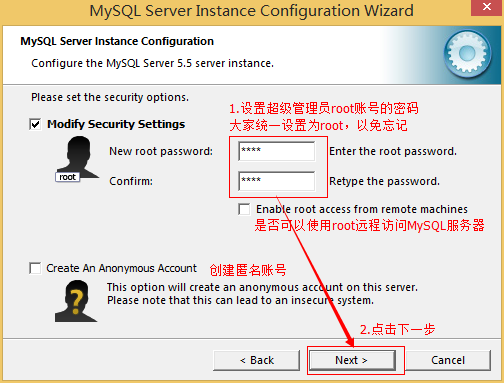
７、就是对mysql默认数据库语言编码进行设置（重要），一般选UTF-8，按 “Next”继续。



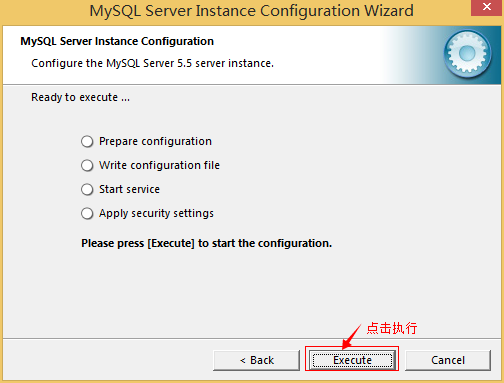
８、选择是否将mysql安装为windows服务，还可以指定Service Name（服务标识名称），是否将mysql的bin目录加入到Windows PATH（加入后，就可以直接使用bin下的文件，而不用指出目录名，比如连接，“mysql.exe -uusername -ppassword;”就可以了，不用指出mysql.exe的完整地址，很方便），我这里全部打上了勾，Service Name不变。按“Next”继续。

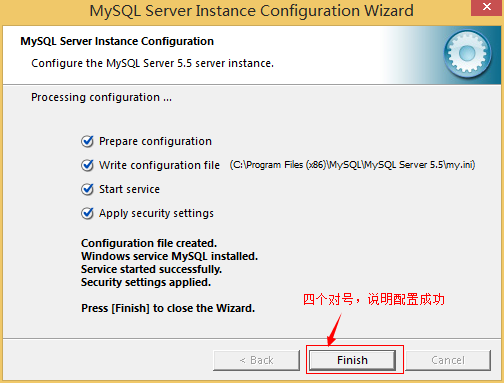


９、询问是否要修改默认root用户（超级管理）的密码。“Enable root access from remote machines（是否允许root用户在其它的机器上登陆，如果要安全，就不要勾上，如果要方便，就勾上它）”。最后“Create An Anonymous Account（新建一个匿名用户，匿名用户可以连接数据库，不能操作数据，包括查询）”，一般就不用勾了，设置完毕，按“Next”继续。

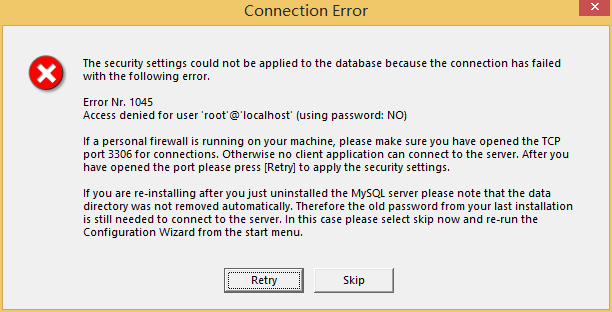


１０、确认设置无误，按“Execute”使设置生效，即完成MYSQL的安装和配置。





**注意：设置完毕，按“Finish”后有一个比较常见的错误，就是不能“Start service”，一般出现在以前有安装mysql的服务器上，解决的办法，先保证以前安装的mysql服务器彻底卸载掉了；不行的话，检查是否按上面一步所说，之前的密码是否有修改，照上面的操作；如果依然不行，将mysql安装目录下的data文件夹备份，然后删除，在安装完成后，将安装生成的 data文件夹删除，备份的data文件夹移回来，再重启mysql服务就可以了，这种情况下，可能需要将数据库检查一下，然后修复一次，防止数据出错。**



