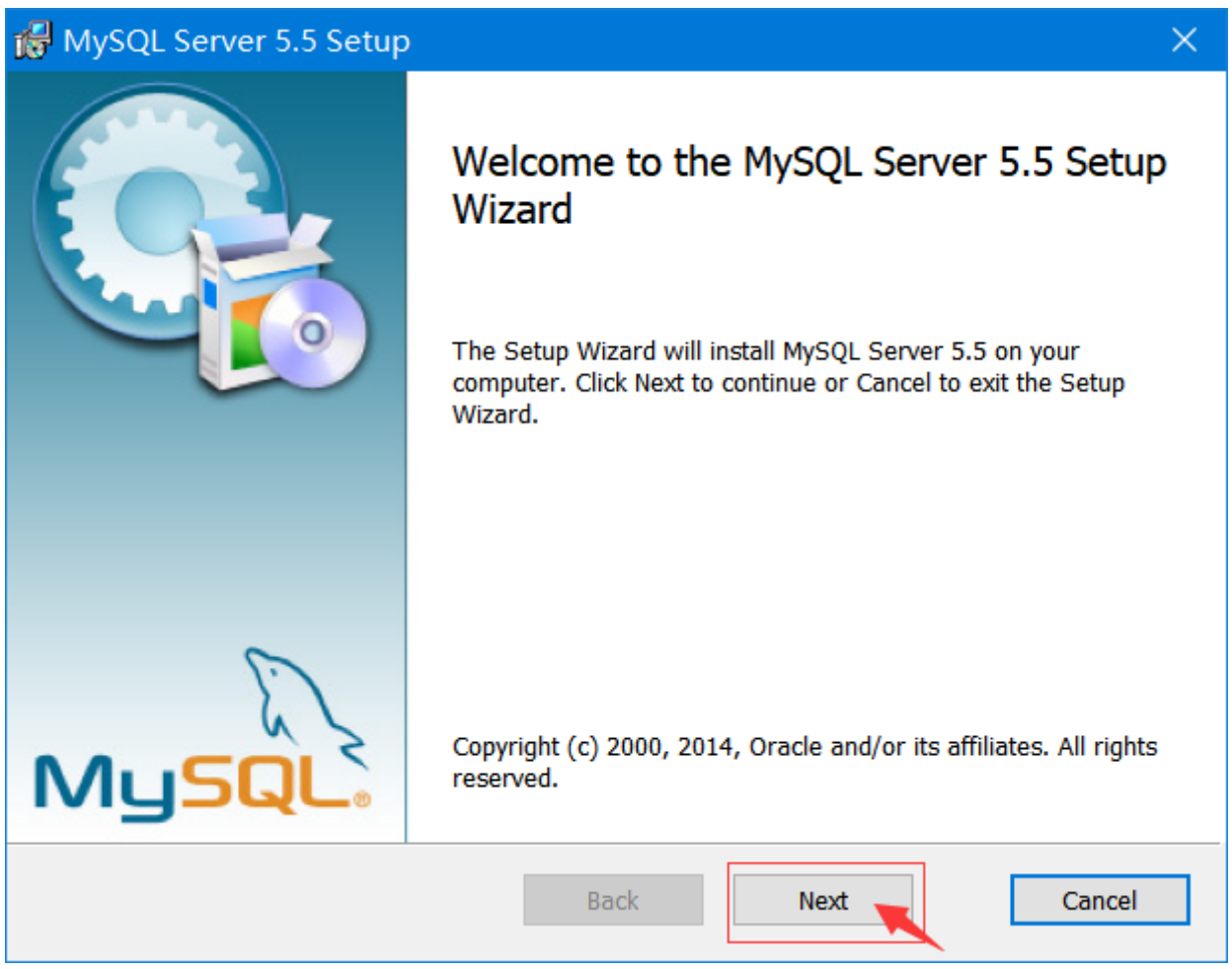
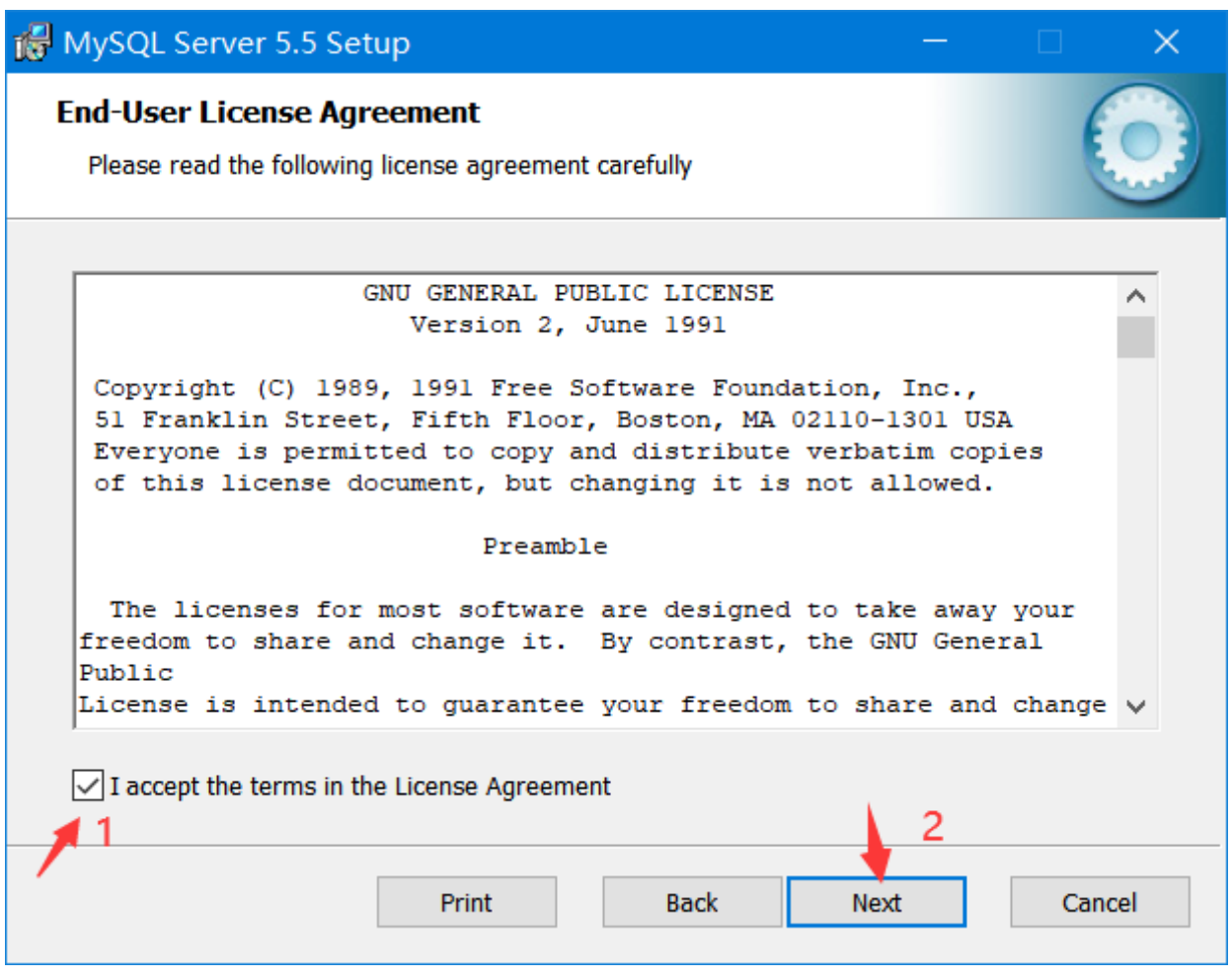
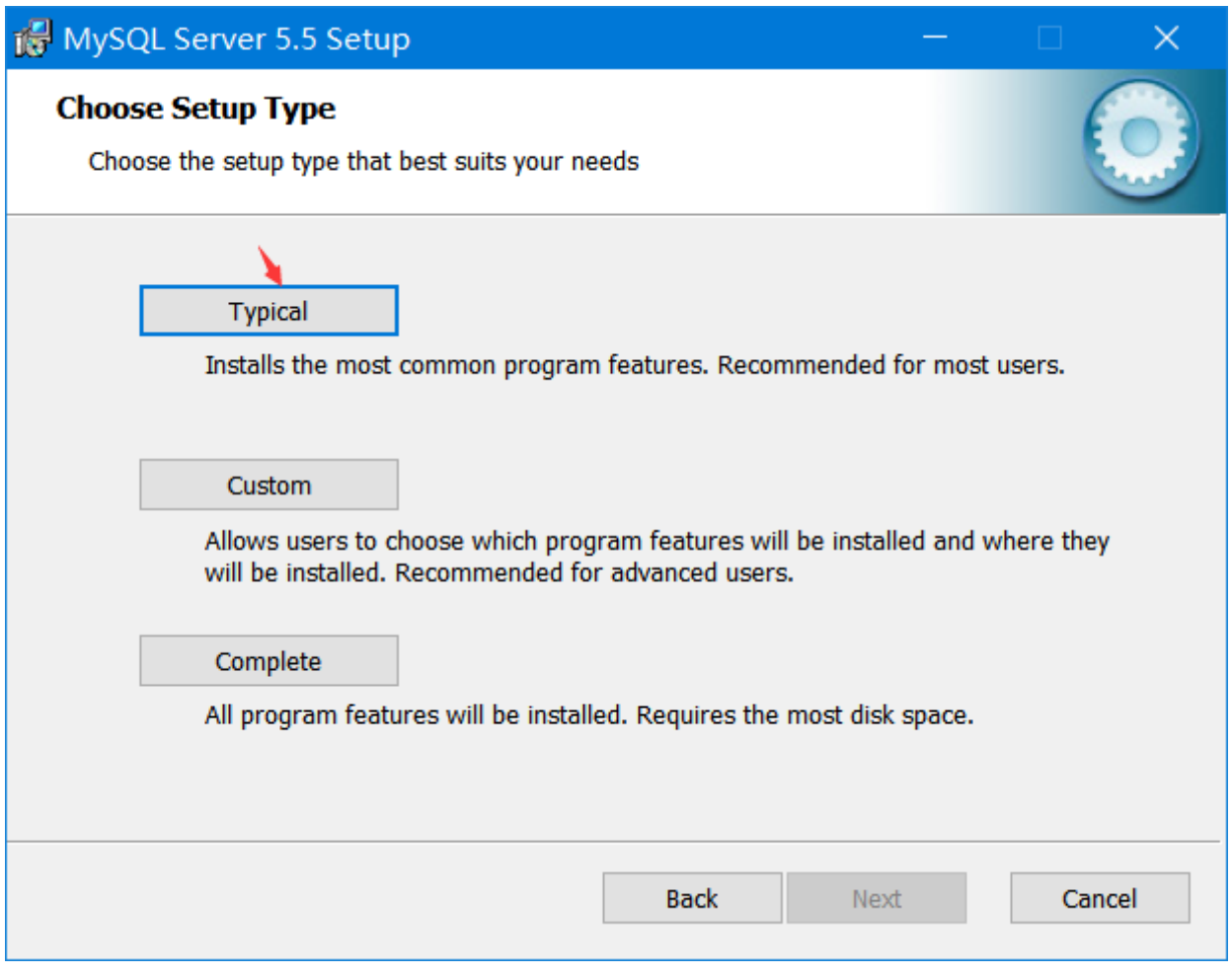
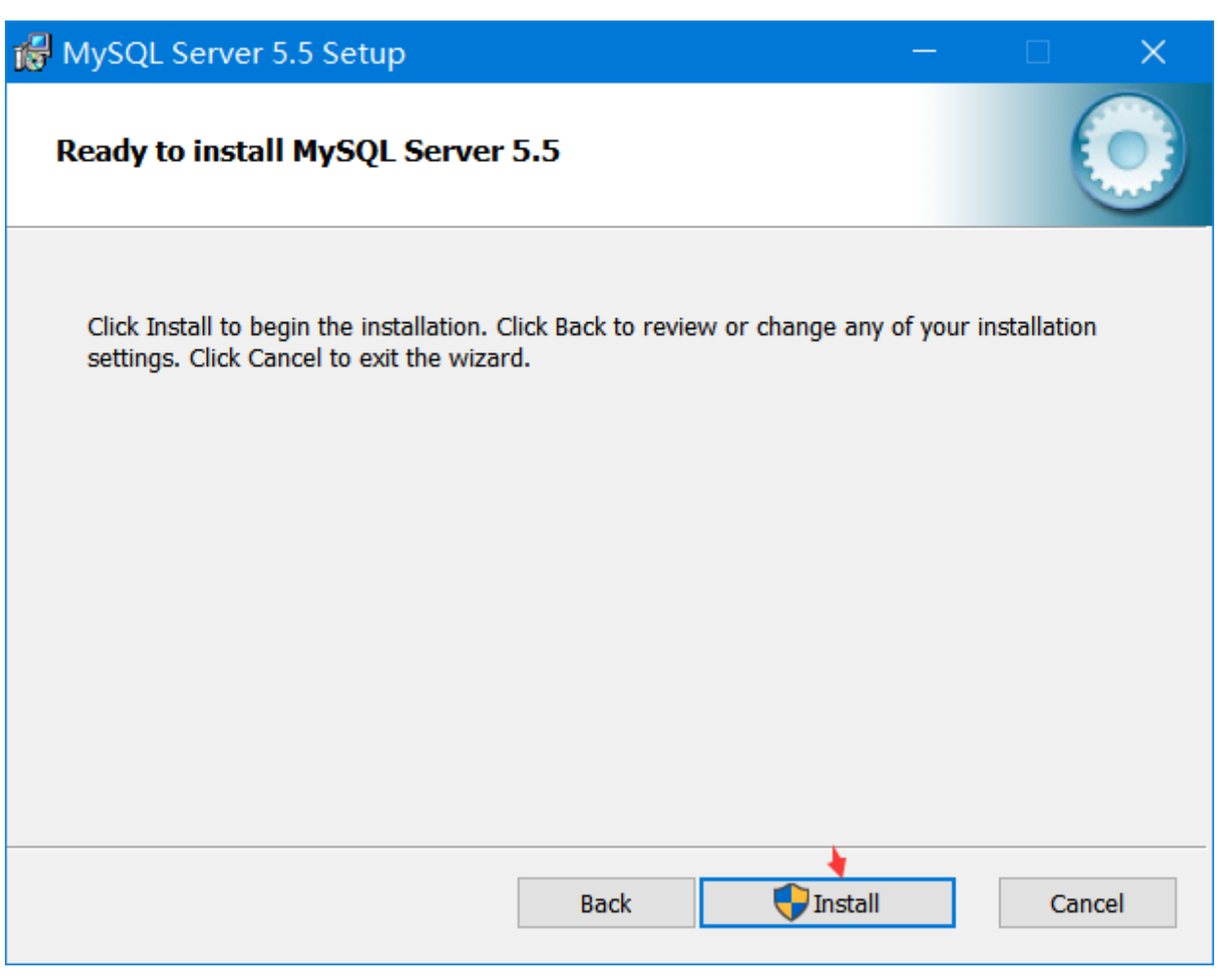
MySQL

一 *MySQL*的安装与配置

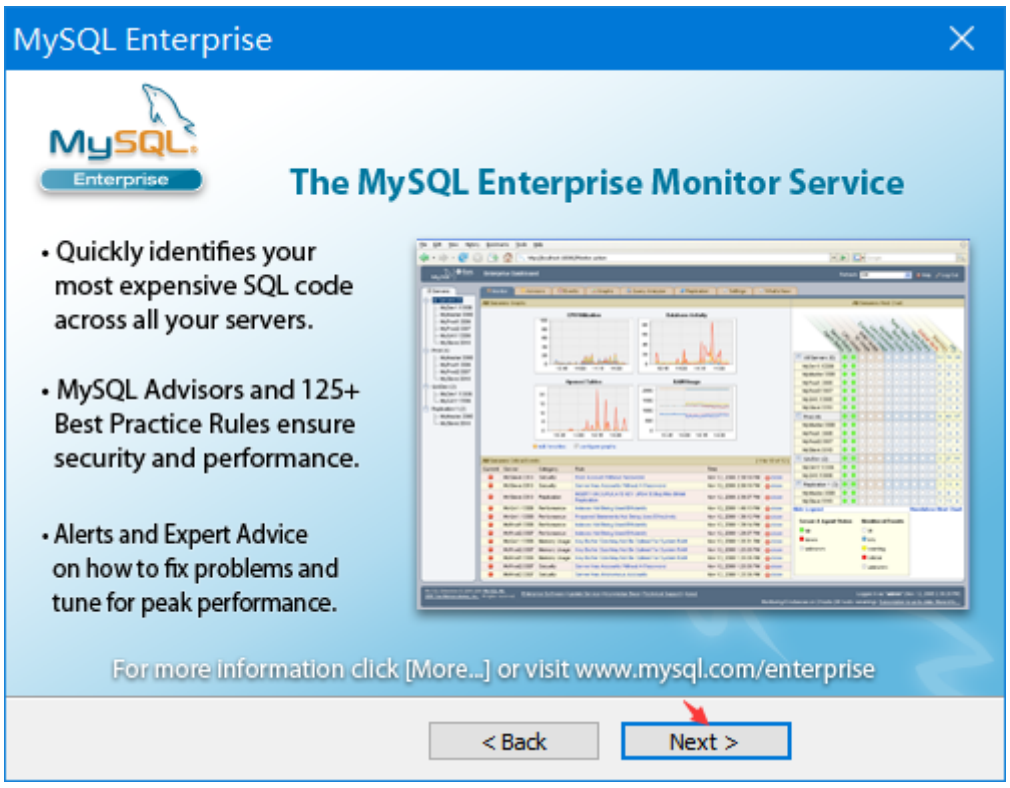


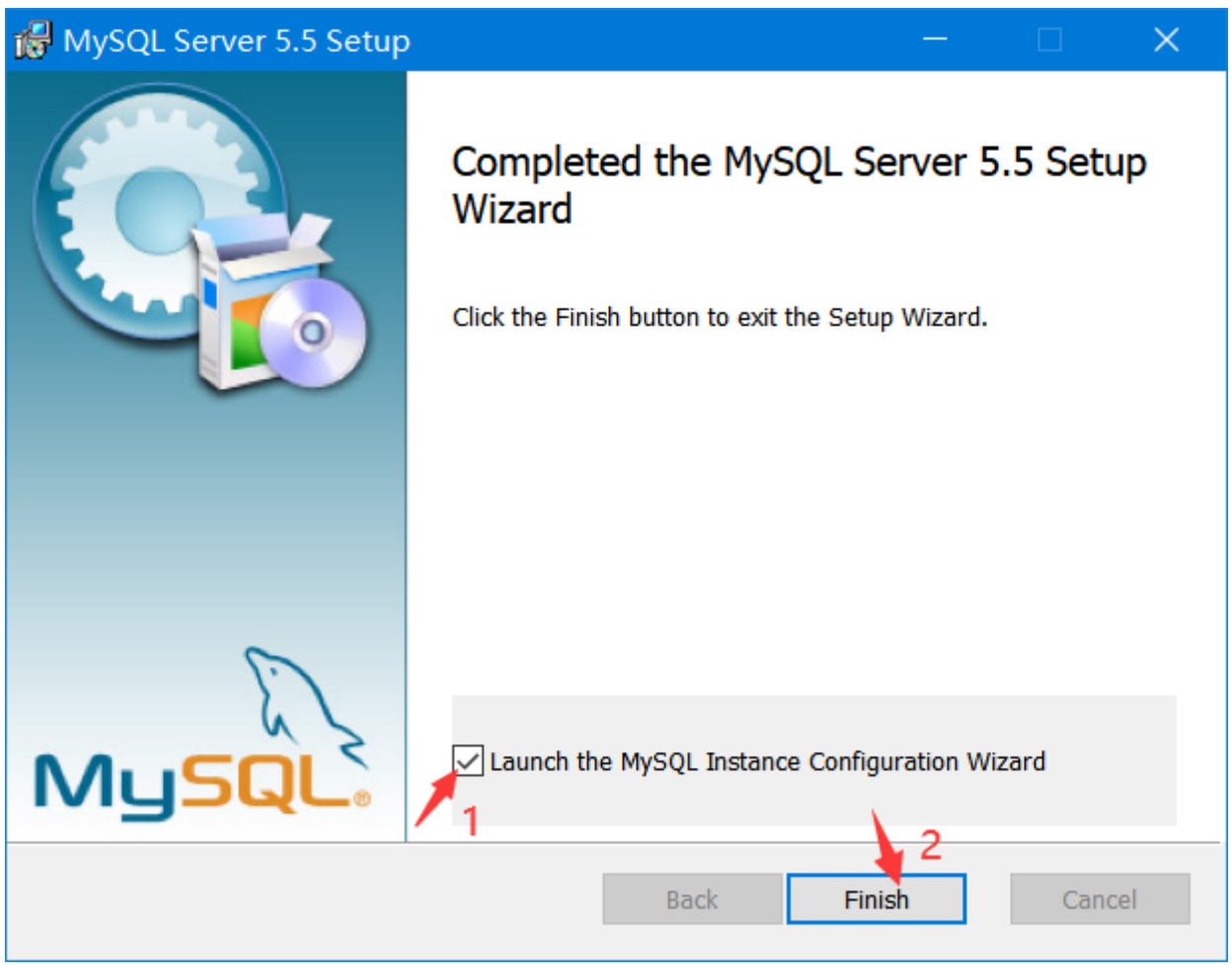


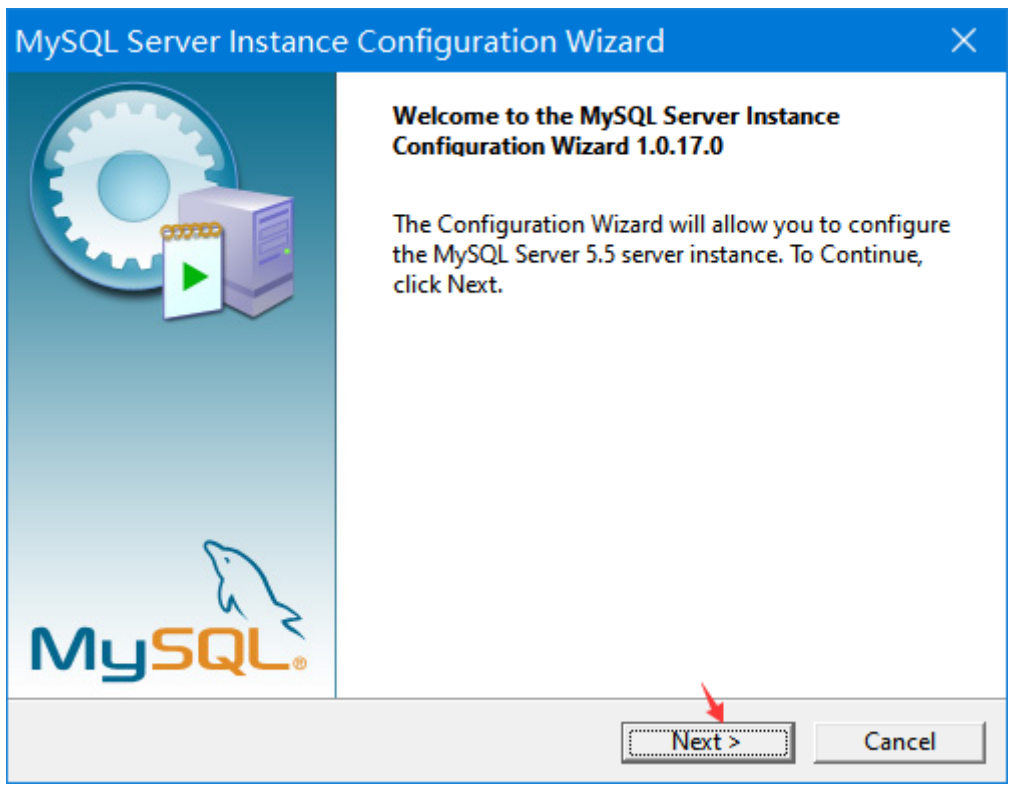

MySQL Server 5.5 Setup
Choose Setup Type
Choose the setup type that best suits your needs
Typical
Installs the most common program features. Recommended for most users.
Custom
Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.
Complete
All program features will be installed. Requires the most disk space.
Back
Next
Cancel


MySQL Server 5.5 Setup
Ready to install MySQL Server 5.5
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

MySQL Enterprise
MySQL
Enterprise
A MySQL Enterprise subscription is the most comprehensive offering of MySQL database software, services, and support to ensure your business achieves the highest levels of reliability, security, and uptime.
An Enterprise Subscription includes:
1. The MySQL Enterprise Server - The most reliable, secure, and up-to-date version of the world's most popular open source database.
2. MySQL Enterprise Monitor Service - An automated virtual database assistant.
3. MySQL Production Support - Technical and consultative support when you need it, along with service packs, hot-fixes, and more.
For more information click [More...] or visit www.mysql.com/enterprise
Next >


MySQL Enterprise
MySQL
Enterprise
The MySQL Enterprise Monitor Service
• Quickly identifies your most expensive SQL code across all your servers.
• MySQL Advisors and 125+ Best Practice Rules ensure security and performance.
• Alerts and Expert Advice on how to fix problems and tune for peak performance.
For more information click [More...] or visit www.mysql.com/enterprise
< Back
Next >


MySQL Server 5.5 Setup
Completed the MySQL Server 5.5 Setup Wizard
Click the Finish button to exit the Setup Wizard.
MySQL
Launch the MySQL Instance Configuration Wizard
1
2
Back
Finish
Cancel


MySQL Server Instance Configuration Wizard
Welcome to the MySQL Server Instance Configuration Wizard 1.0.17.0
The Configuration Wizard will allow you to configure the MySQL Server 5.5 server instance. To Continue, click Next.
MySQL
Next >
Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Please select a configuration type.

Detailed Configuration

1

Choose this configuration type to create the optimal server setup for this machine.

Standard Configuration

Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.

2

< Back | Next > | Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Please select a server type. This will influence memory, disk and CPU usage.

Developer Machine

This is a development machine, and many other applications will be run on it. MySQL Server should only use a minimal amount of memory.

Server Machine

Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage.

Dedicated MySQL Server Machine

This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.

< Back | Next > | Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Please select the database usage.

Multifunctional Database

General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.

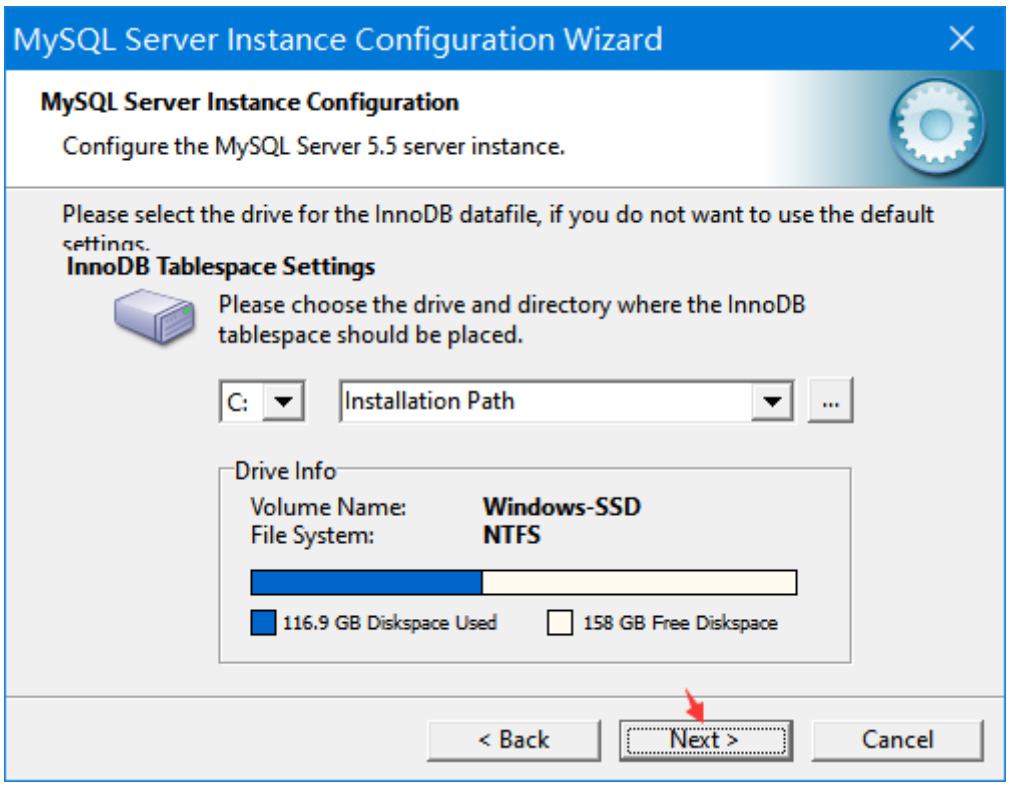
Transactional Database Only

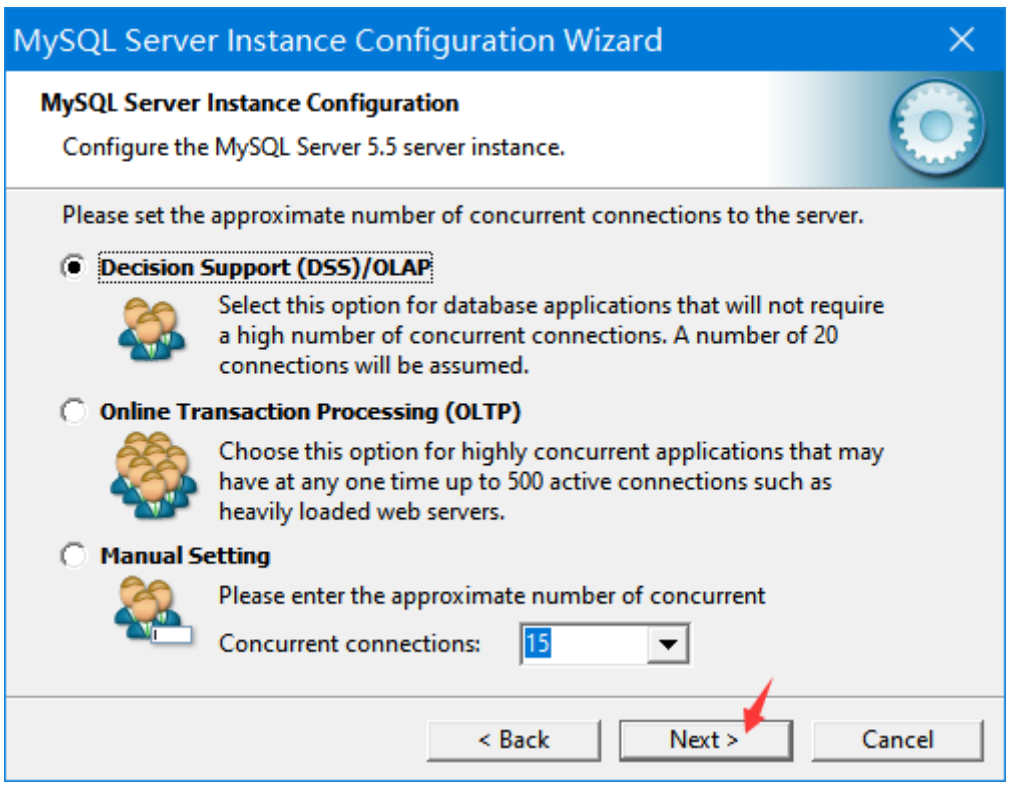
Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.

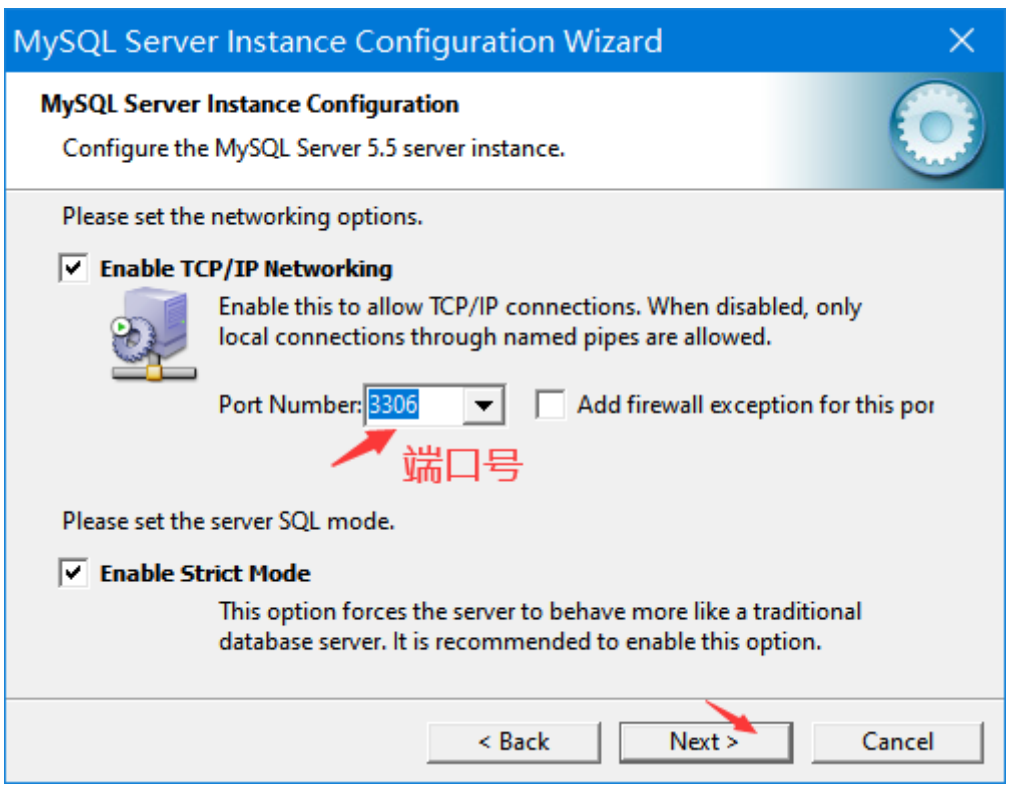
Non-Transactional Database Only

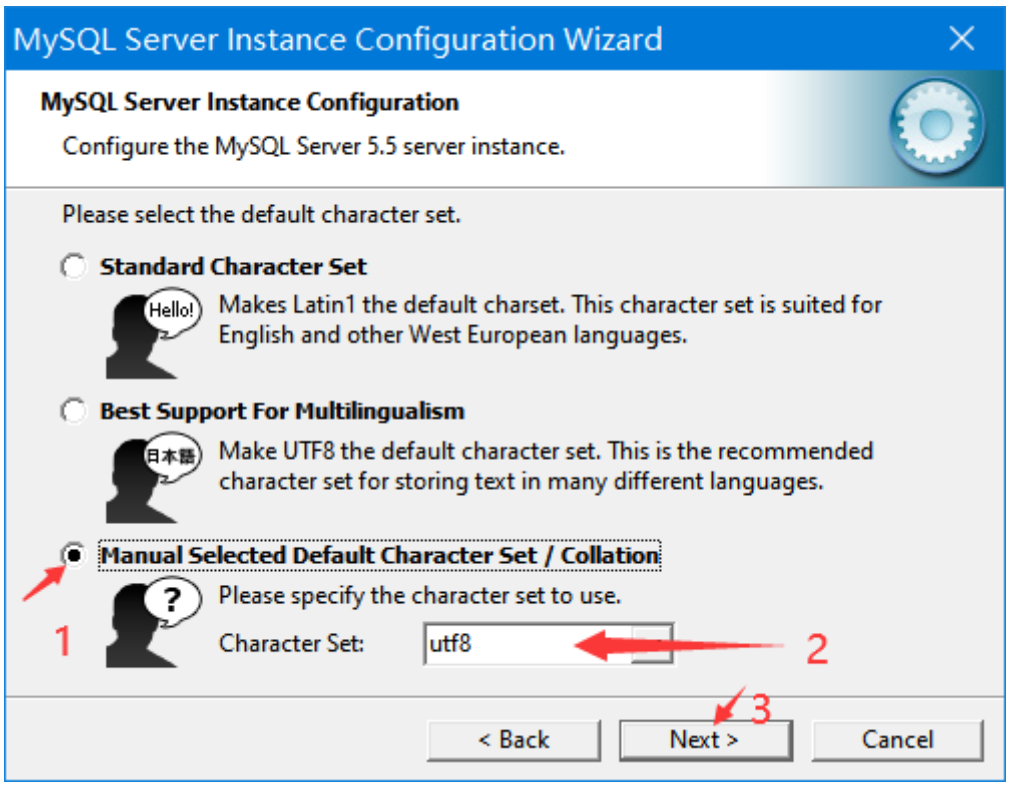
Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.

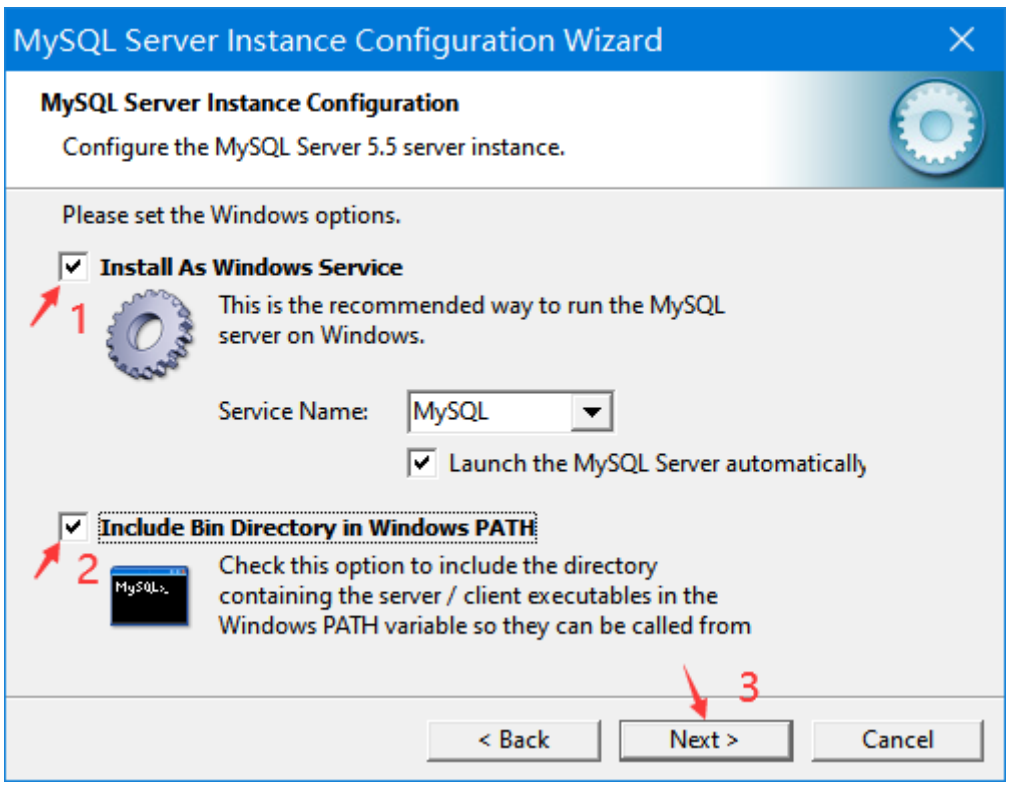
< Back | Next > | Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select the drive for the InnoDB datafile, if you do not want to use the default settings.
InnoDB Tablespace Settings
Please choose the drive and directory where the InnoDB tablespace should be placed.
C:
Installation Path
...
Drive Info
Volume Name: Windows-SSD
File System: NTFS
116.9 GB Diskspace Used
158 GB Free Diskspace
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the approximate number of concurrent connections to the server.
Decision Support (DSS)/OLAP
Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.
Online Transaction Processing (OLTP)
Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.
Manual Setting
Please enter the approximate number of concurrent
Concurrent connections: 15
< Back
Next >
Cancel