**解决方法：**

**1, 卸载MySQL**

**2, Windows Xp系统删除目录 C:\Documents and SETtings\All Users\Application Data\**

**windows 7\8\10操作系统删除目录C:\ProgramData\MySQL**

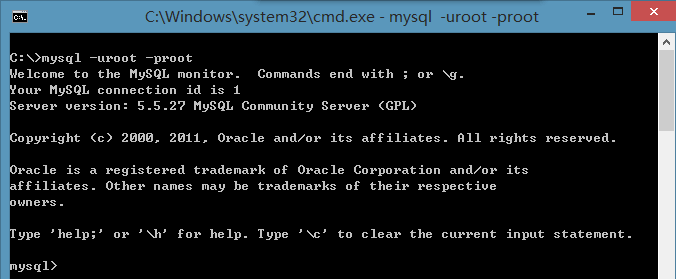
**3、重新安装就可以了**

## MySQL启动与登录

MySQL是一个需要账户名密码登录的数据库，登录后使用，它提供了一个默认的root账号，使用安装时设置的密码即可登录。

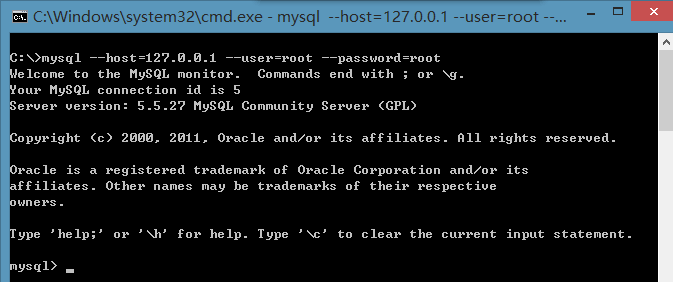
格式1：cmd> mysql –u用户名 –p密码

例如：mysql -uroot –proot



格式2：cmd> mysql --host=ip地址 --user=用户名 --password=密码

例如：mysql --host=127.0.0.1 --user=root --password=root



## Navicat安装与使用

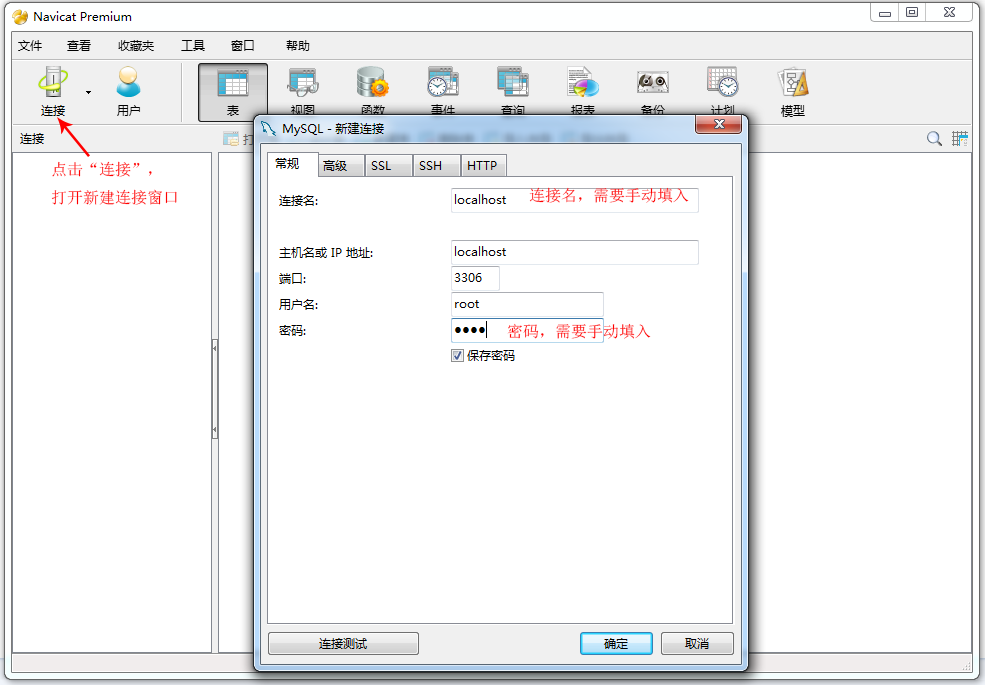
* 安装：



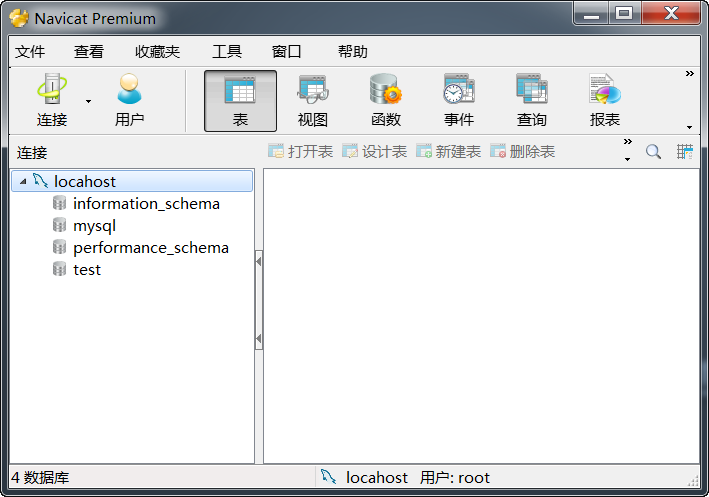


* 使用：

输入连接名、IP地址、端口号、用户名、密码，点击确定按钮，进行访问MySQL数据库操作



连接成功之后双击连接名，打开连接：



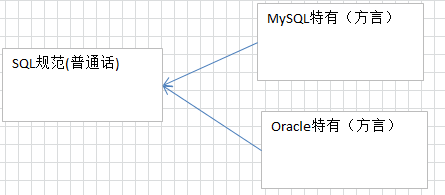
# SQL语句

## SQL语句—概述

### SQL语句介绍

数据库是不认识JAVA语言的，但是我们同样要与数据库交互，这时需要使用到数据库认识的语言SQL语句，它是数据库的代码。

结构化查询语言(Structured Query Language)简称SQL，是关系型数据库管理系统都需要遵循的规范。不同的数据库生产厂商都支持SQL语句，但都有特有内容。

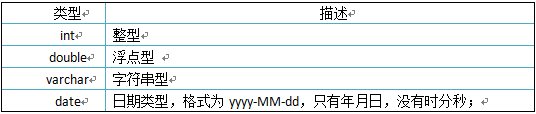


### SQL语句分类

* SQL分类：
  + 数据定义语言：简称DDL(Data Definition Language)，用来定义数据库对象：数据库，表，列等。关键字：create，alter，drop等
  + 数据操作语言：简称DML(Data Manipulation Language)，用来对数据库中表的记录进行更新。关键字：INSERT，DELETE，UPDATE等
  + 数据控制语言：简称DCL(Data Control Language)，用来定义数据库的访问权限和安全级别，及创建用户。
  + 数据查询语言：简称DQL(Data Query Language)，用来查询数据库中表的记录。关键字：select，from，WHERE等

### SQL通用语法

* SQL语句可以单行或多行书写，以分号结尾
* 可使用空格和缩进来增强语句的可读性
* MySQL数据库的SQL语句不区分大小写，关键字建议使用大写
  + 例如：SELECT \* FROM user。
* 同样可以使用/\*\*/的方式完成注释
* MySQL中的我们常使用的数据类型如下