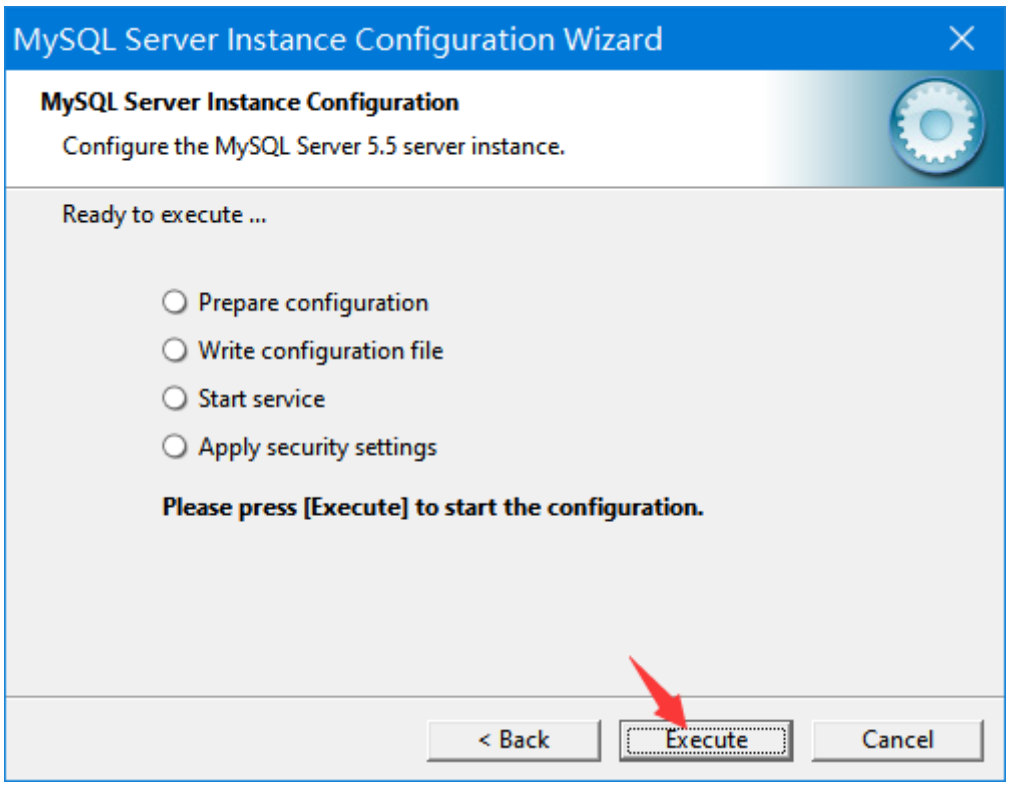

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the networking options.
Enable TCP/IP Networking
Enable this to allow TCP/IP connections. When disabled, only local connections through named pipes are allowed.
Port Number: 3306
Add firewall exception for this por
端口号
Please set the server SQL mode.
Enable Strict Mode
This option forces the server to behave more like a traditional database server. It is recommended to enable this option.
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select the default character set.
Standard Character Set
Hello!
Makes Latin1 the default charset. This character set is suited for English and other West European languages.
Best Support For Multilingualism
日本語
Make UTF8 the default character set. This is the recommended character set for storing text in many different languages.
Manual Selected Default Character Set / Collation
Please specify the character set to use.
Character Set:
utf8
1
2
3
< Back
Next >
Cancel

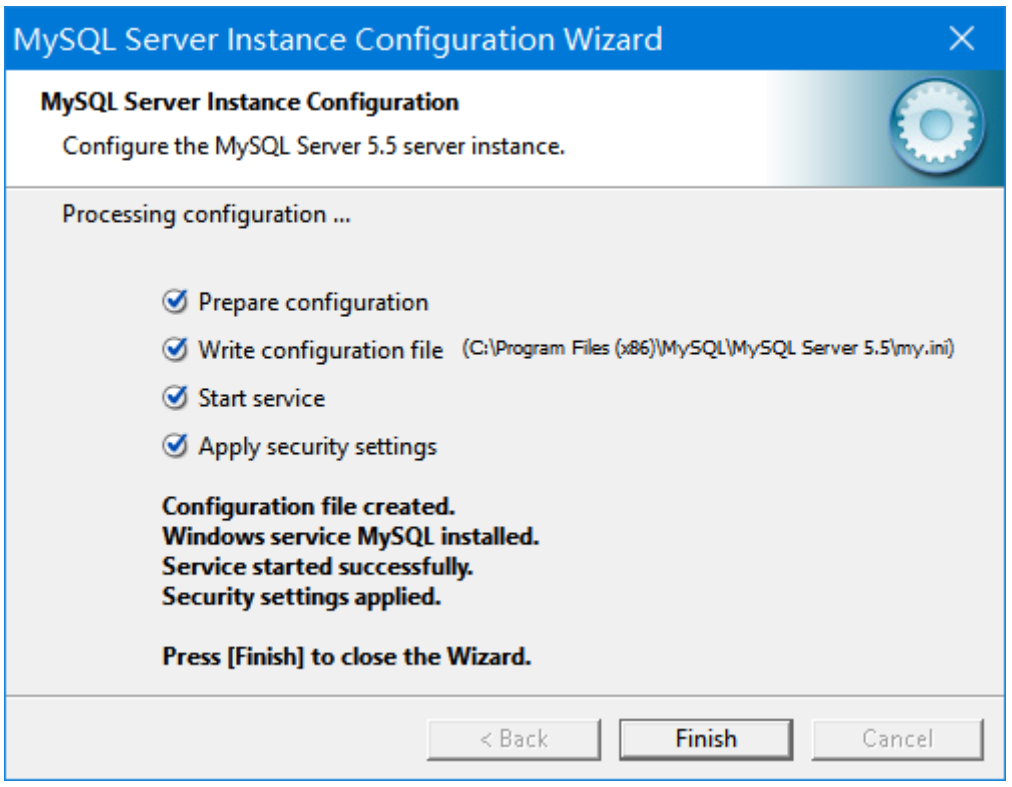

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the Windows options.
Install As Windows Service
1
This is the recommended way to run the MySQL server on Windows.
Service Name:
MySQL
Launch the MySQL Server automatically
Include Bin Directory in Windows PATH
2
MySQL>_
Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from
3
< Back
Next >
Cancel

密码：111


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the security options.
Modify Security Settings
管理员密码
root
New root password:
******
Enter the root password.
Confirm:
******
Retype the password.
Enable root access from remote machines
Create An Anonymous Account
This option will create an anonymous account on this server.
Please note that this can lead to an insecure system.
< Back
Next >
Cancel



以下表示安装完成



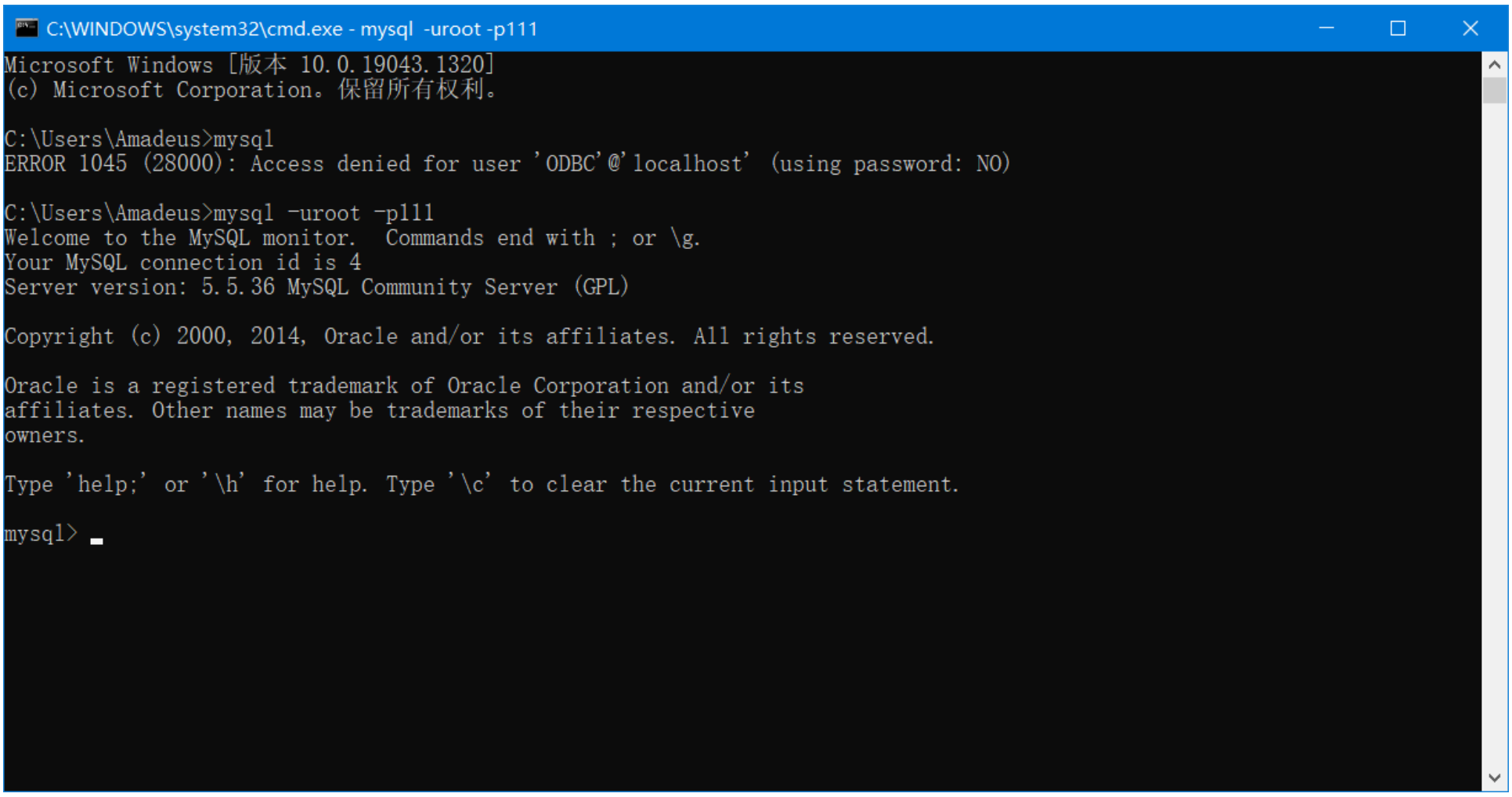
二 完美卸载*MySQL*

```
1、双击安装包，点击下一步，然后点击remove。卸载。
2、手动删除Program Files(x86)中的MySQL目录。
3、手动删除ProgramData目录（这个目录是隐藏的。）中的MySQL目录。
```

三 登录*MySQL*

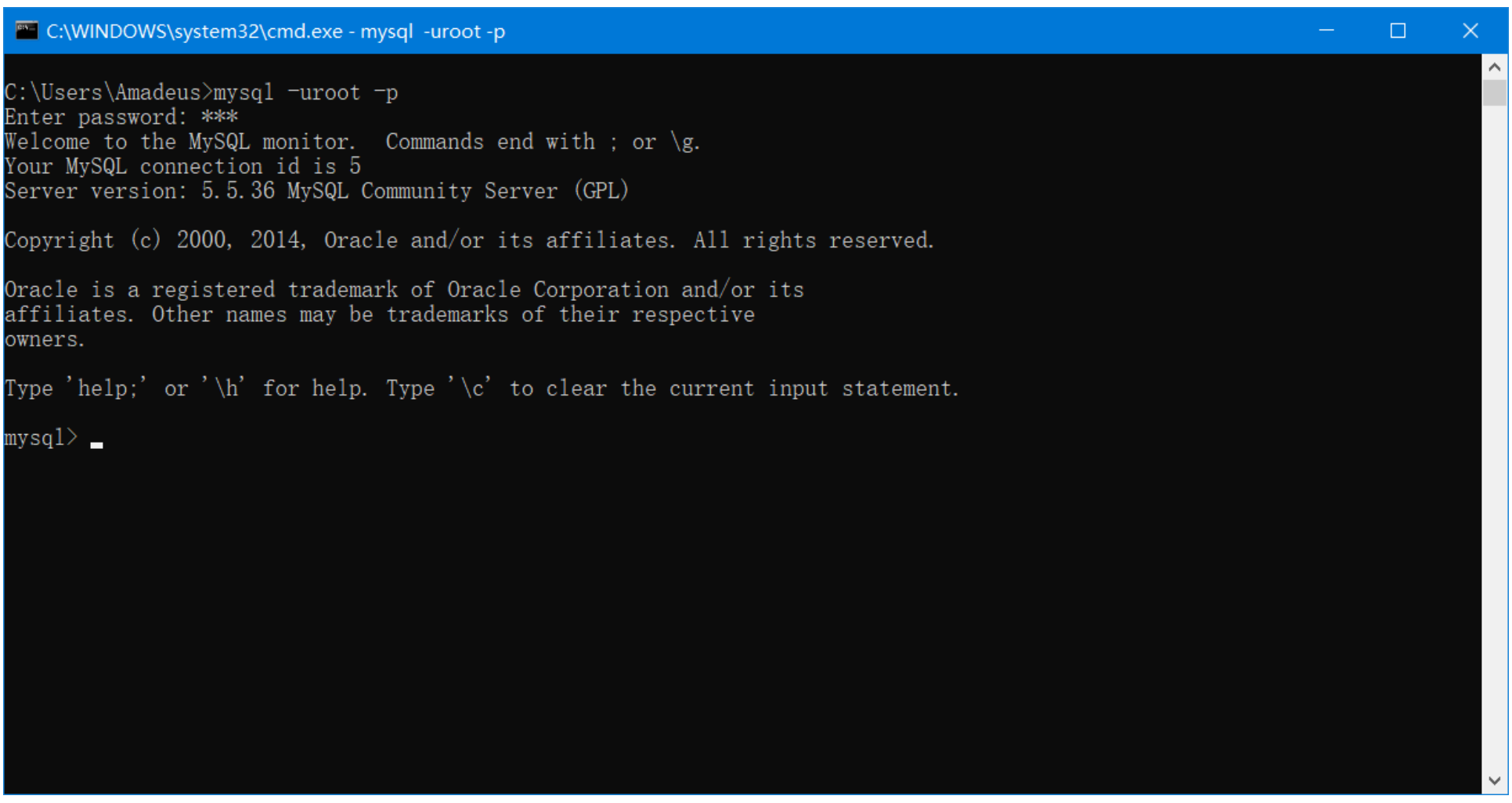
```
mysql -uroot -p111(回车)
#-u表示用户名  -p表示密码

mysql>
```




```
mysql -uroot -p(回车)
Enter password:***(回车)

mysql>
```



四 更改*root*密码

```
方法1： 用SET PASSWORD命令
首先登录MySQL。
格式：mysql> set password for 用户名@localhost = password('新密码');
例子：mysql> set password for root@localhost = password('123');

方法2：用mysqladmin
格式：mysqladmin -u用户名 -p旧密码 password 新密码
例子：mysqladmin -uroot -p123456 password 123

方法3：用UPDATE直接编辑user表
首先登录MySQL。
mysql> use mysql;
mysql> update user set password=password('123') where user='root' and host='localhost';
mysql> flush privileges;

方法4：在忘记root密码的时候，可以这样
```

以windows为例：
1. 关闭正在运行的MySQL服务。
2. 打开DOS窗口，转到mysql\bin目录。
3. 输入mysqld --skip-grant-tables 回车。--skip-grant-tables 的意思是启动MySQL服务的时候跳过权限表认证。
4. 再开一个DOS窗口（因为刚才那个DOS窗口已经不能动了），转到mysql\bin目录。
5. 输入mysql回车，如果成功，将出现MySQL提示符 >。
6. 连接权限数据库： use mysql; 。
6. 改密码：update user set password=password("123") where user="root";（别忘了最后加分号） 。
7. 刷新权限（必须步骤）：flush privileges; 。
8. 退出 quit。
9. 注销系统，再进入，使用用户名root和刚才设置的新密码123登录。

五 课堂笔记

day01

1 sql、DB、DBMS之间的关系

```
1、sql、DB、DBMS分别是什么，他们之间的关系？
    DB:
        DataBase（数据库，数据库实际上在硬盘上以文件的形式存在）
    DBMS:
        DataBase Management System（数据库管理系统，常见的有：MySQL Oracle DB2 Sybase SqlServer...）
    SQL:
        结构化查询语言，是一门标准通用的语言。标准的sql适合于所有的数据库产品。
        SQL属于高级语言。只要能看懂英语单词的，写出来的sql语句，可以读懂什么意思。
        SQL语句在执行的时候，实际上内部也会先进行编译，然后再执行sql。（sql语句的编译由DBMS完成。）
    DBMS负责执行sql语句，通过执行sql语句来操作DB当中的数据。
    DBMS -(执行)-> SQL -(操作)-> DB
```

2 对表的理解

```
2、对表的理解
    表：table
    表：table是数据库的基本组成单元，所有的数据都以表格的形式组织，目的是可读性强。
    一个表包括行和列：
        行：被称为数据/记录(data)
        列：被称为字段(column)

    学号(int)    姓名(varchar)     年龄(int)
    ------------------------------------------
    110          张三              20
    120          李四              21

    每一个字段应该包括哪些属性？
        字段名、数据类型、相关的约束、字段长度
```

3 SQL语句的分类

```
3、学习MySQL主要还是学习通用的SQL语句，那么SQL语句包括增删改查，SQL语句怎么分类呢？
    DQL（数据查询语言）：
        查询语句，凡是select语句都是DQL。
    DML（数据操作语言）：
        insert delete update，对表当中的数据进行增删改。
    DDL（数据定义语言）：
        create drop alter，对表结构的增删改。
    TCL（事务控制语言）：
        commit提交事务，rollback回滚事务。(TCL中的T是Transaction)
    DCL（数据控制语言）：
        grant授权、revoke撤销权限等。
```

4 导入数据

```
4、导入数据（后期大家练习的时候使用这个演示的数据）
    第一步：登录mysql数据库管理系统
        dos命令窗口：
            mysql -uroot -p111
    第二步：查看有哪些数据库
        show databases; (这个不是SQL语句，属于MySQL的命令。)
        +--------------------+
        | Database           |
        +--------------------+
        | information_schema |
        | mysql              |
        | performance_schema |
        | test               |
        +--------------------+
    第三步：创建属于我们自己的数据库
        create database bjpowernode; (这个不是SQL语句，属于MySQL的命令。)
    第四步：使用bjpowernode数据
```

```
        use bjpowernode; (这个不是SQL语句，属于MySQL的命令。)
    第五步：查看当前使用的数据库中有哪些表？
        show tables; (这个不是SQL语句，属于MySQL的命令。)
    第六步：初始化数据
        mysql> source D:\course\05-MySQL\resources\bjpowernode.sql

    注意：数据初始化完成之后，有三张表：
    +-----------------------+
    | Tables_in_bjpowernode |
    +-----------------------+
    | dept                  |
    | emp                   |
    | salgrade              |
    +-----------------------+
```

5 什么是sql脚本

6 删除数据库

```
5、bjpowernode.sql，这个文件以sql结尾，这样的文件被称为“sql脚本”。什么是sql脚本呢？
    当一个文件的扩展名是.sql，并且该文件中编写了大量的sql语句，我们称这样的文件为sql脚本。
    注意：直接使用source命令可以执行sql脚本。
    sql脚本中的数据量太大的时候，无法打开，请使用source命令完成初始化。

6、删除数据库：drop database bjpowernode;
```

7 查看表结构

```
7、查看表结构：
    mysql->show tables;#查看bjpowernode数据库中有哪些表
    +-----------------------+
    | Tables_in_bjpowernode |
    +-----------------------+
    | dept                  |   (部门表)
    | emp                   |   (员工表)
    | salgrade              |   (工资等级表)
    +-----------------------+

    mysql> desc dept;#desc 表名 #查看表结构
    +--------+-------------+------+-----+---------+-------+
    | Field  | Type        | Null | Key | Default | Extra |
    +--------+-------------+------+-----+---------+-------+
    | DEPTNO | int(2)      | NO   | PRI | NULL    |       |     部门编号
    | DNAME  | varchar(14) | YES  |     | NULL    |       |     部门名称
    | LOC    | varchar(13) | YES  |     | NULL    |       |     部门位置
    +--------+-------------+------+-----+---------+-------+

    mysql> desc emp;
    +----------+-------------+------+-----+---------+-------+
    | Field    | Type        | Null | Key | Default | Extra |
    +----------+-------------+------+-----+---------+-------+
    | EMPNO    | int(4)      | NO   | PRI | NULL    |       |   员工编号
    | ENAME    | varchar(10) | YES  |     | NULL    |       |   员工姓名
    | JOB      | varchar(9)  | YES  |     | NULL    |       |   工作岗位
    | MGR      | int(4)      | YES  |     | NULL    |       |   上级领导编号
    | HIREDATE | date        | YES  |     | NULL    |       |   入职日期
    | SAL      | double(7,2) | YES  |     | NULL    |       |   月薪
    | COMM     | double(7,2) | YES  |     | NULL    |       |   补助/津贴
    | DEPTNO   | int(2)      | YES  |     | NULL    |       |   部门编号
    +----------+-------------+------+-----+---------+-------+

    mysql> desc salgrade;
    +-------+---------+------+-----+---------+-------+
    | Field | Type    | Null | Key | Default | Extra |
    +-------+---------+------+-----+---------+-------+
    | GRADE | int(11) | YES  |     | NULL    |       |     等级
    | LOSAL | int(11) | YES  |     | NULL    |       |     最低薪资
    | HISAL | int(11) | YES  |     | NULL    |       |     最高薪资
    +-------+---------+------+-----+---------+-------+
```