详细的数据类型如下(不建议详细阅读！)

|  |  |  |
| --- | --- | --- |
| **分类** | **类型名称** | **说明** |
| **整数类型** | tinyInt | 很小的整数 |
| smallint | 小的整数 |
| mediumint | 中等大小的整数 |
| int(integer) | 普通大小的整数 |
| **小数类型** | float | 单精度浮点数 |
| double | 双精度浮点数 |
| decimal（m,d） | 压缩严格的定点数 |
| **日期类型** | year | YYYY 1901~2155 |
| time | HH:MM:SS -838:59:59~838:59:59 |
| date | YYYY-MM-DD 1000-01-01~9999-12-3 |
| datetime | YYYY-MM-DD HH:MM:SS 1000-01-01 00:00:00~ 9999-12-31 23:59:59 |
| timestamp | YYYY-MM-DD HH:MM:SS 1970~01~01 00:00:01 UTC~2038-01-19 03:14:07UTC |
| **文本、二进制类型** | CHAR(M) | M为0~255之间的整数 |
| VARCHAR(M) | M为0~65535之间的整数 |
| TINYBLOB | 允许长度0~255字节 |
| BLOB | 允许长度0~65535字节 |
| MEDIUMBLOB | 允许长度0~167772150字节 |
| LONGBLOB | 允许长度0~4294967295字节 |
| TINYTEXT | 允许长度0~255字节 |
| TEXT | 允许长度0~65535字节 |
| MEDIUMTEXT | 允许长度0~167772150字节 |
| LONGTEXT | 允许长度0~4294967295字节 |
| VARBINARY(M) | 允许长度0~M个字节的变长字节字符串 |
| BINARY(M) | 允许长度0~M个字节的定长字节字符串 |

## DDL—数据库操作

### 创建数据库

格式:

\* create database 数据库名;

\* create database 数据库名 character SET 字符集;

例如：

#创建数据库 数据库中数据的编码采用的是安装数据库时指定的默认编码 utf8

CREATE DATABASE igeek\_01;

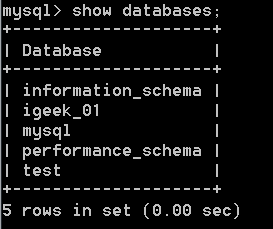
#创建数据库 并指定数据库中数据的编码

CREATE DATABASE igeek\_02 CHARACTER SET utf8;

### 查看数据库

查看数据库MySQL服务器中的所有的数据库:

show databases;

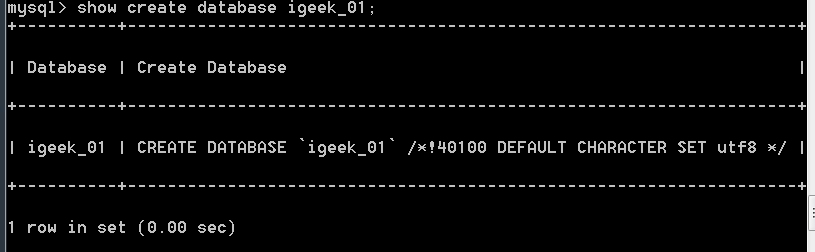


查看某个数据库的定义的信息:

show create database 数据库名;

例如：

show create database igeek\_01;



### 删除数据库

drop database 数据库名称;

例如：

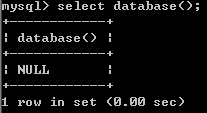
drop database igeek\_02;



### 使用数据库

* 查看正在使用的数据库:

select database();



* 其他的数据库操作命令

切换数据库：

use 数据库名;

例如：

use igeek\_01;

### 案例代码一

###数据库操作

# 创建数据库 (默认字符集: UTF-8 , 此编码为安装时设置)

CREATE DATABASE igeek\_01;

CREATE DATABASE igeek\_02 CHARACTER SET gbk;

#显示所有数据

SHOW DATABASES;

#显示数据库的创建语句

SHOW CREATE DATABASE igeek\_01;

SHOW CREATE DATABASE igeek\_02;

#删除数据库

DROP DATABASE igeek\_02;

## DDL--表结构操作

### 创建表

* 格式：

create table 表名(

字段名 类型(长度) [约束],

字段名 类型(长度) [约束],

...

);

类型：

varchar(n) 字符串

int 整形

double 浮点

date 时间

timestamp 时间戳

约束：

Primary key 主键，被主键修饰字段中的数据，不能重复、不能为null。

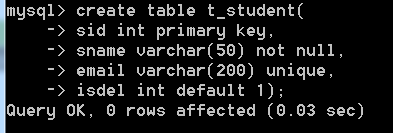
Not null 非空约束

Unique 唯一约束

Default 默认约束

例如：

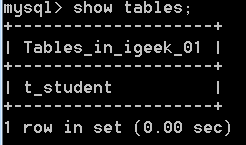
###创建学生表



### 查看表

* 查看数据库中的所有表：

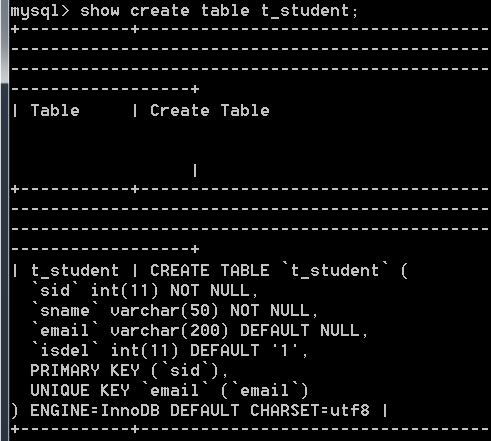
格式：show tables;



* 查看创建表语句：

格式：show create table表名;

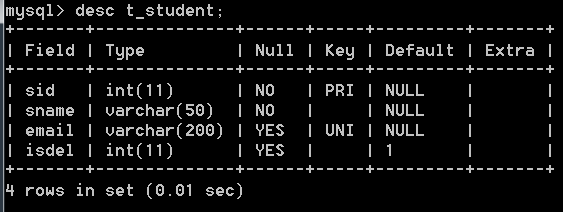
例如：show create tablet\_student;



* 查看表结构：

格式：desc 表名;

例如：desc t\_student;



### 删除表

* 格式：drop table 表名;

例如：drop table t\_student;



### 修改表结构格式

* alter table 表名 add 列名 类型(长度) [约束];

作用：修改表添加列.

例如：

#1，为学生表添加一个新的字段为 班级名称varchar(50)

ALTER TABLE t\_student ADD classname VARCHAR(50);

* alter table 表名 modify 列名 类型(长度) 约束;

作用：修改表修改列的类型长度及约束.

例如：

#2, 将学生表的班级名称修改成int类型

ALTER TABLE t\_student MODIFY classname int;

* alter table 表名 change 旧列名 新列名 类型(长度) 约束;

作用：修改表修改列名.

例如：

#3,将学生表的班级名称字段更换为 classid int

ALTER TABLE t\_student CHANGE classname classid int;

* alter table 表名 drop 列名;

作用：修改表删除列.

例如：

#4, 删除学生表中classid这列

ALTER TABLE t\_student DROP classid;

* rename table 表名 to 新表名;

作用：修改表名

例如：

#5, 为将学生表名t\_student 改名成 t\_ students

RENAME TABLE t\_student TO t\_students;

* alter table 表名 character SET 字符集;

作用：修改表的字符集

例如：

#6, 将学生表 t\_students 的编码表进行修改，修改成 gbk

ALTER TABLE t\_students CHARACTER SET gbk;

### 案例代码二

#######表 table

# 创建表

## 有主键

CREATE TABLE t\_student(

Sid int PRIMARY KEY ,

Sname VARCHAR(50) NOT NULL,

Email VARCHAR(200) UNIQUE,

Isdel INT DEFAULT 1

);

## 创建表之后，可以查询创建表语句 (数据在可视化视图中显示不全)

SHOW CREATE TABLE t\_student;

##查看表结构

DESC t\_student;

#删除表

DROP TABLE t\_student;

#修改表结构 -- 如果SQL语句中使用到关键字，建议使用重音符括住 `

##添加字段

ALTER TABLE t\_student ADD classname VARCHAR(50);

##修改字段类型

ALTER TABLE t\_student MODIFY classname int;

##修改整个字段

ALTER TABLE t\_student CHANGE classname classid int;

##删除字段

ALTER TABLE t\_student DROP classid;

##修改表名

RENAME TABLE t\_student TO t\_students;

##修改字符集

ALTER TABLE t\_students CHARACTER SET gbk;