8 查看表中的数据

```
8、表中的数据？
mysql> select * from emp;
+-------+--------+-----------+------+------------+---------+---------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+--------+-----------+------+------------+---------+---------+--------+
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-17 |  800.00 |    NULL |     20 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-20 | 1600.00 |  300.00 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-22 | 1250.00 |  500.00 |     30 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-02 | 2975.00 |    NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-28 | 1250.00 | 1400.00 |     30 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850.00 |    NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-06-09 | 2450.00 |    NULL |     10 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1987-04-19 | 3000.00 |    NULL |     20 |
```

```
| 7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000.00 |    NULL |     10 |
| 7844 | TURNER | SALESMAN  | 7698 | 1981-09-08 | 1500.00 |    0.00 |     30 |
| 7876 | ADAMS  | CLERK     | 7788 | 1987-05-23 | 1100.00 |    NULL |     20 |
| 7900 | JAMES  | CLERK     | 7698 | 1981-12-03 |  950.00 |    NULL |     30 |
| 7902 | FORD   | ANALYST   | 7566 | 1981-12-03 | 3000.00 |    NULL |     20 |
| 7934 | MILLER | CLERK     | 7782 | 1982-01-23 | 1300.00 |    NULL |     10 |
+-------+--------+-----------+------+------------+---------+---------+--------+

mysql> select * from dept;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+

mysql> select * from salgrade;
+-------+-------+-------+
| GRADE | LOSAL | HISAL |
+-------+-------+-------+
|     1 |   700 |  1200 |
|     2 |  1201 |  1400 |
|     3 |  1401 |  2000 |
|     4 |  2001 |  3000 |
|     5 |  3001 |  9999 |
+-------+-------+-------+
```

9 常用命令

```
9、常用命令
mysql> select database(); #查看当前使用的是哪个数据库
+-------------+
| database()  |
+-------------+
| bjpowernode |
+-------------+

mysql> select version(); #查看mysql的版本号。
+-----------+
| version() |
+-----------+
| 5.5.36    |
+-----------+

\c   命令，结束一条语句。

exit; 命令，退出mysql。
```

10 查看创建表的语句

```
10、查看创建表的语句：
    show create table emp;
mysql> show create table emp
| Table | Create Table
+-----------------------------------------------+
| emp   | CREATE TABLE `emp` (
  `EMPNO` int(4) NOT NULL,
  `ENAME` varchar(10) DEFAULT NULL,
  `JOB` varchar(9) DEFAULT NULL,
  `MGR` int(4) DEFAULT NULL,
  `HIREDATE` date DEFAULT NULL,
  `SAL` double(7,2) DEFAULT NULL,
  `COMM` double(7,2) DEFAULT NULL,
  `DEPTNO` int(2) DEFAULT NULL,
  PRIMARY KEY (`EMPNO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

11 简单的查询语句（DQL）

```
11、简单的查询语句（DQL）
语法格式：
    select 字段名1,字段名2,字段名3,.... from 表名;
提示：
    1、任何一条sql语句以“;”结尾。
    2、sql语句不区分大小写。
查询员工的年薪？（字段可以参与数学运算。）
    select ename,sal * 12 from emp;
        +--------+----------+
        | ename  | sal * 12 |
```

```
        +--------+----------+
        | SMITH  |  9600.00 |
        | ALLEN  | 19200.00 |
        | WARD   | 15000.00 |
        | JONES  | 35700.00 |
        | MARTIN | 15000.00 |
        | BLAKE  | 34200.00 |
        | CLARK  | 29400.00 |
        | SCOTT  | 36000.00 |
        | KING   | 60000.00 |
        | TURNER | 18000.00 |
        | ADAMS  | 13200.00 |
        | JAMES  | 11400.00 |
        | FORD   | 36000.00 |
        | MILLER | 15600.00 |
        +--------+----------+

给查询结果的列重命名?
    select ename,sal * 12 as yearsal from emp;
别名中有中文?
    select ename,sal * 12 as 年薪 from emp; // 错误
    select ename,sal * 12 as '年薪' from emp;
        +--------+----------+
        | ename  | 年薪       |
        +--------+----------+
        | SMITH  |  9600.00 |
        | ALLEN  | 19200.00 |
        | WARD   | 15000.00 |
        | JONES  | 35700.00 |
        | MARTIN | 15000.00 |
        | BLAKE  | 34200.00 |
        | CLARK  | 29400.00 |
        | SCOTT  | 36000.00 |
        | KING   | 60000.00 |
        | TURNER | 18000.00 |
        | ADAMS  | 13200.00 |
        | JAMES  | 11400.00 |
        | FORD   | 36000.00 |
        | MILLER | 15600.00 |
        +--------+----------+
# 注意：标准sql语句中要求字符串使用单引号括起来。虽然mysql支持双引号，尽量别用。

as关键字可以省略?
    mysql> select empno,ename,sal * 12 yearsal from emp;
        +-------+--------+----------+
        | empno | ename  | yearsal  |
        +-------+--------+----------+
        |  7369 | SMITH  |  9600.00 |
        |  7499 | ALLEN  | 19200.00 |
        |  7521 | WARD   | 15000.00 |
        |  7566 | JONES  | 35700.00 |
        |  7654 | MARTIN | 15000.00 |
        |  7698 | BLAKE  | 34200.00 |
        |  7782 | CLARK  | 29400.00 |
        |  7788 | SCOTT  | 36000.00 |
        |  7839 | KING   | 60000.00 |
        |  7844 | TURNER | 18000.00 |
        |  7876 | ADAMS  | 13200.00 |
        |  7900 | JAMES  | 11400.00 |
        |  7902 | FORD   | 36000.00 |
        |  7934 | MILLER | 15600.00 |
        +-------+--------+----------+

查询所有字段?
    select * from emp; // 实际开发中不建议使用*，效率较低。
```

12 条件查询

```
12、条件查询。
语法格式:
    select
        字段,字段...
    from
        表名
    where
        条件;
#执行顺序：先from，然后where，最后select

查询工资等于5000的员工姓名?
    select ename from emp where sal = 5000;
        +-------+
        | ename |
        +-------+
        | KING  |
        +-------+
查询SMITH的工资?
    select sal from emp where ename = 'SMITH'; # 字符串使用单引号括起来。
```

```
        +--------+
        | sal    |
        +--------+
        | 800.00 |
        +--------+
找出工资高于3000的员工？
    select ename,sal from emp where sal > 3000;
找出工资不低于3000的员工？
    select ename,sal from emp where sal >= 3000;
找出工资低于3000的员工？
    select ename,sal from emp where sal < 3000;
找出工资不高于3000的员工？
    select ename,sal from emp where sal <= 3000;

找出工资不等于3000的？
    select ename,sal from emp where sal <> 3000;
    select ename,sal from emp where sal != 3000;

找出工资在1100和3000之间的员工，包括1100和3000？
    select ename,sal from emp where sal >= 1100 and sal <= 3000;
    select ename,sal from emp where sal between 1100 and 3000;# between...and...是闭区间 [1100 ~ 3000]
    select ename,sal from emp where sal between 3000 and 1100; # 查询不到任何数据 [3000~1100]
    # between and在使用的时候必须左小右大。
    # between and除了可以使用在数字方面之外，还可以使用在字符串方面。
    select ename from emp where ename between 'A' and 'C';
        +-------+
        | ename |
        +-------+
        | ALLEN |
        | BLAKE |
        | ADAMS |
        +-------+
    select ename from emp where ename between 'A' and 'D'; # 左闭右开。
        +-------+
        | ename |
        +-------+
        | ALLEN |
        | BLAKE |
        | CLARK |
        | ADAMS |
        +-------+

    找出哪些人津贴为NULL？
        # 在数据库当中NULL不是一个值，代表什么也没有，为空。
        # 空不是一个值，不能用等号衡量。
        # 必须使用 is null或者is not null
        select ename,sal,comm from emp where comm is null;
            +--------+---------+------+
            | ename  | sal     | comm |
            +--------+---------+------+
            | SMITH  |  800.00 | NULL |
            | JONES  | 2975.00 | NULL |
            | BLAKE  | 2850.00 | NULL |
            | CLARK  | 2450.00 | NULL |
            | SCOTT  | 3000.00 | NULL |
            | KING   | 5000.00 | NULL |
            | ADAMS  | 1100.00 | NULL |
            | JAMES  |  950.00 | NULL |
            | FORD   | 3000.00 | NULL |
            | MILLER | 1300.00 | NULL |
            +--------+---------+------+
        select ename,sal,comm from emp where comm = null;
        Empty set (0.00 sec) #失效

    找出哪些人津贴不为NULL？
        select ename,sal,comm from emp where comm is not null;
            +--------+---------+---------+
            | ename  | sal     | comm    |
            +--------+---------+---------+
            | ALLEN  | 1600.00 |  300.00 |
            | WARD   | 1250.00 |  500.00 |
            | MARTIN | 1250.00 | 1400.00 |
            | TURNER | 1500.00 |    0.00 |
            +--------+---------+---------+

    找出哪些人没有津贴？
        select ename,sal,comm from emp where comm is null or comm = 0;
            +--------+---------+------+
            | ename  | sal     | comm |
            +--------+---------+------+
            | SMITH  |  800.00 | NULL |
            | JONES  | 2975.00 | NULL |
            | BLAKE  | 2850.00 | NULL |
            | CLARK  | 2450.00 | NULL |
            | SCOTT  | 3000.00 | NULL |
            | KING   | 5000.00 | NULL |
            | TURNER | 1500.00 | 0.00 |
            | ADAMS  | 1100.00 | NULL |
```

```
| JAMES  |  950.00 | NULL |
| FORD   | 3000.00 | NULL |
| MILLER | 1300.00 | NULL |
+--------+---------+------+
```

找出工作岗位是MANAGER和SALESMAN的员工？

```
select ename,job from emp where job = 'MANAGER' or job = 'SALESMAN';
```

```
+--------+----------+
| ename  | job      |
+--------+----------+
| ALLEN  | SALESMAN |
| WARD   | SALESMAN |
| JONES  | MANAGER  |
| MARTIN | SALESMAN |
| BLAKE  | MANAGER  |
| CLARK  | MANAGER  |
| TURNER | SALESMAN |
+--------+----------+
```

and 和 or 联合起来用：找出薪资大于1000的并且部门编号是20或30部门的员工。

```
select ename,sal,deptno from emp where sal > 1000 and deptno = 20 or deptno = 30;#错误的
select ename,sal,deptno from emp where sal > 1000 and (deptno = 20 or deptno = 30);#正确的。
# 注意：当运算符的优先级不确定的时候加小括号。
```

in 等同于 or：找出工作岗位是MANAGER和SALESMAN的员工？

```
select ename,job from emp where job = 'SALESMAN' or job = 'MANAGER';
select ename,job from emp where job in('SALESMAN', 'MANAGER');

select ename,sal,job from emp where sal in(800, 5000); # in后面的值不是区间，是具体的值。
```

```
+-------+---------+-----------+
| ename | sal     | job       |
+-------+---------+-----------+
| SMITH |  800.00 | CLERK     |
| KING  | 5000.00 | PRESIDENT |
+-------+---------+-----------+
```

not in: 不在这几个值当中。

```
select ename,job from emp where sal not in(800, 5000);
```

模糊查询like ?

找出名字当中含有O的？

```
#（在模糊查询当中，必须掌握两个特殊的符号，一个是%，一个是_ ）
# %代表任意多个字符，_代表任意1个字符。
select ename from emp where ename like '%O%';
```

```
+-------+
| ename |
+-------+
| JONES |
| SCOTT |
| FORD  |
+-------+
```

找出名字中第二个字母是A的？

```
select ename from emp where ename like '_A%';
```

```
+--------+
| ename  |
+--------+
| WARD   |
| MARTIN |
| JAMES  |
+--------+
```

找出名字中有下划线的？

```
mysql> select * from t_user;
```

```
+------+----------+
| id   | name     |
+------+----------+
|    1 | zhangsan |
|    2 | lisi     |
|    3 | WANG_WU  |
+------+----------+
```

```
select name from t_user where name like '%_%';
```

```
+----------+
| name     |
+----------+
| zhangsan |
| lisi     |
| WANG_WU  |
+----------+
```

```
select name from t_user where name like '%\_%';
```

```
+---------+
| name    |
+---------+
| WANG_WU |
+---------+
```

找出名字中最后一个字母是T的？

```
select ename from emp where ename like '%T';
```

```
+-------+
| ename |
```

```
                +-------+
                | SCOTT |
                +-------+
```

13 排序

```
13、排序（升序、降序）
按照工资升序，找出员工名和薪资？
    select
        ename, sal
    from
        emp
    order by
        sal;
+--------+---------+
| ename  | sal     |
+--------+---------+
| SMITH  |  800.00 |
| JAMES  |  950.00 |
| ADAMS  | 1100.00 |
| WARD   | 1250.00 |
| MARTIN | 1250.00 |
| MILLER | 1300.00 |
| TURNER | 1500.00 |
| ALLEN  | 1600.00 |
| CLARK  | 2450.00 |
| BLAKE  | 2850.00 |
| JONES  | 2975.00 |
| FORD   | 3000.00 |
| SCOTT  | 3000.00 |
| KING   | 5000.00 |
+--------+---------+

#注意：默认是升序。怎么指定升序或者降序呢？asc表示升序，desc表示降序。
    select ename , sal from emp order by sal; // 升序
    select ename , sal from emp order by sal asc; // 升序
    select ename , sal from emp order by sal desc; // 降序。

按照工资的降序排列，当工资相同的时候再按照名字的升序排列。
    select ename, sal from emp order by sal desc; # 按照工资降序排列
    select ename, sal from emp order by sal desc , ename asc;
    #注意：越靠前的字段越能起到主导作用。只有当前面的字段无法完成排序的时候，才会启用后面的字段。

找出工作岗位是SALESMAN的员工，并且要求按照薪资的降序排列。
    select
        ename, job, sal
    from
        emp
    where
        job = 'SALESMAN'
    order by
        sal desc;
+--------+----------+---------+
| ename  | job      | sal     |
+--------+----------+---------+
| ALLEN  | SALESMAN | 1600.00 |
| TURNER | SALESMAN | 1500.00 |
| WARD   | SALESMAN | 1250.00 |
| MARTIN | SALESMAN | 1250.00 |
+--------+----------+---------+

    select
        字段                    3
    from
        表名                    1
    where
        条件                    2
    order by
        ....                    4
    # order by是最后执行的。
```

14 分组函数（多行处理函数）

```
14、分组函数
count 计数
sum 求和
avg 平均值
max 最大值
min 最小值
# 记住：所有的分组函数都是对“某一组”数据进行操作的。
找出工资总和？
    select sum(sal) from emp;
找出最高工资？
    select max(sal) from emp;
找出最低工资？
    select min(sal) from emp;
```

```
找出平均工资？
    select avg(sal) from emp;
找出总人数？
    select count(*) from emp;
    select count(ename) from emp;

# 分组函数一共5个。
# 分组函数还有另一个名字：多行处理函数。
# 多行处理函数的特点：输入多行，最终输出的结果是1行。

# 分组函数自动忽略NULL。
    select count(comm) from emp;
        +-------------+
        | count(comm) |
        +-------------+
        |           4 |
        +-------------+

    select sum(comm) from emp;
        +-----------+
        | sum(comm) |
        +-----------+
        |   2200.00 |
        +-----------+

    select sum(comm) from emp where comm is not null; #不需要额外添加这个过滤条件。sum函数自动忽略NULL。

    找出工资高于平均工资的员工？
        select avg(sal) from emp; # 平均工资
            +-------------+
            | avg(sal)    |
            +-------------+
            | 2073.214286 |
            +-------------+

        select ename,sal from emp where sal > avg(sal); #ERROR 1111 (HY000): Invalid use of group function
        思考以上的错误信息：无效的使用了分组函数？
            原因：SQL语句当中有一个语法规则，分组函数不可直接使用在 where子句当中。why????
            怎么解释？
                因为 group by是在 where执行之后才会执行的。
                select      5 #要查的数据（属性）
                    ..
                from            1 #从那个表中查询语句
                    ..
                where           2 #条件查询
                    ..
                group by        3 #分组查询
                    ..
                having      4 #对分组之后的数据进行再次过滤。
                    ..
                order by        6 #排序
                    ..

count(*)和 count(具体的某个字段)，他们有什么区别？
    count(*):不是统计某个字段中数据的个数，而是统计总记录条数。（和某个字段无关）
    count(comm): 表示统计comm字段中不为NULL的数据总数量。

分组函数也能组合起来用：
    select count(*),sum(sal),avg(sal),max(sal),min(sal) from emp;
        +----------+----------+-------------+----------+----------+
        | count(*) | sum(sal) | avg(sal)    | max(sal) | min(sal) |
        +----------+----------+-------------+----------+----------+
        |       14 | 29025.00 | 2073.214286 |  5000.00 |   800.00 |
        +----------+----------+-------------+----------+----------+

找出工资高于平均工资的员工？
    第一步：找出平均工资
        select avg(sal) from emp;
            +-------------+
            | avg(sal)    |
            +-------------+
            | 2073.214286 |
            +-------------+
    第二步：找出高于平均工资的员工
        select ename,sal from emp where sal > 2073.214286;
            +-------+---------+
            | ename | sal     |
            +-------+---------+
            | JONES | 2975.00 |
            | BLAKE | 2850.00 |
            | CLARK | 2450.00 |
            | SCOTT | 3000.00 |
            | KING  | 5000.00 |
            | FORD  | 3000.00 |
            +-------+---------+

        select ename,sal from emp where sal > (select avg(sal) from emp);#子查询
```

15 单行处理函数

```
15、单行处理函数
    什么是单行处理函数？
        输入一行，输出一行。

    计算每个员工的年薪？
        select ename,(sal+comm)*12 as yearsal from emp;
        # 重点：所有数据库都是这样规定的，只要有NULL参与的运算结果一定是NULL。
        使用ifnull函数：
        select ename,(sal+ifnull(comm,0))*12 as yearsal from emp;

    ifnull() 空处理函数？
        # ifnull(可能为NULL的数据,被当做什么处理) ：  属于单行处理函数。
        select ename,ifnull(comm,0) as comm from emp;
        +--------+---------+
        | ename  | comm    |
        +--------+---------+
        | SMITH  |    0.00 |
        | ALLEN  |  300.00 |
        | WARD   |  500.00 |
        | JONES  |    0.00 |
        | MARTIN | 1400.00 |
        | BLAKE  |    0.00 |
        | CLARK  |    0.00 |
        | SCOTT  |    0.00 |
        | KING   |    0.00 |
        | TURNER |    0.00 |
        | ADAMS  |    0.00 |
        | JAMES  |    0.00 |
        | FORD   |    0.00 |
        | MILLER |    0.00 |
        +--------+---------+
```

16 group by 和 having

```
16、 group by 和 having
    group by: 按照某个字段或者某些字段进行分组。
    having: having是对分组之后的数据进行再次过滤。

    案例:找出每个工作岗位的最高薪资。
    select max(sal),job from emp group by job;
    +----------+-----------+
    | max(sal) | job       |
    +----------+-----------+
    |  3000.00 | ANALYST   |
    |  1300.00 | CLERK     |
    |  2975.00 | MANAGER   |
    |  5000.00 | PRESIDENT |
    |  1600.00 | SALESMAN  |
    +----------+-----------+