## DML数据操作语言

### 插入表记录：INSERT

* 语法：

-- 向表中插入某些字段

INSERT INTO 表 (字段1,字段2,字段3..) VALUES (值1,值2,值3..);

--向表中插入所有字段,字段的顺序为创建表时的顺序

INSERT INTO 表 VALUES (值1,值2,值3..);

* 注意：
  + 值与字段必须对应，个数相同，类型相同
  + 值的数据大小必须在字段的长度范围内
  + 除了数值类型外，其它的字段类型的值必须使用引号引起。（建议单引号）
  + 如果要插入空值，可以不写字段，或者插入 null。
  + 注意每一列的约束问题
* 例如：

INSERT INTO t\_Students(sid,sname,email,isdel) VALUES(1001,'张三','张三@qq.com',1);

INSERT INTO t\_Students(sid,sname,email) VALUES(1002,'abc','abc');

INSERT INTO t\_Students(sid,sname) VALUES(1004,'lisi');

### 案例代码三

#插入数据

INSERT INTO t\_Students(sid,sname,email,isdel) VALUES(1001,'张三','张三@qq.com',1);

#插入部分数据：

INSERT INTO t\_Students(sid,sname,email) VALUES(1002,'abc','abc');

没有插入值的默认是null或者是默认值

INSERT INTO t\_Students(sid,sname) VALUES(1004,'lisi');

#所有值都加上’’

INSERT INTO t\_Students(sid,sname,email,isdel) VALUES('1003','张三','zhangsan@qq.com','1');

#约束问题

主键约束

INSERT INTO t\_Students(sid,sname) VALUES('1003','张三');

非空约束

INSERT INTO t\_Students(sid) VALUES('1004');

唯一约束

INSERT INTO t\_Students(sid,sname,email) VALUES('1005','张三','zhangsan@qq.com');

#插入所有列的简写方式

INSERT INTO t\_Students VALUES('1009','zhaosi','zhangsan3@qq.com','0');

### 更新表记录：UPDATE

用来修改指定条件的数据，将满足条件的记录指定列修改为指定值

* 语法：

--更新所有记录的指定字段

UPDATE 表名 SET 字段名=值,字段名=值,...;

--更新符号条件记录的指定字段

UPDATE 表名 SET 字段名=值,字段名=值,... WHERE 条件;

* 注意：
  + - 列名的类型与修改的值要一致.
    - 修改值的时候不能超过最大长度.
    - 除了数值类型外，其它的字段类型的值必须使用引号引起

### 案例代码四

##更新语句 UPDATE

UPDATE category SET cid='c005' WHERE cname = '测试';

UPDATE category SET cname='食品'; #更新所有数据

### 删除记录：DELETE

* 语法：

DELETE from 表名 [WHERE 条件];

或者

truncate table 表名;

* 面试题：

删除表中所有记录使用DELETE from 表名; 还是用truncate table 表名;

删除方式：DELETE 一条一条删除，不清空auto\_increment记录数。

truncate 直接将表删除，重新建表，auto\_increment将置为零，从新开始。

### 案例代码五

##删除语句 DELETE

DELETE FROM category WHERE cid = 'c005';

## 删除所有，不清空自动增长列记录数。

DELETE FROM category;

## 删除所有，将清空自动增长列记录数。

TRUNCATE TABLE category;

## DOS命令窗口乱码解决方案

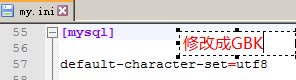
我们在dos命令行操作中文时，会报错



解决方案1：在cmd命令窗口中输入命令，此操作当前窗口有效，为临时方案。

SET names gbk;

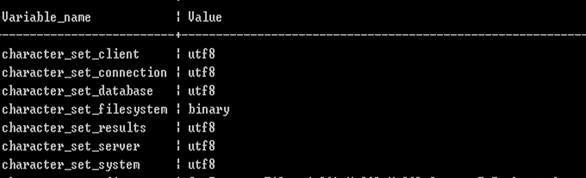
解决方案2：安装目录下修改my.ini文件，重启服务所有地方生效。[不建议]



错误原因:因为mysql的客户端设置编码是utf8,而系统的cmd窗口编码是gbk

1）查看MySQL内部设置的编码

show variables like 'character%'; 查看所有mysql的编码



2）修改client、connection、results的编码一致（GBK编码）

client connetion result 和客户端保持一致，都为GBK

* 将客户端编码修改为gbk.