    # 注意：分组函数一般都会和 group by联合使用，这也是为什么它被称为分组函数的原因。
    # 并且任何一个分组函数（ count sum avg max min）都是在 group by语句执行结束之后才会执行的。
    # 当一条 sql语句没有 group by的话，整张表的数据会自成一组。

    select ename,max(sal),job from emp group by job;
    以上在mysql当中，查询结果是有的，但是结果没有意义，在Oracle数据库当中会报错。语法错误。
    Oracle的语法规则比MySQL语法规则严谨。
    # 记住一个规则：当一条语句中有 group by的话，select后面只能跟分组函数和参与分组的字段。

    每个工作岗位的平均薪资？
        select job,avg(sal) from emp group by job;
        +-----------+-------------+
        | job       | avg(sal)    |
        +-----------+-------------+
        | ANALYST   | 3000.000000 |
        | CLERK     | 1037.500000 |
        | MANAGER   | 2758.333333 |
        | PRESIDENT | 5000.000000 |
        | SALESMAN  | 1400.000000 |
        +-----------+-------------+

    多个字段能不能联合起来一块分组？
    案例：找出每个部门不同工作岗位的最高薪资。
        select
            deptno,job,max(sal)
        from
            emp
        group by
            deptno,job;
        +--------+-----------+----------+
        | deptno | job       | max(sal) |
        +--------+-----------+----------+
        |     10 | CLERK     |  1300.00 |
        |     10 | MANAGER   |  2450.00 |
        |     10 | PRESIDENT |  5000.00 |
```

```
|     20 | ANALYST   |  3000.00 |
|     20 | CLERK     |  1100.00 |
|     20 | MANAGER   |  2975.00 |
|     30 | CLERK     |   950.00 |
|     30 | MANAGER   |  2850.00 |
|     30 | SALESMAN  |  1600.00 |
+--------+-----------+----------+
```

找出每个部门的最高薪资，要求显示薪资大于2900的数据。

第一步：找出每个部门的最高薪资

```
select max(sal),deptno from emp group by deptno;
+----------+--------+
| max(sal) | deptno |
+----------+--------+
|  5000.00 |     10 |
|  3000.00 |     20 |
|  2850.00 |     30 |
+----------+--------+
```

第二步：找出薪资大于2900

```
select max(sal),deptno from emp group by deptno having max(sal) > 2900; # 这种方式效率低。
+----------+--------+
| max(sal) | deptno |
+----------+--------+
|  5000.00 |     10 |
|  3000.00 |     20 |
+----------+--------+

select max(sal),deptno from emp where sal > 2900 group by deptno;  # 效率较高，建议能够使用where过滤的尽量使用where。
+----------+--------+
| max(sal) | deptno |
+----------+--------+
|  5000.00 |     10 |
|  3000.00 |     20 |
+----------+--------+
```

找出每个部门的平均薪资，要求显示薪资大于2000的数据。

第一步：找出每个部门的平均薪资

```
select deptno,avg(sal) from emp group by deptno;
+--------+-------------+
| deptno | avg(sal)    |
+--------+-------------+
|     10 | 2916.666667 |
|     20 | 2175.000000 |
|     30 | 1566.666667 |
+--------+-------------+
```

第二步：要求显示薪资大于2000的数据

```
select deptno,avg(sal) from emp group by deptno having avg(sal) > 2000;
+--------+-------------+
| deptno | avg(sal)    |
+--------+-------------+
|     10 | 2916.666667 |
|     20 | 2175.000000 |
+--------+-------------+
```

where后面不能使用分组函数：

```
select deptno,avg(sal) from emp where avg(sal) > 2000 group by deptno;  # 错误了。
```

这种情况只能使用having过滤。

17 总结DQL语句怎么写

```
17、总结一个完整的DQL语句怎么写？
    select
            ..      5 #要查的数据（属性）
    from
            ..      1 #从那个表中查询语句
    where
            ..      2 #条件查询
    group by
            ..      3 #分组查询
    having
            ..      4 #对分组之后的数据进行再次过滤。
    order by
            ..      6 #排序
```

day02

1 去除重复函数

```
1、关于查询结果集的去重？
mysql> select distinct job from emp; # distinct关键字去除重复记录。
+-----------+
| job       |
+-----------+
| CLERK     |
| SALESMAN  |
```

```
| MANAGER   |
| ANALYST   |
| PRESIDENT |
+-----------+

mysql> select ename,distinct job from emp;
# 以上的sql语句是错误的。
# 记住：distinct只能出现在所有字段的最前面。

mysql> select distinct deptno,job from emp;
+--------+-----------+
| deptno | job       |
+--------+-----------+
|     20 | CLERK     |
|     30 | SALESMAN  |
|     20 | MANAGER   |
|     30 | MANAGER   |
|     10 | MANAGER   |
|     20 | ANALYST   |
|     10 | PRESIDENT |
|     30 | CLERK     |
|     10 | CLERK     |
+--------+-----------+

案例：统计岗位的数量？
select count(distinct job) from emp;
+----------------------+
| count(distinct job) |
+----------------------+
|                    5 |
+----------------------+
```

2 连接查询

2.1 什么是连接查询

```
2.1、什么是连接查询？
    在实际开发中，大部分的情况下都不是从单表中查询数据，一般都是多张表联合查询取出最终的结果。
    在实际开发中，一般一个业务都会对应多张表，比如：学生和班级，起码两张表。
        stuno       stuname         classno     classname
        ------------------------------------------------------------------------------
        1           zs              1           北京大兴区亦庄经济技术开发区第二中学高三1班
        2           ls              1           北京大兴区亦庄经济技术开发区第二中学高三1班
        ...
        学生和班级信息存储到一张表中，结果就像上面一样，数据会存在大量的重复，导致数据的冗余。
```

2.2 连接查询的分类

```
2.2、连接查询的分类？
    根据语法出现的年代来划分的话，包括：
        SQL92（一些老的DBA可能还在使用这种语法。DBA：DataBase Administrator，数据库管理员）
        SQL99（比较新的语法）

    根据表的连接方式来划分，包括：
        内连接：
            等值连接
            非等值连接
            自连接
        外连接：
            左外连接（左连接）
            右外连接（右连接）
        全连接（这个不讲，很少用！）
```

2.3 笛卡尔积现象

```
2.3、在表的连接查询方面有一种现象被称为：笛卡尔积现象。（笛卡尔乘积现象）
案例：找出每一个员工的部门名称，要求显示员工名和部门名。
EMP表
+--------+--------+
| ename  | deptno |
+--------+--------+
| SMITH  |     20 |
| ALLEN  |     30 |
| WARD   |     30 |
| JONES  |     20 |
| MARTIN |     30 |
| BLAKE  |     30 |
| CLARK  |     10 |
| SCOTT  |     20 |
| KING   |     10 |
| TURNER |     30 |
| ADAMS  |     20 |
| JAMES  |     30 |
| FORD   |     20 |
| MILLER |     10 |
+--------+--------+
```

```

DEPT表
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+

select ename,dname from emp,dept;
+--------+------------+
| ename  | dname      |
+--------+------------+
| SMITH  | ACCOUNTING |
| SMITH  | RESEARCH   |
| SMITH  | SALES      |
| SMITH  | OPERATIONS |
| ALLEN  | ACCOUNTING |
| ALLEN  | RESEARCH   |
| ALLEN  | SALES      |
| ALLEN  | OPERATIONS |
.............
56 rows in set (0.00 sec)

# 笛卡尔积现象：当两张表进行连接查询的时候，没有任何条件进行限制，最终的查询结果条数是两张表记录条数的乘积。

关于表的别名：
    select e.ename,d.dname from emp e,dept d;
    表的别名有什么好处？
        第一：执行效率高。
        第二：可读性好。
```

2.4 怎么避免笛卡尔积现象

```
2.4、怎么避免笛卡尔积现象？当然是加条件进行过滤。
思考：避免了笛卡尔积现象，会减少记录的匹配次数吗？
    不会，次数还是56次。只不过显示的是有效记录。

案例：找出每一个员工的部门名称，要求显示员工名和部门名。
    select
        e.ename,d.dname
    from
        emp e , dept d
    where
        e.deptno = d.deptno; #SQL92，以后不用。
    +--------+------------+
    | ename  | dname      |
    +--------+------------+
    | CLARK  | ACCOUNTING |
    | KING   | ACCOUNTING |
    | MILLER | ACCOUNTING |
    | SMITH  | RESEARCH   |
    | JONES  | RESEARCH   |
    | SCOTT  | RESEARCH   |
    | ADAMS  | RESEARCH   |
    | FORD   | RESEARCH   |
    | ALLEN  | SALES      |
    | WARD   | SALES      |
    | MARTIN | SALES      |
    | BLAKE  | SALES      |
    | TURNER | SALES      |
    | JAMES  | SALES      |
    +--------+------------+
```

2.5 内连接之等值连接

```
2.5、内连接之等值连接：# 最大特点是：条件是等量关系。
案例：查询每个员工的部门名称，要求显示员工名和部门名。
SQL92：（太老，不用了）
    select
        e.ename,d.dname
    from
        emp e, dept d
    where
        e.deptno = d.deptno;

SQL99：（常用的）
    select
        e.ename,d.dname
    from
        emp e
    join
        dept d
    on
        e.deptno = d.deptno;
```

```
# inner可以省略的，带着inner目的是可读性好一些。
select
    e.ename,d.dname
from
    emp e
inner join
    dept d
on
    e.deptno = d.deptno;

语法：
    ...
        A
    join
        B
    on
        连接条件
    where
        ...
# SQL99语法结构更清晰一些：表的连接条件和后来的where条件分离了。
+--------+------------+
| ename  | dname      |
+--------+------------+
| CLARK  | ACCOUNTING |
| KING   | ACCOUNTING |
| MILLER | ACCOUNTING |
| SMITH  | RESEARCH   |
| JONES  | RESEARCH   |
| SCOTT  | RESEARCH   |
| ADAMS  | RESEARCH   |
| FORD   | RESEARCH   |
| ALLEN  | SALES      |
| WARD   | SALES      |
| MARTIN | SALES      |
| BLAKE  | SALES      |
| TURNER | SALES      |
| JAMES  | SALES      |
+--------+------------+
```

2.6 内连接之非等值连接

```
2.6、内连接之非等值连接：# 最大的特点是：连接条件中的关系是非等量关系。
案例：找出每个员工的工资等级，要求显示员工名、工资、工资等级。
mysql> select ename,sal from emp; e
+--------+---------+
| ename  | sal     |
+--------+---------+
| SMITH  |  800.00 |
| ALLEN  | 1600.00 |
| WARD   | 1250.00 |
| JONES  | 2975.00 |
| MARTIN | 1250.00 |
| BLAKE  | 2850.00 |
| CLARK  | 2450.00 |
| SCOTT  | 3000.00 |
| KING   | 5000.00 |
| TURNER | 1500.00 |
| ADAMS  | 1100.00 |
| JAMES  |  950.00 |
| FORD   | 3000.00 |
| MILLER | 1300.00 |
+--------+---------+

mysql> select * from salgrade;
+-------+-------+-------+
| GRADE | LOSAL | HISAL |
+-------+-------+-------+
|     1 |   700 |  1200 |
|     2 |  1201 |  1400 |
|     3 |  1401 |  2000 |
|     4 |  2001 |  3000 |
|     5 |  3001 |  9999 |
+-------+-------+-------+

select
    e.ename,e.sal,s.grade
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal;

# inner可以省略
select
    e.ename,e.sal,s.grade
from
```

```
    emp e
inner join
    salgrade s
on
    e.sal between s.losal and s.hisal;
+--------+---------+-------+
| ename  | sal     | grade |
+--------+---------+-------+
| SMITH  |  800.00 |     1 |
| ALLEN  | 1600.00 |     3 |
| WARD   | 1250.00 |     2 |
| JONES  | 2975.00 |     4 |
| MARTIN | 1250.00 |     2 |
| BLAKE  | 2850.00 |     4 |
| CLARK  | 2450.00 |     4 |
| SCOTT  | 3000.00 |     4 |
| KING   | 5000.00 |     5 |
| TURNER | 1500.00 |     3 |
| ADAMS  | 1100.00 |     1 |
| JAMES  |  950.00 |     1 |
| FORD   | 3000.00 |     4 |
| MILLER | 1300.00 |     2 |
+--------+---------+-------+
```

2.7 内连接之自连接

```
2.7、自连接：# 最大的特点是：一张表看做两张表。自己连接自己。
案例：找出每个员工的上级领导，要求显示员工名和对应的领导名。
mysql> select empno,ename,mgr from emp;
emp a 员工表
+-------+--------+------+
| empno | ename  | mgr  |
+-------+--------+------+
|  7369 | SMITH  | 7902 |
|  7499 | ALLEN  | 7698 |
|  7521 | WARD   | 7698 |
|  7566 | JONES  | 7839 |
|  7654 | MARTIN | 7698 |
|  7698 | BLAKE  | 7839 |
|  7782 | CLARK  | 7839 |
|  7788 | SCOTT  | 7566 |
|  7839 | KING   | NULL |
|  7844 | TURNER | 7698 |
|  7876 | ADAMS  | 7788 |
|  7900 | JAMES  | 7698 |
|  7902 | FORD   | 7566 |
|  7934 | MILLER | 7782 |
+-------+--------+------+
emp b 领导表
+-------+--------+
| empno | ename  |
+-------+--------+
|  7566 | JONES  |
|  7698 | BLAKE  |
|  7782 | CLARK  |
|  7788 | SCOTT  |
|  7839 | KING   |
|  7902 | FORD   |
+-------+--------+
员工的领导编号 = 领导的员工编号 # a.mgr = b.empno

select
    a.ename as '员工名',b.ename as '领导名'
from
    emp a
inner join
    emp b
on
    a.mgr = b.empno;
+--------+--------+
| 员工名 | 领导名 |
+--------+--------+
| SMITH  | FORD   |
| ALLEN  | BLAKE  |
| WARD   | BLAKE  |
| JONES  | KING   |
| MARTIN | BLAKE  |
| BLAKE  | KING   |
| CLARK  | KING   |
| SCOTT  | JONES  |
| TURNER | BLAKE  |
| ADAMS  | SCOTT  |
| JAMES  | BLAKE  |
| FORD   | JONES  |
| MILLER | CLARK  |
+--------+--------+
```

2.8 外连接

```
2.8、外连接
什么是外连接，和内连接有什么区别？
    内连接：
        假设A和B表进行连接，使用内连接的话，凡是A表和B表能够匹配上的记录查询出来，这就是内连接。
        AB两张表没有主副之分，两张表是平等的。
    外连接：
        假设A和B表进行连接，使用外连接的话，AB两张表中有一张表是主表，一张表是副表，主要查询主表中
        的数据，捎带着查询副表，当副表中的数据没有和主表中的数据匹配上，副表自动模拟出NULL与之匹配。
    外连接的分类？
        左外连接（左连接）：表示左边的这张表是主表。
        右外连接（右连接）：表示右边的这张表是主表。

        左连接有右连接的写法，右连接也会有对应的左连接的写法。

案例：找出每个员工的上级领导？（所有员工必须全部查询出来。）
emp a 员工表
+-------+--------+------+
| empno | ename  | mgr  |
+-------+--------+------+
|  7369 | SMITH  | 7902 |
|  7499 | ALLEN  | 7698 |
|  7521 | WARD   | 7698 |
|  7566 | JONES  | 7839 |
|  7654 | MARTIN | 7698 |
|  7698 | BLAKE  | 7839 |
|  7782 | CLARK  | 7839 |
|  7788 | SCOTT  | 7566 |
|  7839 | KING   | NULL |
|  7844 | TURNER | 7698 |
|  7876 | ADAMS  | 7788 |
|  7900 | JAMES  | 7698 |
|  7902 | FORD   | 7566 |
|  7934 | MILLER | 7782 |
+-------+--------+------+
emp b 领导表
+-------+--------+
| empno | ename  |
+-------+--------+
|  7566 | JONES  |
|  7698 | BLAKE  |
|  7782 | CLARK  |
|  7788 | SCOTT  |
|  7839 | KING   |
|  7902 | FORD   |
+-------+--------+
# 内连接：
select
    a.ename '员工', b.ename '领导'
from
    emp a
join
    emp b
on
    a.mgr = b.empno;

# 外连接：（左外连接/左连接）
select
    a.ename '员工', b.ename '领导'
from
    emp a
left join
    emp b
on
    a.mgr = b.empno;
# outer是可以省略的。
select
    a.ename '员工', b.ename '领导'
from
    emp a
left outer join
    emp b
on
    a.mgr = b.empno;

#外连接：（右外连接/右连接）
select
    a.ename '员工', b.ename '领导'
from
    emp b
right join
    emp a
on
    a.mgr = b.empno;
# outer可以省略。
select
    a.ename '员工', b.ename '领导'
```

```
from
    emp b
right outer join
    emp a
on
    a.mgr = b.empno;
+--------+-------+
| 员工   |    领导|
+--------+-------+
| SMITH  | FORD  |
| ALLEN  | BLAKE |
| WARD   | BLAKE |
| JONES  | KING  |
| MARTIN | BLAKE |
| BLAKE  | KING  |
| CLARK  | KING  |
| SCOTT  | JONES |
| KING   | NULL  |
| TURNER | BLAKE |
| ADAMS  | SCOTT |
| JAMES  | BLAKE |
| FORD   | JONES |
| MILLER | CLARK |
+--------+-------+

# 外连接最重要的特点是：主表的数据无条件的全部查询出来。

案例：找出哪个部门没有员工？
EMP表
+-------+--------+-----------+------+------------+---------+---------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+--------+-----------+------+------------+---------+---------+--------+
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-17 |  800.00 |    NULL |     20 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-20 | 1600.00 |  300.00 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-22 | 1250.00 |  500.00 |     30 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-02 | 2975.00 |    NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-28 | 1250.00 | 1400.00 |     30 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850.00 |    NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-06-09 | 2450.00 |    NULL |     10 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1987-04-19 | 3000.00 |    NULL |     20 |
|  7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000.00 |    NULL |     10 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-09-08 | 1500.00 |    0.00 |     30 |
|  7876 | ADAMS  | CLERK     | 7788 | 1987-05-23 | 1100.00 |    NULL |     20 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-03 |  950.00 |    NULL |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-03 | 3000.00 |    NULL |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-23 | 1300.00 |    NULL |     10 |
+-------+--------+-----------+------+------------+---------+---------+--------+
DEPT
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
select
    d.*
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno
where
    e.empno is null;
+--------+------------+--------+
| DEPTNO | DNAME      | LOC    |
+--------+------------+--------+
|     40 | OPERATIONS | BOSTON |
+--------+------------+--------+
```

2.9 三张表怎么连接查询

```
2.9、三张表怎么连接查询？
案例：找出每一个员工的部门名称以及工资等级。
EMP e
+-------+--------+---------+--------+
| empno | ename  | sal     | deptno |
+-------+--------+---------+--------+
|  7369 | SMITH  |  800.00 |     20 |
|  7499 | ALLEN  | 1600.00 |     30 |
|  7521 | WARD   | 1250.00 |     30 |
|  7566 | JONES  | 2975.00 |     20 |
|  7654 | MARTIN | 1250.00 |     30 |
|  7698 | BLAKE  | 2850.00 |     30 |
|  7782 | CLARK  | 2450.00 |     10 |
|  7788 | SCOTT  | 3000.00 |     20 |
```

```
|  7839 | KING   | 5000.00 |     10 |
|  7844 | TURNER | 1500.00 |     30 |
|  7876 | ADAMS  | 1100.00 |     20 |
|  7900 | JAMES  |  950.00 |     30 |
|  7902 | FORD   | 3000.00 |     20 |
|  7934 | MILLER | 1300.00 |     10 |
+-------+--------+---------+--------+
DEPT d
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
SALGRADE s
+-------+-------+-------+
| GRADE | LOSAL | HISAL |
+-------+-------+-------+
|     1 |   700 |  1200 |
|     2 |  1201 |  1400 |
|     3 |  1401 |  2000 |
|     4 |  2001 |  3000 |
|     5 |  3001 |  9999 |
+-------+-------+-------+

注意，解释一下：
    ....
        A
    join
        B
    on
        ...
    join
        C
    on
        ...