#方式1：单独设置

SET character\_SET\_client=gbk;

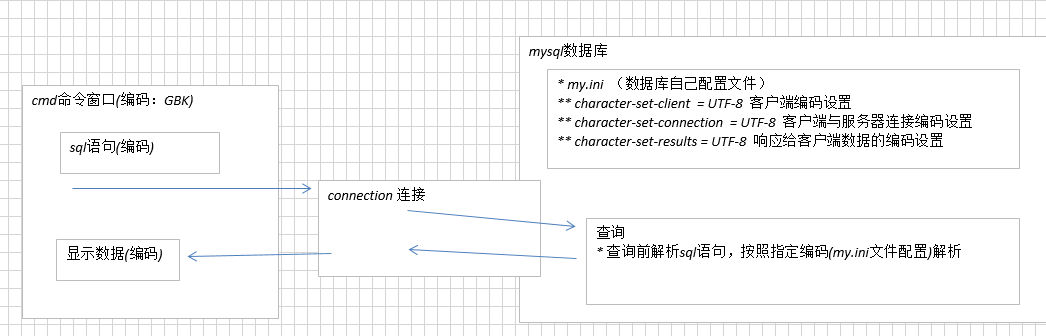
SET character\_SET\_connection=gbk;

SET character\_SET\_results=gbk;

#方式2：快捷设置

SET names gbk;

案例图片



## DQL--简单查询

### 准备工作

#创建商品表：

create table d\_product(

pid int primary key,

pname varchar(20),

price double,

category\_id varchar(32)

);

INSERT INTO `d\_product` VALUES ('1', '苹果', '5000', 'c001');

INSERT INTO `d\_product` VALUES ('2', 'vivo', '3000', 'c001');

INSERT INTO `d\_product` VALUES ('3', '华为', '5000', 'c001');

INSERT INTO `d\_product` VALUES ('4', 'oppo', '1000', 'c001');

INSERT INTO `d\_product` VALUES ('5', '真维斯', '200', 'c002');

INSERT INTO `d\_product` VALUES ('6', 'only', '440', 'c002');

INSERT INTO `d\_product` VALUES ('7', '圣迪奥', '800', 'c002');

INSERT INTO `d\_product` VALUES ('8', 'vero moda', '800', 'c003');

INSERT INTO `d\_product` VALUES ('9', '相宜本草', '200', 'c003');

INSERT INTO `d\_product` VALUES ('10', '卫龙辣条', '5', NULL);

INSERT INTO `d\_product` VALUES ('11', '好想你枣', '56', NULL);

INSERT INTO `d\_product` VALUES ('12', '香飘飘奶茶', '3', NULL);

INSERT INTO `d\_product` VALUES ('13', '小浣熊干脆面', '1', NULL);

### 语法

select [distinct]

\* | 列名,列名

from 表

WHERE 条件

### 简单查询\_列筛选

1.查询所有的商品. select \* from d\_product;

2.查询商品名和商品价格. select pname,price from d\_product;

3.别名查询.使用的关键字是as（as可以省略的）.

3.1表别名: select \* from d\_product as p;

3.2列别名：select pname as pn from d\_product;

4.去掉重复值. select distinct price from d\_product;

5.查询结果是表达式（运算查询）：将所有商品的价格打九折进行显示.

select pname,price\*0.9 as newPrice from d\_product;

### 案例代码六

/\*列筛选\*/

/\*1、查询出所有列\*/

select pid,pname,price,category\_id from d\_product;

select \* from d\_product;

/\*2、查询部分列\*/

select pid,pname from d\_product;

/\*3、给列起别名\*/

select pid as 商品id,pname 商品名称 from d\_product;

/\*4、给表起别名\*/

select pid,a.pname from d\_product a;