    # 表示：A表和B表先进行表连接，连接之后A表继续和C表进行连接。

    select
        e.ename,d.dname,s.grade
    from
        emp e
    join
        dept d
    on
        e.deptno = d.deptno #等值连接
    join
        salgrade s
    on
        e.sal between s.losal and s.hisal; #非等值连接
    +--------+------------+-------+
    | ename  | dname      | grade |
    +--------+------------+-------+
    | SMITH  | RESEARCH   |     1 |
    | ALLEN  | SALES      |     3 |
    | WARD   | SALES      |     2 |
    | JONES  | RESEARCH   |     4 |
    | MARTIN | SALES      |     2 |
    | BLAKE  | SALES      |     4 |
    | CLARK  | ACCOUNTING |     4 |
    | SCOTT  | RESEARCH   |     4 |
    | KING   | ACCOUNTING |     5 |
    | TURNER | SALES      |     3 |
    | ADAMS  | RESEARCH   |     1 |
    | JAMES  | SALES      |     1 |
    | FORD   | RESEARCH   |     4 |
    | MILLER | ACCOUNTING |     2 |
    +--------+------------+-------+

案例：找出每一个员工的部门名称、工资等级、以及上级领导。
    select
        e.ename '员工',d.dname,s.grade,e1.ename '领导'
    from
        emp e
    join
        dept d
    on
        e.deptno = d.deptno
    join
        salgrade s
    on
        e.sal between s.losal and s.hisal
    left join
        emp e1
    on
```

```
        e.mgr = e1.empno;

+--------+------------+-------+-------+
| 员工   | dname      | grade | 领导  |
+--------+------------+-------+-------+
| SMITH  | RESEARCH   |     1 | FORD  |
| ALLEN  | SALES      |     3 | BLAKE |
| WARD   | SALES      |     2 | BLAKE |
| JONES  | RESEARCH   |     4 | KING  |
| MARTIN | SALES      |     2 | BLAKE |
| BLAKE  | SALES      |     4 | KING  |
| CLARK  | ACCOUNTING |     4 | KING  |
| SCOTT  | RESEARCH   |     4 | JONES |
| KING   | ACCOUNTING |     5 | NULL  |
| TURNER | SALES      |     3 | BLAKE |
| ADAMS  | RESEARCH   |     1 | SCOTT |
| JAMES  | SALES      |     1 | BLAKE |
| FORD   | RESEARCH   |     4 | JONES |
| MILLER | ACCOUNTING |     2 | CLARK |
+--------+------------+-------+-------+
```

3 子查询

3.1 什么是子查询？子查询都可以出现在哪里？

```
3.1、什么是子查询？子查询都可以出现在哪里？
    select 语句当中嵌套 select 语句，被嵌套的 select 语句是子查询。
    子查询可以出现在哪里？
        select
            ..(select).
        from
            ..(select).
        where
            ..(select).
```

3.2 where子句中使用子查询

```
3.2、where子句中使用子查询
案例：找出高于平均薪资的员工信息。
select * from emp where sal > avg(sal); #错误的写法，where后面不能直接使用分组函数。

第一步：找出平均薪资
    select avg(sal) from emp;
    +-------------+
    | avg(sal)    |
    +-------------+
    | 2073.214286 |
    +-------------+
第二步：where过滤
    select * from emp where sal > 2073.214286;
    +-------+-------+-----------+------+------------+---------+------+--------+
    | EMPNO | ENAME | JOB       | MGR  | HIREDATE   | SAL     | COMM | DEPTNO |
    +-------+-------+-----------+------+------------+---------+------+--------+
    |  7566 | JONES | MANAGER   | 7839 | 1981-04-02 | 2975.00 | NULL |     20 |
    |  7698 | BLAKE | MANAGER   | 7839 | 1981-05-01 | 2850.00 | NULL |     30 |
    |  7782 | CLARK | MANAGER   | 7839 | 1981-06-09 | 2450.00 | NULL |     10 |
    |  7788 | SCOTT | ANALYST   | 7566 | 1987-04-19 | 3000.00 | NULL |     20 |
    |  7839 | KING  | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL |     10 |
    |  7902 | FORD  | ANALYST   | 7566 | 1981-12-03 | 3000.00 | NULL |     20 |
    +-------+-------+-----------+------+------------+---------+------+--------+
第一步和第二步合并：
    select * from emp where sal > (select avg(sal) from emp);
```

3.3 from后面嵌套子查询

```
3.3、from后面嵌套子查询
案例：找出每个部门平均薪水的等级。
第一步：找出每个部门平均薪水（按照部门编号分组，求sal的平均值）
select deptno, avg(sal) as avgsal from emp group by deptno;
+--------+-------------+
| deptno | avgsal      |
+--------+-------------+
|     10 | 2916.666667 |
|     20 | 2175.000000 |
|     30 | 1566.666667 |
+--------+-------------+
第二步：将以上的查询结果当做临时表t，让t表和salgrade s表连接，条件是：t.avgsal between s.losal and s.hisal
select
    t.*, s.grade
from
    (select deptno, avg(sal) as avgsal from emp group by deptno) t
join
    salgrade s
on
    t.avgsal between s.losal and s.hisal;
+--------+-------------+-------+
```

```
| deptno | avgsal      | grade |
+--------+-------------+-------+
|     30 | 1566.666667 |     3 |
|     10 | 2916.666667 |     4 |
|     20 | 2175.000000 |     4 |
+--------+-------------+-------+
```

案例：找出每个部门平均的薪水等级。
第一步：找出每个员工的薪水等级。

```
select e.ename,e.sal,e.deptno,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal;
+--------+---------+--------+-------+
| ename  | sal     | deptno | grade |
+--------+---------+--------+-------+
| SMITH  |  800.00 |     20 |     1 |
| ALLEN  | 1600.00 |     30 |     3 |
| WARD   | 1250.00 |     30 |     2 |
| JONES  | 2975.00 |     20 |     4 |
| MARTIN | 1250.00 |     30 |     2 |
| BLAKE  | 2850.00 |     30 |     4 |
| CLARK  | 2450.00 |     10 |     4 |
| SCOTT  | 3000.00 |     20 |     4 |
| KING   | 5000.00 |     10 |     5 |
| TURNER | 1500.00 |     30 |     3 |
| ADAMS  | 1100.00 |     20 |     1 |
| JAMES  |  950.00 |     30 |     1 |
| FORD   | 3000.00 |     20 |     4 |
| MILLER | 1300.00 |     10 |     2 |
+--------+---------+--------+-------+
```

第二步：基于以上结果，继续按照deptno分组，求grade平均值。

```
select
    e.deptno, avg(s.grade)
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal
group by
    e.deptno;
+--------+--------------+
| deptno | avg(s.grade) |
+--------+--------------+
|     10 |       3.6667 |
|     20 |       2.8000 |
|     30 |       2.5000 |
+--------+--------------+
```

3.4 在select后面嵌套子查询

```
3.4、在select后面嵌套子查询。
案例：找出每个员工所在的部门名称，要求显示员工名和部门名。
select
    e.ename, d.dname
from
    emp e
join
    dept d
on
    e.deptno = d.deptno;

select
    e.ename,(select d.dname from dept d where e.deptno = d.deptno) as dname
from
    emp e;
+--------+------------+
| ename  | dname      |
+--------+------------+
| SMITH  | RESEARCH   |
| ALLEN  | SALES      |
| WARD   | SALES      |
| JONES  | RESEARCH   |
| MARTIN | SALES      |
| BLAKE  | SALES      |
| CLARK  | ACCOUNTING |
| SCOTT  | RESEARCH   |
| KING   | ACCOUNTING |
| TURNER | SALES      |
| ADAMS  | RESEARCH   |
| JAMES  | SALES      |
| FORD   | RESEARCH   |
| MILLER | ACCOUNTING |
+--------+------------+
```

4 union

```
4、union （可以将查询结果集相加）
案例：找出工作岗位是 SALESMAN 和 MANAGER 的员工？
第一种：select ename,job from emp where job = 'MANAGER' or job = 'SALESMAN';
第二种：select ename,job from emp where job in('MANAGER','SALESMAN');
+--------+----------+
| ename  | job      |
+--------+----------+
| ALLEN  | SALESMAN |
| WARD   | SALESMAN |
| JONES  | MANAGER  |
| MARTIN | SALESMAN |
| BLAKE  | MANAGER  |
| CLARK  | MANAGER  |
| TURNER | SALESMAN |
+--------+----------+
第三种：union
select ename,job from emp where job = 'MANAGER'
union
select ename,job from emp where job = 'SALESMAN';
+--------+----------+
| ename  | job      |
+--------+----------+
| JONES  | MANAGER  |
| BLAKE  | MANAGER  |
| CLARK  | MANAGER  |
| ALLEN  | SALESMAN |
| WARD   | SALESMAN |
| MARTIN | SALESMAN |
| TURNER | SALESMAN |
+--------+----------+

两张不相干的表中的数据拼接在一起显示？
select ename from emp
union
select dname from dept;
+------------+
| ename      |
+------------+
| SMITH      |
| ALLEN      |
| WARD       |
| JONES      |
| MARTIN     |
| BLAKE      |
| CLARK      |
| SCOTT      |
| KING       |
| TURNER     |
| ADAMS      |
| JAMES      |
| FORD       |
| MILLER     |
| ACCOUNTING |
| RESEARCH   |
| SALES      |
| OPERATIONS |
+------------+

mysql> select ename,sal from emp
    -> union
    -> select dname from dept;
ERROR 1222 (21000): The used SELECT statements have a different number of columns
```

5 limit

```
5、limit (重点中的重点，以后分页查询全靠它了。)
5.1、limit是mysql特有的，其他数据库中没有，不通用。（Oracle中有一个相同的机制，叫做rownum）
5.2、limit取结果集中的部分数据，这是它的作用。
5.3、语法机制：
    limit startIndex, length
        startIndex表示起始位置，从0开始，0表示第一条数据。
        length表示取几个

    案例：取出工资前5名的员工（思路：降序取前5个）
        select ename,sal from emp order by sal desc;
        取前5个：
            select ename,sal from emp order by sal desc limit 0, 5;
            select ename,sal from emp order by sal desc limit 5;

5.4、limit是sql语句最后执行的一个环节：
    select          5
        ...
    from            1
        ...
    where           2
```

```
        ...
    group by        3
        ...
    having          4
        ...
    order by        6
        ...
    limit           7
        ... ;
```

5.5、案例：找出工资排名在第4到第9名的员工？

```
    select ename,sal from emp order by sal desc limit 3, 6;
    +--------+---------+
    | ename  | sal     |
    +--------+---------+
    | JONES  | 2975.00 |
    | BLAKE  | 2850.00 |
    | CLARK  | 2450.00 |
    | ALLEN  | 1600.00 |
    | TURNER | 1500.00 |
    | MILLER | 1300.00 |
    +--------+---------+
```

5.6、通用的标准分页sql？

每页显示3条记录：

第1页：0, 3

第2页：3, 3

第3页：6, 3

第4页：9, 3

第5页：12, 3

每页显示pageSize条记录：

第pageNo页：(pageNo - 1) * pageSize, pageSize

pageSize是什么？是每页显示多少条记录

pageNo是什么？显示第几页

```
java代码{
    int pageNo = 2; # 页码是2
    int pageSize = 10; # 每页显示10条

    limit (pageNo - 1) * pageSize, pageSize #limit 10, 10
}
```

6 创建表

```
6、创建表：
建表语句的语法格式：
    create table 表名(
        字段名1 数据类型,
        字段名2 数据类型,
        字段名3 数据类型,
        ....
            字段名4 数据类型
    );

关于MySQL当中字段的数据类型？以下只说常见的
    int         整数型(java 中的 int)
    bigint      长整型(java 中的 long)
    float       浮点型(java 中的 float double)
    char        定长字符串(java 中的 String)
    varchar      可变长字符串(java 中的 StringBuffer/StringBuilder)
    date        日期类型 （对应Java中的 java.sql.Date类型）
    BLOB        二进制大对象（存储图片、视频等流媒体信息） Binary Large OBject （对应java中的Object）
    CLOB        字符大对象（存储较大文本，比如，可以存储4G的字符串。） Character Large OBject（对应java中的Object）
    ......

char 和 varchar 怎么选择？
    在实际的开发中，当某个字段中的数据长度不发生改变的时候，是定长的，例如：性别、生日等都是采用 char。
    当一个字段的数据长度不确定，例如：简介、姓名等都是采用 varchar。

BLOB 和 CLOB 类型的使用？
    电影表：t_movie
    id(int)   name(varchar)   playtime(date/char)   haibao(BLOB)   history(CLOB)
-------------------------------------------------------------------------------
    1        蜘蛛侠
    2
    3

表名在数据库当中一般建议以：t_ 或者 tbl_ 开始。

创建学生表：
    学生信息包括：
        学号：bigint
        姓名：varchar
        性别：char
        班级编号：int
```

```
        生日：char

    create table t_student(
        no bigint,
        name varchar(255),
        sex char(1),
        classno varchar(255),
        birth char(10)
    );
```

7 insert 语句插入数据

```
7、insert语句插入数据
语法格式：
    insert into 表名(字段名1,字段名2,字段名3,....) values(值1,值2,值3,....)
    要求：字段的数量和值的数量相同，并且数据类型要对应相同。

insert into t_student(no,name,sex,classno,birth) values(1,'zhangsan','1','gaosan1ban');
ERROR 1136 (21S01): Column count does not match value count at row 1

insert into t_student(no,name,sex,classno,birth) values(1,'zhangsan','1','gaosan1ban', '1950-10-12');
mysql> select * from t_student;
    +------+----------+------+------------+------------+
    | no   | name     | sex  | classno    | birth      |
    +------+----------+------+------------+------------+
    |    1 | zhangsan | 1    | gaosan1ban | 1950-10-12 |
    +------+----------+------+------------+------------+

insert into t_student(name,sex,classno,birth,no) values('lisi','1','gaosan1ban', '1950-10-12',2);
mysql> select * from t_student;
    +------+----------+------+------------+------------+
    | no   | name     | sex  | classno    | birth      |
    +------+----------+------+------------+------------+
    |    1 | zhangsan | 1    | gaosan1ban | 1950-10-12 |
    |    2 | lisi     | 1    | gaosan1ban | 1950-10-12 |
    +------+----------+------+------------+------------+

insert into t_student(name) values('wangwu'); # 除name字段之外，剩下的所有字段自动插入NULL。
mysql> select * from t_student;
    +------+----------+------+------------+------------+
    | no   | name     | sex  | classno    | birth      |
    +------+----------+------+------------+------------+
    |    1 | zhangsan | 1    | gaosan1ban | 1950-10-12 |
    |    2 | lisi     | 1    | gaosan1ban | 1950-10-12 |
    | NULL | wangwu   | NULL | NULL       | NULL       |
    +------+----------+------+------------+------------+

insert into t_student(no) values(3);
mysql> select * from t_student;
    +------+----------+------+------------+------------+
    | no   | name     | sex  | classno    | birth      |
    +------+----------+------+------------+------------+
    |    1 | zhangsan | 1    | gaosan1ban | 1950-10-12 |
    |    2 | lisi     | 1    | gaosan1ban | 1950-10-12 |
    | NULL | wangwu   | NULL | NULL       | NULL       |
    |    3 | NULL     | NULL | NULL       | NULL       |
    +------+----------+------+------------+------------+

drop table if exists t_student; # 当这个表存在的话删除。
create table t_student(
    no bigint,
    name varchar(255),
    sex char(1) default 1,
    classno varchar(255),
    birth char(10)
);

insert into t_student(name) values('zhangsan');
mysql> select * from t_student;
    +------+----------+------+---------+-------+
    | no   | name     | sex  | classno | birth |
    +------+----------+------+---------+-------+
    | NULL | zhangsan | 1    | NULL    | NULL  |
    +------+----------+------+---------+-------+

需要注意的地方：
    当一条insert语句执行成功之后，表格当中必然会多一行记录。
    即使多的这一行记录当中某些字段是NULL，后期也没有办法在执行
    insert 语句插入数据了，只能使用 update 进行更新。

# 字段可以省略不写，但是后面的value对数量和顺序都有要求。
insert into t_student values(1,'jack','0','gaosan2ban','1986-10-23');
mysql> select * from t_student;
    +------+----------+------+------------+------------+
    | no   | name     | sex  | classno    | birth      |
    +------+----------+------+------------+------------+
    | NULL | zhangsan | 1    | NULL       | NULL       |
```

```
    |    1 | jack     | 0    | gaosan2ban | 1986-10-23 |
    +------+----------+------+------------+------------+

insert into t_student values(1,'jack','0','gaosan2ban');
ERROR 1136 (21S01): Column count does not match value count at row 1
# 一次插入多行数据
insert into t_student
    (no,name,sex,classno,birth)
values
    (3,'rose','1','gaosi2ban','1952-12-14'),(4,'laotie','1','gaosi2ban','1955-12-14');
mysql> select * from t_student;
    +------+----------+------+------------+------------+
    | no   | name     | sex  | classno    | birth      |
    +------+----------+------+------------+------------+
    | NULL | zhangsan | 1    | NULL       | NULL       |
    |    1 | jack     | 0    | gaosan2ban | 1986-10-23 |
    |    3 | rose     | 1    | gaosi2ban  | 1952-12-14 |
    |    4 | laotie   | 1    | gaosi2ban  | 1955-12-14 |
    +------+----------+------+------------+------------+
```

8 表的复制

```
8、表的复制
    语法:
        create table 表名 as select 语句;#将查询结果当做表创建出来。

mysql> create table emp1 as select empno, ename from emp;#复制表
Query OK, 14 rows affected (0.02 sec)
Records: 14  Duplicates: 0  Warnings: 0

mysql> desc emp1;#查看复制表的表结构
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| empno | int(4)      | NO   |     | NULL    |       |
| ename | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from emp1;
+-------+--------+
| empno | ename  |
+-------+--------+
|  7369 | SMITH  |
|  7499 | ALLEN  |
|  7521 | WARD   |
|  7566 | JONES  |
|  7654 | MARTIN |
|  7698 | BLAKE  |
|  7782 | CLARK  |
|  7788 | SCOTT  |
|  7839 | KING   |
|  7844 | TURNER |
|  7876 | ADAMS  |
|  7900 | JAMES  |
|  7902 | FORD   |
|  7934 | MILLER |
+-------+--------+
```

9 将查询结果插入到一张表中

```
9、将查询结果插入到一张表中?
    mysql> create table dept1 as select * from dept;
    mysql> insert into dept1 select * from dept;
    mysql> select * from dept1;
    +--------+------------+----------+
    | DEPTNO | DNAME      | LOC      |
    +--------+------------+----------+
    |     10 | ACCOUNTING | NEW YORK |
    |     20 | RESEARCH   | DALLAS   |
    |     30 | SALES      | CHICAGO  |
    |     40 | OPERATIONS | BOSTON   |
    |     10 | ACCOUNTING | NEW YORK |
    |     20 | RESEARCH   | DALLAS   |
    |     30 | SALES      | CHICAGO  |
    |     40 | OPERATIONS | BOSTON   |
    +--------+------------+----------+
```

10 修改数据：update

```
10、修改数据：update
语法格式:
    update 表名 set 字段名1=值1,字段名2=值2... where 条件;
# 注意：没有条件整张表数据全部更新。

```

```
案例：将部门10的LOC修改为SHANGHAI，将部门名称修改为RENSHIBU
update dept1 set loc = 'SHANGHAI', dname = 'RENSHIBU' where deptno = 10;
mysql> select * from dept1;
    +--------+------------+----------+
    | DEPTNO | DNAME      | LOC      |
    +--------+------------+----------+
    |     10 | RENSHIBU   | SHANGHAI |
    |     20 | RESEARCH   | DALLAS   |
    |     30 | SALES      | CHICAGO  |
    |     40 | OPERATIONS | BOSTON   |
    |     10 | RENSHIBU   | SHANGHAI |
    |     20 | RESEARCH   | DALLAS   |
    |     30 | SALES      | CHICAGO  |
    |     40 | OPERATIONS | BOSTON   |
    +--------+------------+----------+

更新所有记录
    update dept1 set loc = 'x', dname = 'y';
    mysql> select * from dept1;
        +--------+-------+------+
        | DEPTNO | DNAME | LOC  |
        +--------+-------+------+
        |     10 | y     | x    |
        |     20 | y     | x    |
        |     30 | y     | x    |
        |     40 | y     | x    |
        |     10 | y     | x    |
        |     20 | y     | x    |
        |     30 | y     | x    |
        |     40 | y     | x    |
        +--------+-------+------+
```

11 删除数据：delete

```
11、删除数据？
语法格式：
    delete from 表名 where 条件;
#注意：没有条件全部删除。
删除10部门数据？
    delete from dept1 where deptno = 10;
删除所有记录？
    delete from dept1;

怎么删除大表中的数据？（重点）
    truncate table 表名; # 表被截断，不可回滚。永久丢失。
删除表？
    drop table 表名; # 这个通用。
    drop table if exists 表名; # oracle不支持这种写法。
```

12 CRUD操作

```
12、对于表结构的修改，这里不讲了，大家使用工具完成即可，因为在实际开发中表一旦
设计好之后，对表结构的修改是很少的，修改表结构就是对之前的设计进行了否定，即使
需要修改表结构，我们也可以直接使用工具操作。修改表结构的语句不会出现在Java代码当中。
出现在java代码当中的sql包括：insert delete update select（这些都是表中的数据操作。）

增删改查有一个术语：CRUD操作
Create（增） Retrieve（检索） Update（修改） Delete（删除）
```

13 约束(Constraint)

```
13、约束(Constraint)
什么是约束？常见的约束有哪些呢？
在创建表的时候，可以给表的字段添加相应的约束，添加约束的目的是为了保证表中数据的
合法性、有效性、完整性。
常见的约束有哪些呢？
    非空约束(not null)：约束的字段不能为NULL
    唯一约束(unique)：约束的字段不能重复
    主键约束(primary key)：约束的字段既不能为NULL，也不能重复（简称PK）
    外键约束(foreign key)：...（简称FK）
    检查约束(check)：注意Oracle数据库有check约束，但是mysql没有，目前mysql不支持该约束。
```

day03

1 约束

1.1 非空约束

```
1.1、非空约束（ not null）
drop table if exists t_user;
create table t_user(
    id int,
    username varchar(255) not null,
    password varchar(255)
);

insert into t_user(id,password) values(1,'123');
ERROR 1364 (HY000): Field 'username' does not have a default value

insert into t_user(id,username,password) values(1,'lisi','123');
```

1.2 唯一性约束

```
1.2、唯一性约束（unique）
* 唯一约束修饰的字段具有唯一性，不能重复。但可以为 NULL。
* 案例：给某一列添加 unique
    drop table if exists t_user;
    create table t_user(
        id int,
        username varchar(255) unique  # 列级约束
    );
    insert into t_user values(1,'zhangsan');
    insert into t_user values(2,'zhangsan');
    ERROR 1062 (23000): Duplicate entry 'zhangsan' for key 'username'

    insert into t_user(id) values(2);
    insert into t_user(id) values(3);
    insert into t_user(id) values(4);

# 案例：给两个列或者多个列添加unique
    drop table if exists t_user;
    create table t_user(
        id int,
        usercode varchar(255),
        username varchar(255),
        unique(usercode, username) # 多个字段联合起来添加1个约束unique 【表级约束】
    );

    insert into t_user values(1,'111','zs');
    insert into t_user values(2,'111','ls');
    insert into t_user values(3,'222','zs');
    select * from t_user;
    insert into t_user values(4,'111','zs');
    ERROR 1062 (23000): Duplicate entry '111-zs' for key 'usercode'

    drop table if exists t_user;
    create table t_user(
        id int,
        usercode varchar(255) unique,  # 列级约束
        username varchar(255) unique   # 列级约束
    );
    insert into t_user values(1,'111','zs');
    insert into t_user values(2,'111','ls');
    ERROR 1062 (23000): Duplicate entry '111' for key 'usercode'
# 注意： not null约束只有列级约束。没有表级约束。
```

1.3 主键约束

```
1.3、主键约束
* 怎么给一张表添加主键约束呢？
    drop table if exists t_user;
    create table t_user(
        id int primary key,  # 列级约束
        username varchar(255),
        email varchar(255)
    );
    insert into t_user(id,username,email) values(1,'zs','zs@123.com');
    insert into t_user(id,username,email) values(2,'ls','ls@123.com');
    insert into t_user(id,username,email) values(3,'ww','ww@123.com');
    select * from t_user;
        +----+----------+------------+
        | id | username | email      |
        +----+----------+------------+
        |  1 | zs       | zs@123.com |
        |  2 | ls       | ls@123.com |
        |  3 | ww       | ww@123.com |
        +----+----------+------------+

    insert into t_user(id,username,email) values(1,'jack','jack@123.com');
    ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

    insert into t_user(username,email) values('jack','jack@123.com');
    ERROR 1364 (HY000): Field 'id' does not have a default value
```

```
    # 根据以上的测试得出：id是主键，因为添加了主键约束，主键字段中的数据不能为NULL，也不能重复。
    # 主键的特点：不能为NULL，也不能重复。

* 主键相关的术语？
    主键约束 : primary key
    主键字段 : id字段添加 primary key 之后，id叫做主键字段
    主键值 : id字段中的每一个值都是主键值。

* 主键有什么作用？
    - 表的设计三范式中有要求，第一范式就要求任何一张表都应该有主键。
    - 主键的作用：主键值是这行记录在这张表当中的唯一标识。（就像一个人的身份证号码一样。）

* 主键的分类？
    根据主键字段的字段数量来划分：
        单一主键（推荐的，常用的。）
        复合主键(多个字段联合起来添加一个主键约束)（复合主键不建议使用，因为复合主键违背三范式。）
    根据主键性质来划分：
        自然主键：主键值最好就是一个和业务没有任何关系的自然数。（这种方式是推荐的）
        业务主键：主键值和系统的业务挂钩，例如：拿着银行卡的卡号做主键，拿着身份证号码作为主键。（不推荐用）
            最好不要拿着和业务挂钩的字段作为主键。因为以后的业务一旦发生改变的时候，主键值可能也需要
            随着发生变化，但有的时候没有办法变化，因为变化可能会导致主键值重复。

* 一张表的主键约束只能有1个。（必须记住）

* 使用表级约束方式定义主键：
    drop table if exists t_user;
    create table t_user(
        id int,
        username varchar(255),
        primary key(id)
    );
    insert into t_user(id,username) values(1,'zs');
    insert into t_user(id,username) values(2,'ls');
    insert into t_user(id,username) values(3,'ws');
    insert into t_user(id,username) values(4,'cs');
    select * from t_user;

    insert into t_user(id,username) values(4,'cx');
    ERROR 1062 (23000): Duplicate entry '4' for key 'PRIMARY'

    以下内容是演示以下复合主键，不需要掌握：
        drop table if exists t_user;
        create table t_user(
            id int,
            username varchar(255),
            password varchar(255),
            primary key(id,username)
        );
        insert .......

* mysql提供主键值自增：（非常重要。）
    drop table if exists t_user;
    create table t_user(
        id int primary key auto_increment, # id字段自动维护一个自增的数字，从1开始，以1递增。
        username varchar(255)
    );
    insert into t_user(username) values('a');
    insert into t_user(username) values('b');
    insert into t_user(username) values('c');
    insert into t_user(username) values('d');
    insert into t_user(username) values('e');
    insert into t_user(username) values('f');
    select * from t_user;
        +----+----------+
        | id | username |
        +----+----------+
        |  1 | a        |
        |  2 | b        |
        |  3 | c        |
        |  4 | d        |
        |  5 | e        |
        |  6 | f        |
        +----+----------+

    提示:Oracle当中也提供了一个自增机制，叫做：序列（sequence）对象。
```

1.4 外键约束

```
1.4、外键约束
* 关于外键约束的相关术语：
    外键约束: foreign key
    外键字段：添加有外键约束的字段
    外键值：外键字段中的每一个值。

* 业务背景：
    请设计数据库表，用来维护学生和班级的信息？
    第一种方案：一张表存储所有数据
    no(pk)      name          classno       classname
```

```
-----------------------------------------------------------------------------------------
    1           zs1             101             北京大兴区经济技术开发区亦庄二中高三1班
    2           zs2             101             北京大兴区经济技术开发区亦庄二中高三1班
    3           zs3             102             北京大兴区经济技术开发区亦庄二中高三2班
    4           zs4             102             北京大兴区经济技术开发区亦庄二中高三2班
    5           zs5             102             北京大兴区经济技术开发区亦庄二中高三2班
    缺点：冗余。【不推荐】

    第二种方案：两张表（班级表和学生表）
            t_class 班级表
            cno(pk)     cname
            ------------------------------------------------------
            101         北京大兴区经济技术开发区亦庄二中高三1班
            102         北京大兴区经济技术开发区亦庄二中高三2班

            t_student 学生表
            sno(pk)     sname               classno(该字段添加外键约束fk)
            ------------------------------------------------------------
            1           zs1                 101
            2           zs2                 101
            3           zs3                 102
            4           zs4                 102
            5           zs5                 102

* 将以上表的建表语句写出来：
    t_student 中的 classno 字段引用 t_class 表中的 cno 字段，
    此时 t_student 表叫做子表；t_class 表叫做父表。

    顺序要求：
        删除数据的时候，先删除子表，再删除父表。
        添加数据的时候，先添加父表，在添加子表。
        创建表的时候，先创建父表，再创建子表。
        删除表的时候，先删除子表，在删除父表。
```

```sql
drop table if exists t_student;
drop table if exists t_class;

create table t_class(
    cno int,
    cname varchar(255),
    primary key(cno)
);

create table t_student(
    sno int,
    sname varchar(255),
    classno int,
    primary key(sno),
    foreign key(classno) references t_class(cno)
);

insert into t_class values(101,'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx');
insert into t_class values(102,'yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy');

insert into t_student values(1,'zs1',101);
insert into t_student values(2,'zs2',101);
insert into t_student values(3,'zs3',102);
insert into t_student values(4,'zs4',102);
insert into t_student values(5,'zs5',102);
insert into t_student values(6,'zs6',102);
```

```
    select * from t_class;
    select * from t_student;

    insert into t_student values(7,'lisi',103);
    ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`bjpowernode`.INT `t_student_ibfk_1` FOREIGN KEY (`classno`) REFERENCES `t_class` (`cno`))

* 外键值可以为NULL？
    外键可以为NULL。

* 外键字段引用其他表的某个字段的时候，被引用的字段必须是主键吗？
    注意：被引用的字段不一定是主键，但至少具有unique约束。
```

2 存储引擎

```
2.1、完整的建表语句
    CREATE TABLE `t_x` (
        `id` int(11) DEFAULT NULL
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

    注意：在MySQL当中，凡是标识符是可以使用飘号括起来的。最好别用，不通用。

    建表的时候可以指定存储引擎，也可以指定字符集。

    mysql默认使用的存储引擎是InnoDB方式。
    默认采用的字符集是UTF8
```

```
2.2、什么是存储引擎呢？
存储引擎这个名字只有在mysql中存在。（Oracle中有对应的机制，但是不叫做存储引擎。Oracle中没有特殊的名字，
就是“表的存储方式”）

mysql支持很多存储引擎，每一个存储引擎都对应了一种不同的存储方式。
每一个存储引擎都有自己的优缺点，需要在合适的时机选择合适的存储引擎。
```

```
2.3、查看当前mysql支持的存储引擎？
show engines \G
mysql 5.5.36版本支持的存储引擎有9个：
*************************** 1. row ***************************
        Engine: FEDERATED
        Support: NO
        Comment: Federated MySQL storage engine
        Transactions: NULL
        XA: NULL
        Savepoints: NULL
*************************** 2. row ***************************
        Engine: MRG_MYISAM
        Support: YES
        Comment: Collection of identical MyISAM tables
        Transactions: NO
        XA: NO
        Savepoints: NO
*************************** 3. row ***************************
        Engine: MyISAM
        Support: YES
        Comment: MyISAM storage engine
        Transactions: NO
        XA: NO
        Savepoints: NO
*************************** 4. row ***************************
        Engine: BLACKHOLE
        Support: YES
        Comment: /dev/null storage engine (anything you write to it disappears)
        Transactions: NO
        XA: NO
        Savepoints: NO
*************************** 5. row ***************************
        Engine: CSV
        Support: YES
        Comment: CSV storage engine
        Transactions: NO
        XA: NO
        Savepoints: NO
*************************** 6. row ***************************
        Engine: MEMORY
        Support: YES
        Comment: Hash based, stored in memory, useful for temporary tables
        Transactions: NO
        XA: NO
        Savepoints: NO
*************************** 7. row ***************************
        Engine: ARCHIVE
        Support: YES
        Comment: Archive storage engine
        Transactions: NO
        XA: NO
        Savepoints: NO
*************************** 8. row ***************************
        Engine: InnoDB
        Support: DEFAULT
        Comment: Supports transactions, row-level locking, and foreign keys
        Transactions: YES
        XA: YES
        Savepoints: YES
*************************** 9. row ***************************
        Engine: PERFORMANCE_SCHEMA
        Support: YES
        Comment: Performance Schema
        Transactions: NO
        XA: NO
        Savepoints: NO
```

```
2.4、常见的存储引擎？
    Engine: MyISAM
    Support: YES
    Comment: MyISAM storage engine
    Transactions: NO    #是否支持事务
    XA: NO              #
    Savepoints: NO      #

MyISAM这种存储引擎不支持事务。
MyISAM是mysql最常用的存储引擎，但是这种引擎不是默认的。
MyISAM采用三个文件组织一张表：
    xxx.frm（存储格式的文件）
```

```
    xxx.MYD（存储表中数据的文件）
    xxx.MYI（存储表中索引的文件）
优点：可被压缩，节省存储空间。并且可以转换为只读表，提高检索效率。
缺点：不支持事务。
--------------------------------------------------------------------------------

    Engine: InnoDB
    Support: DEFAULT     #默认的
    Comment: Supports transactions, row-level locking, and foreign keys
    Transactions: YES
    XA: YES
    Savepoints: YES

优点：支持事务、行级锁、外键等。这种存储引擎数据的安全得到保障。

表的结构存储在xxx.frm文件中
数据存储在tablespace这样的表空间中（逻辑概念），无法被压缩，无法转换成只读。
这种InnoDB存储引擎在MySQL数据库崩溃之后提供自动恢复机制。
InnoDB支持级联删除和级联更新。
--------------------------------------------------------------------------------------

    Engine: MEMORY
    Support: YES
    Comment: Hash based, stored in memory, useful for temporary tables
    Transactions: NO
    XA: NO
    Savepoints: NO

缺点：不支持事务。数据容易丢失。因为所有数据和索引都是存储在内存当中的。
优点：查询速度最快。
以前叫做HEPA引擎。
```

3 事务

```
3、事务（Transaction）
3.1、什么是事务？
一个事务是一个完整的业务逻辑单元，不可再分。
比如：银行账户转账，从A账户向B账户转账10000.需要执行两条update语句：
    update t_act set balance = balance - 10000 where actno = 'act-001';
    update t_act set balance = balance + 10000 where actno = 'act-002';
以上两条DML语句必须同时成功，或者同时失败，不允许出现一条成功，一条失败。
要想保证以上的两条DML语句同时成功或者同时失败，那么就需要使用数据库的“事务机制”。

3.2、和事务相关的语句只有：DML语句。（ insert delete update）
    为什么？因为它们这三个语句都是和数据库表当中的“数据”相关的。
    事务的存在是为了保证数据的完整性，安全性。

3.3、假设所有的业务都能使用1条DML语句搞定，还需要事务机制吗？
    不需要事务。
    但实际情况不是这样的，通常一个“事儿（事务【业务】）”需要多条DML语句共同联合完成。

3.4、事务的特性？
    事务包括四大特性：ACID
    A：原子性：事务是最小的工作单元，不可再分。
    C：一致性：事务必须保证多条DML语句同时成功或者同时失败。
    I：隔离性：事务A与事务B之间具有隔离。
    D：持久性：持久性说的是最终数据必须持久化到硬盘文件中，事务才算成功的结束。

3.5、关于事务之间的隔离性
事务隔离性存在隔离级别，理论上隔离级别包括4个：
    第一级别：读未提交（ read uncommitted）
        对方事务还没有提交，我们当前事务可以读取到对方未提交的数据。
        读未提交存在脏读（ Dirty Read）现象：表示读到了脏的数据。
    第二级别：读已提交（ read committed）
        对方事务提交之后的数据我方可以读取到。
        这种隔离级别解决了：脏读现象没有了。
        读已提交存在的问题是：不可重复读。
    第三级别：可重复读（ repeatable read）
        这种隔离级别解决了：不可重复读问题。
        这种隔离级别存在的问题是：读取到的数据是幻象。（幻读）
    第四级别：序列化读/串行化读（ serializable）
        解决了所有问题。
        效率低。需要事务排队。

    oracle数据库默认的隔离级别是：读已提交。
    mysql数据库默认的隔离级别是：可重复读。

    -脏读现象（Dirty Read）
        一个事务开始读取了某行数据，但是另外一个事务已经更新了此数据但没有能够及时提交，这就出现了脏读现象。
    -不可重复读（Non-repeatable Read）
        在同一个事务中，同一个读操作对同一个数据的前后两次读取产生了不同的结果，这就是不可重复读。
    -幻读（Phantom Read）
        幻读是指在同一个事务中以前没有的行，由于其他事务的提交而出现的新行。

3.6、演示事务
* mysql事务默认情况下是自动提交的。
    （什么是自动提交？只要执行任意一条DML语句则提交一次。）怎么关闭自动提交？
```

```
     start transaction;

* 准备表：
    drop table if exists t_user;
    create table t_user(
        id int primary key auto_increment,
        username varchar(255)
    );

* 演示：mysql中的事务是支持自动提交的，只要执行一条DML，则提交一次。
    mysql> insert into t_user(username) values('zs');
    Query OK, 1 row affected (0.03 sec)

    mysql> select * from t_user;
    +----+----------+
    | id | username |
    +----+----------+
    |  1 | zs       |
    +----+----------+
    1 row in set (0.00 sec)

    mysql> rollback;
    Query OK, 0 rows affected (0.00 sec)

    mysql> select * from t_user;
    +----+----------+
    | id | username |
    +----+----------+
    |  1 | zs       |
    +----+----------+
    1 row in set (0.00 sec)

* 演示：使用start transaction;#关闭自动提交机制。
    mysql> start transaction;
    Query OK, 0 rows affected (0.00 sec)

    mysql> insert into t_user(username) values('lisi');
    Query OK, 1 row affected (0.00 sec)

    mysql> select * from t_user;
    +----+----------+
    | id | username |
    +----+----------+
    |  1 | zs       |
    |  2 | lisi     |
    +----+----------+
    2 rows in set (0.00 sec)

    mysql> insert into t_user(username) values('wangwu');
    Query OK, 1 row affected (0.00 sec)

    mysql> select * from t_user;
    +----+----------+
    | id | username |
    +----+----------+
    |  1 | zs       |
    |  2 | lisi     |
    |  3 | wangwu   |
    +----+----------+
    3 rows in set (0.00 sec)


    mysql> rollback;
    Query OK, 0 rows affected (0.02 sec)

    mysql> select * from t_user;
    +----+----------+
    | id | username |
    +----+----------+
    |  1 | zs       |
    +----+----------+
    1 row in set (0.00 sec)
    ----------------------------------------------------------------------
    mysql> start transaction;#关闭Mysql自动提交机制
    Query OK, 0 rows affected (0.00 sec)

    mysql> insert into t_user(username) values('wangwu');
    Query OK, 1 row affected (0.00 sec)

    mysql> insert into t_user(username) values('rose');
    Query OK, 1 row affected (0.00 sec)

    mysql> insert into t_user(username) values('jack');
    Query OK, 1 row affected (0.00 sec)

    mysql> select * from t_user;
    +----+----------+
    | id | username |
    +----+----------+
```

```
|  1 | zs       |
|  4 | wangwu   |
|  5 | rose     |
|  6 | jack     |
+----+----------+
4 rows in set (0.00 sec)

mysql> commit;#提交
Query OK, 0 rows affected (0.04 sec)

mysql> select * from t_user;
+----+----------+
| id | username |
+----+----------+
|  1 | zs       |
|  4 | wangwu   |
|  5 | rose     |
|  6 | jack     |
+----+----------+
4 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t_user;
+----+----------+
| id | username |
+----+----------+
|  1 | zs       |
|  4 | wangwu   |
|  5 | rose     |
|  6 | jack     |
+----+----------+
4 rows in set (0.00 sec)