/\*5、列是表达式\*/

select pid,pname,price,category\_id from d\_product;

/\*6、去除重复项 distinct\*/

select distinct price from d\_product;

## DQL—行筛选

### 条件查询

|  |  |  |
| --- | --- | --- |
| 比较运算符 | > < <= >= = <> | 大于、小于、大于(小于)等于、不等于 |
| BETWEEN ...AND... | 显示在某一区间的值(含头含尾) |
| IN(SET) | 显示在in列表中的值，例：in(100,200) |
| IS NULL | 判断是否为空 |
| LIKE ‘张pattern’ | 模糊查询，Like语句中，  % 代表零个或多个任意字符，  \_ 代表一个任意字符，  例如：first\_name like ‘\_a%’; |
| 逻辑运算符 | and | 多个条件同时成立 |
| or | 多个条件任一成立 |
| not | 不成立，例：WHERE not(salary>100); |

#查询价格为800商品

SELECT \* FROM d\_product WHERE price = 800;

#查询价格不是800的所有商品

SELECT \* FROM d\_product WHERE price != 800;

SELECT \* FROM d\_product WHERE price <> 800;

SELECT \* FROM d\_product WHERE NOT(price = 800);

#查询商品价格大于800元的所有商品信息

SELECT \* FROM d\_product WHERE price > 800;

#查询商品价格在800到2000之间所有商品

SELECT \* FROM d\_product WHERE price >= 800 AND price <=2000;

SELECT \* FROM d\_product WHERE price BETWEEN 800 AND 2000;

#查询商品价格是200或800的所有商品

SELECT \* FROM d\_product WHERE price = 200 OR price = 800;

SELECT \* FROM d\_product WHERE price IN (200,800);

#查询含有'o'字的所有商品

SELECT \* FROM d\_product WHERE pname LIKE '%o%';

#查询以'o'开头的所有商品

SELECT \* FROM d\_product WHERE pname LIKE 'o%';

#查询第四个字为'o'的所有商品

SELECT \* FROM d\_product WHERE pname LIKE '\_\_\_o%';

#商品没有分类的商品

SELECT \* FROM d\_product WHERE category\_id IS NULL

#查询有分类的商品

SELECT \* FROM d\_product WHERE category\_id IS NOT NULL

### 案例代码七

/\*行筛选\*/

/\*1、比较运算符 = <> > >= < <=\*/

/\*筛选价格等于800的商品\*/

/\*select \* from d\_product where price=800;\*/

/\*筛选价格不等于800的商品\*/

/\*select \* from d\_product where price<>800;\*/

/\*筛选价格大于800的商品\*/

/\*select \* from d\_product where price>800;\*/

/\*筛选价格大于等于800同时小于等于3000的商品\*/

/\*select \* from d\_product where price >=800 and price<=3000;\*/

/\*2、使用关键字 between and\*/

/\*select \* from d\_product where price BETWEEN 800 and 3000;/\*注意数值的位置\*/

/\*筛选价格是200或者800的商品\*/

/\*select \* from d\_product where price=200 or price=800;\*/

/\*3、 in(Set) \*/

/\*select \* from d\_product where price in(200,800);\*/

/\*筛选价格既不是200也不是800的商品\*/

/\*select \* from d\_product where price not in(200,800);\*/

/\*4、空和非空\*/

/\*筛选出category\_id是NULL的商品\*/

/\*select \* from d\_product where category\_id is null;\*/

/\*筛选出category\_id不是NULL的商品\*/

/\*select \* from d\_product where category\_id is not null;\*/

/\*5、模糊查询like的用法 \*/

/\*商品名称中包含o \*/

/\*select \* from d\_product where pname like'%o%';\*/

/\*商品名称以o结尾\*/

/\*select \* from d\_product where pname like'%o';\*/

/\*商品名称XXXo\*/

/\*select \* from d\_product where pname like'\_\_\_o%';\*/

/\*6、组合情况\*/

/\*筛选出价格在800以上，同时包含o字符的商品id和名称、价格\*/

select pid,pname,price from d\_product where pname like'%o%' and price>800;