* 演示两个事务，假如隔离级别
    演示第1级别：读未提交
        set global transaction isolation level read uncommitted;#将全局事务隔离级别设置为“读取未提交”
    演示第2级别：读已提交
        set global transaction isolation level read committed;#将全局事务隔离级别设置为“读已提交”
    演示第3级别：可重复读
        set global transaction isolation level repeatable read;#将全局事务隔离级别设置为“可重复读”
    演示第4级别：
        set global transaction isolation level serializable;#将全局事务隔离级别设置为“序列化读”
* mysql远程登录：mysql -h192.168.151.18 -uroot -p444
```

4 索引

```
4、索引
4.1、什么是索引？有什么用？
    索引就相当于一本书的目录，通过目录可以快速的找到对应的资源。
    在数据库方面，查询一张表的时候有两种检索方式：
        第一种方式：全表扫描
        第二种方式：根据索引检索（效率很高）

    索引为什么可以提高检索效率呢？
        其实最根本的原理是缩小了扫描的范围。

    索引虽然可以提高检索效率，但是不能随意的添加索引，因为索引也是数据库当中
    的对象，也需要数据库不断的维护。是有维护成本的。比如，表中的数据经常被修改
    这样就不适合添加索引，因为数据一旦修改，索引需要重新排序，进行维护。

    添加索引是给某一个字段，或者说某些字段添加索引。

    select ename,sal from emp where ename = 'SMITH';
    当ename字段上没有添加索引的时候，以上sql语句会进行全表扫描，扫描ename字段中所有的值。
    当ename字段上添加索引的时候，以上sql语句会根据索引扫描，快速定位。

4.2、怎么创建索引对象？怎么删除索引对象？
    创建索引对象：
        create index 索引名称 on 表名(字段名);
    删除索引对象：
        drop index 索引名称 on 表名;

4.3、什么时候考虑给字段添加索引？（满足什么条件）
    * 数据量庞大。（根据客户的需求，根据线上的环境）
    * 该字段很少的DML操作。（因为字段进行修改操作，索引也需要维护）
    * 该字段经常出现在where子句中。（经常根据哪个字段查询）

4.4、注意：主键和具有unique约束的字段自动会添加索引。
    根据主键查询效率较高。尽量根据主键检索。

4.5、查看sql语句的执行计划：
mysql> explain select ename,sal from emp where sal = 5000;
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
```

```
|  1 | SIMPLE      | emp   | ALL  | NULL          | NULL | NULL    | NULL |   14 | Using where |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
```

给薪资sal字段添加索引：

```
    create index emp_sal_index on emp(sal);

mysql> explain select ename,sal from emp where sal = 5000;
+----+-------------+-------+------+---------------+---------------+---------+-------+------+-------------+
| id | select_type | table | type | possible_keys | key           | key_len | ref   | rows | Extra       |
+----+-------------+-------+------+---------------+---------------+---------+-------+------+-------------+
|  1 | SIMPLE      | emp   | ref  | emp_sal_index | emp_sal_index | 9       | const |    1 | Using where |
+----+-------------+-------+------+---------------+---------------+---------+-------+------+-------------+
```

4.6、索引底层采用的数据结构是：B + Tree

4.7、索引的实现原理？

通过B Tree缩小扫描范围，底层索引进行了排序，分区，索引会携带数据在表中的“物理地址”，
最终通过索引检索到数据之后，获取到关联的物理地址，通过物理地址定位表中的数据，效率
是最高的。

```
    select ename from emp where ename = 'SMITH';
    通过索引转换为：
    select ename from emp where 物理地址 = 0x3;
```

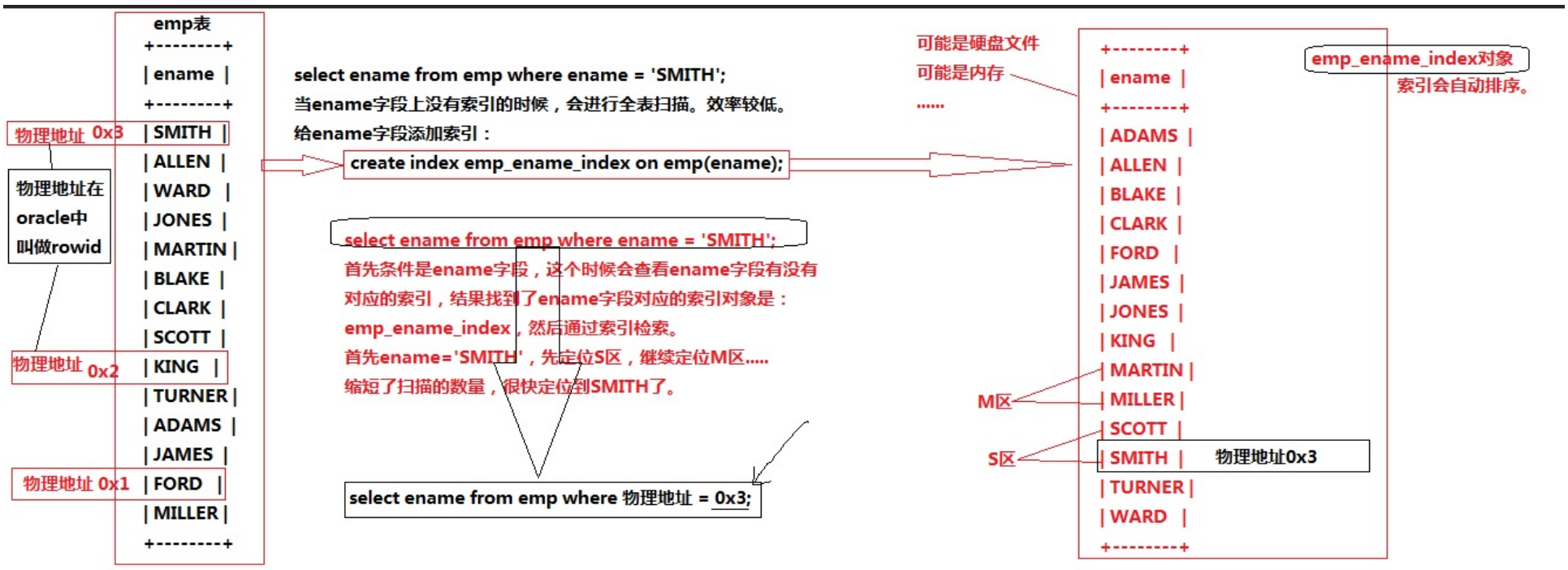
4.8、索引的分类？

单一索引：给单个字段添加索引
复合索引：给多个字段联合起来添加1个索引
主键索引：主键上会自动添加索引
唯一索引：有unique约束的字段上会自动添加索引
....

4.9、索引什么时候失效？

```
    select ename from emp where ename like '%A%';
```

模糊查询的时候，第一个通配符使用的是%，这个时候索引是失效的。



5 视图

```
5、视图(view)
5.1、什么是视图？
    站在不同的角度去看到数据。（同一张表的数据，通过不同的角度去看待）。

5.2、怎么创建视图？怎么删除视图？
    create view myview as select empno,ename from emp;
    drop view myview;

    注意：只有DQL语句才能以视图对象的方式创建出来。

5.3、对视图进行增删改查，会影响到原表数据。（通过视图影响原表数据的，不是直接操作的原表）
可以对视图进行CRUD操作。

5.4、面向视图操作？
    mysql> select * from myview;
        +-------+--------+
        | empno | ename  |
        +-------+--------+
        |  7369 | SMITH  |
        |  7499 | ALLEN  |
        |  7521 | WARD   |
        |  7566 | JONES  |
        |  7654 | MARTIN |
        |  7698 | BLAKE  |
        |  7782 | CLARK  |
        |  7788 | SCOTT  |
        |  7839 | KING   |
        |  7844 | TURNER |
```

```
        |  7876 | ADAMS  |
        |  7900 | JAMES  |
        |  7902 | FORD   |
        |  7934 | MILLER |
        +-------+--------+
```

```
    create table emp_bak as select * from emp;
    create view myview1 as select empno,ename,sal from emp_bak;
    update myview1 set ename='hehe',sal=1 where empno = 7369; # 通过视图修改原表数据。
    delete from myview1 where empno = 7369; # 通过视图删除原表数据。

5.5、视图的作用？
    视图可以隐藏表的实现细节。保密级别较高的系统，数据库只对外提供相关的视图，java程序员
    只对视图对象进行CRUD。
```

6 DBA命令

```
6.1、将数据库当中的数据导出
    在windows的dos命令窗口中执行：#（导出整个库）
        mysqldump bjpowernode>D:\bjpowernode.sql -uroot -p333

    在windows的dos命令窗口中执行：#（导出指定数据库当中的指定表）
        mysqldump bjpowernode emp>D:\bjpowernode.sql -uroot -p123

6.2、导入数据
    create database bjpowernode;
    use bjpowernode;
    source D:\bjpowernode.sql
```

7 数据库三大范式

```
7、数据库设计三范式（重点内容，面试经常问）
7.1、什么是设计范式？
    设计表的依据。按照这个三范式设计的表不会出现数据冗余。
7.2、三范式都是哪些？
    第一范式：任何一张表都应该有主键，并且每一个字段原子性不可再分。
    第二范式：建立在第一范式的基础之上，所有非主键字段完全依赖主键，不能产生部分依赖。
        多对多？三张表，关系表两个外键。
            t_student学生表
            sno(pk)     sname
            --------------------
            1           张三
            2           李四
            3           王五

            t_teacher 讲师表
            tno(pk)     tname
            ----------------------
            1           王老师
            2           张老师
            3           李老师

            t_student_teacher_relation 学生讲师关系表
            id(pk)      sno(fk)     tno(fk)
            ---------------------------------
            1           1           3
            2           1           1
            3           2           2
            4           2           3
            5           3           1
            6           3           3

    第三范式：建立在第二范式的基础之上，所有非主键字段直接依赖主键，不能产生传递依赖。
        一对多？两张表，多的表加外键。
            班级t_class
            cno(pk)         cname
            --------------------------
            1               班级1
            2               班级2

            学生t_student
            sno(pk)         sname           classno(fk)
            ---------------------------------------------
            101             张1              1
            102             张2              1
            103             张3              2
            104             张4              2
            105             张5              2

    提醒：在实际的开发中，以满足客户的需求为主，有的时候会拿冗余换执行速度。

7.3、一对一怎么设计？
        一对一设计有两种方案：主键共享
            t_user_login  用户登录表
            id(pk)      username            password
            ------------------------------------------
```

```
            1           zs                  123
            2           ls                  456

            t_user_detail 用户详细信息表
            id(pk+fk)   realname            tel         ....
            ------------------------------------------------
            1           张三              1111111111
            2           李四              1111415621

        一对一设计有两种方案：外键唯一。
            t_user_login  用户登录表
            id(pk)      username            password
            -----------------------------------------
            1           ls                  123
            2           zs                  456

            t_user_detail 用户详细信息表
            id(pk)     realname         tel             userid(fk+unique)....
            ------------------------------------------------------------
            1           张三          1111111111          2
            2           李四          1111415621          1
```

六 *MySql*作业

0 表中的数据

```
0 表中的数据
mysql> select * from emp;
+-------+--------+-----------+------+------------+---------+---------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO |
+-------+--------+-----------+------+------------+---------+---------+--------+
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-17 |  800.00 |    NULL |     20 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-20 | 1600.00 |  300.00 |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-22 | 1250.00 |  500.00 |     30 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-02 | 2975.00 |    NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-28 | 1250.00 | 1400.00 |     30 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850.00 |    NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-06-09 | 2450.00 |    NULL |     10 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1987-04-19 | 3000.00 |    NULL |     20 |
|  7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000.00 |    NULL |     10 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-09-08 | 1500.00 |    0.00 |     30 |
|  7876 | ADAMS  | CLERK     | 7788 | 1987-05-23 | 1100.00 |    NULL |     20 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-03 |  950.00 |    NULL |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-03 | 3000.00 |    NULL |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-23 | 1300.00 |    NULL |     10 |
+-------+--------+-----------+------+------------+---------+---------+--------+

mysql> select * from dept;
+--------+------------+----------+
| DEPTNO | DNAME      | LOC      |
+--------+------------+----------+
|     10 | ACCOUNTING | NEW YORK |
|     20 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+

mysql> select * from salgrade;
+-------+-------+-------+
| GRADE | LOSAL | HISAL |
+-------+-------+-------+
|     1 |   700 |  1200 |
|     2 |  1201 |  1400 |
|     3 |  1401 |  2000 |
|     4 |  2001 |  3000 |
|     5 |  3001 |  9999 |
+-------+-------+-------+
```

1 取得每个部门最高薪水的人员名称

```
1、取得每个部门最高薪水的人员名称
第一步：取得每个部门最高薪水(按照部门编号分组，找出每一组最大值)
mysql> select deptno,max(sal) as maxsal from emp group by deptno;
+--------+---------+
| deptno | maxsal  |
+--------+---------+
|     10 | 5000.00 |
|     20 | 3000.00 |
|     30 | 2850.00 |
+--------+---------+

第二步：将以上的查询结果当做一张临时表t，
t和emp表连接，条件：t.deptno = e.deptno and t.maxsal = e.sal
```

```
select
    e.ename, t.*
from
    emp e
join
    (select deptno,max(sal) as maxsal from emp group by deptno) t
on
    t.deptno = e.deptno and t.maxsal = e.sal;
+-------+--------+---------+
| ename | deptno | maxsal  |
+-------+--------+---------+
| BLAKE |     30 | 2850.00 |
| SCOTT |     20 | 3000.00 |
| KING  |     10 | 5000.00 |
| FORD  |     20 | 3000.00 |
+-------+--------+---------+
```

2 哪些人的薪水在部门的平均薪水之上

```
2、哪些人的薪水在部门的平均薪水之上
第一步：找出每个部门的平均薪水
select deptno,avg(sal) as avgsal from emp group by deptno;
+--------+-------------+
| deptno | avgsal      |
+--------+-------------+
|     10 | 2916.666667 |
|     20 | 2175.000000 |
|     30 | 1566.666667 |
+--------+-------------+

第二步：将以上查询结果当做t表，t和emp表连接
条件：部门编号相同，并且emp的sal大于t表的avgsal
select
    t.*, e.ename, e.sal
from
    emp e
join
    (select deptno,avg(sal) as avgsal from emp group by deptno) t
on
    e.deptno = t.deptno and e.sal > t.avgsal;
+--------+-------------+-------+---------+
| deptno | avgsal      | ename | sal     |
+--------+-------------+-------+---------+
|     30 | 1566.666667 | ALLEN | 1600.00 |
|     20 | 2175.000000 | JONES | 2975.00 |
|     30 | 1566.666667 | BLAKE | 2850.00 |
|     20 | 2175.000000 | SCOTT | 3000.00 |
|     10 | 2916.666667 | KING  | 5000.00 |
|     20 | 2175.000000 | FORD  | 3000.00 |
+--------+-------------+-------+---------+
```

3 取得部门中（所有人的）平均的薪水等级

```
3、取得部门中（所有人的）平均的薪水等级
平均的薪水等级：先计算每一个薪水的等级，然后找出薪水等级的平均值。
平均薪水的等级：先计算平均薪水，然后找出每个平均薪水的等级值。

第一步：找出每个人的薪水等级
emp e和salgrade s表连接。
连接条件：e.sal between s.losal and s.hisal
select
    e.ename,e.sal,e.deptno,s.grade
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal;
    +--------+---------+--------+-------+
    | ename  | sal     | deptno | grade |
    +--------+---------+--------+-------+
    | CLARK  | 2450.00 |     10 |     4 |
    | KING   | 5000.00 |     10 |     5 |
    | MILLER | 1300.00 |     10 |     2 |
    | SMITH  |  800.00 |     20 |     1 |
    | ADAMS  | 1100.00 |     20 |     1 |
    | SCOTT  | 3000.00 |     20 |     4 |
    | FORD   | 3000.00 |     20 |     4 |
    | JONES  | 2975.00 |     20 |     4 |
    | MARTIN | 1250.00 |     30 |     2 |
    | TURNER | 1500.00 |     30 |     3 |
    | BLAKE  | 2850.00 |     30 |     4 |
    | ALLEN  | 1600.00 |     30 |     3 |
    | JAMES  |  950.00 |     30 |     1 |
    | WARD   | 1250.00 |     30 |     2 |
    +--------+---------+--------+-------+
```

```
第二步：基于以上的结果继续按照deptno分组，求grade的平均值。
select
    e.deptno,avg(s.grade)
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal
group by
    e.deptno;
    +--------+--------------+
    | deptno | avg(s.grade) |
    +--------+--------------+
    |     10 |       3.6667 |
    |     20 |       2.8000 |
    |     30 |       2.5000 |
    +--------+--------------+
```

4 不准用组函数（Max），取得最高薪水

```
4、不准用组函数（Max ），取得最高薪水
第一种：sal降序，limit 1
select ename,sal from emp order by sal desc limit 1;
+-------+---------+
| ename | sal     |
+-------+---------+
| KING  | 5000.00 |
+-------+---------+

第二种方案：select max(sal) from emp;

第三种方案：表的自连接
select sal from emp where sal not in(select distinct a.sal from emp a join emp b on a.sal < b.sal);
+---------+
| sal     |
+---------+
| 5000.00 |
+---------+

select
    distinct a.sal
from
    emp a
join
    emp b
on
    a.sal < b.sal
+---------+
| sal     |
+---------+
|  800.00 |
| 1250.00 |
| 1500.00 |
| 1100.00 |
|  950.00 |
| 1300.00 |
| 1600.00 |
| 2850.00 |
| 2450.00 |
| 2975.00 |
| 3000.00 |
+---------+

a表  b表
+---------+
| sal     |
+---------+
|  800.00 |
| 1600.00 |
| 1250.00 |
| 2975.00 |
| 1250.00 |
| 2850.00 |
| 2450.00 |
| 3000.00 |
| 5000.00 |
| 1500.00 |
| 1100.00 |
|  950.00 |
| 3000.00 |
| 1300.00 |
+---------+
```

5 取得平均薪水最高的部门的部门编号

```
5、取得平均薪水最高的部门的部门编号
第一种方案：降序取第一个。
    第一步：找出每个部门的平均薪水
        select deptno,avg(sal) as avgsal from emp group by deptno;
        +--------+-------------+
        | deptno | avgsal      |
        +--------+-------------+
        |     10 | 2916.666667 |
        |     20 | 2175.000000 |
        |     30 | 1566.666667 |
        +--------+-------------+
    第二步：降序选第一个。
        select deptno,avg(sal) as avgsal from emp group by deptno order by avgsal desc limit 1;
        +--------+-------------+
        | deptno | avgsal      |
        +--------+-------------+
        |     10 | 2916.666667 |
        +--------+-------------+

第二种方案：max
    第一步：找出每个部门的平均薪水
    select deptno,avg(sal) as avgsal from emp group by deptno;
    +--------+-------------+
    | deptno | avgsal      |
    +--------+-------------+
    |     10 | 2916.666667 |
    |     20 | 2175.000000 |
    |     30 | 1566.666667 |
    +--------+-------------+
    第二步：找出以上结果中avgsal最大的值。
    select max(t.avgsal) from (select avg(sal) as avgsal from emp group by deptno) t;
    +---------------+
    | max(t.avgsal) |
    +---------------+
    |   2916.666667 |
    +---------------+
    第三步：
    select
        deptno,avg(sal) as avgsal
    from
        emp
    group by
        deptno
    having
        avgsal = (select max(t.avgsal) from (select avg(sal) as avgsal from emp group by deptno) t);
    +--------+-------------+
    | deptno | avgsal      |
    +--------+-------------+
    |     10 | 2916.666667 |
    +--------+-------------+
```

6 取得平均薪水最高的部门的部门名称

```
6、取得平均薪水最高的部门的部门名称
select
    d.dname,avg(e.sal) as avgsal
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
group by
    d.dname
order by
    avgsal desc
limit
    1;
+------------+-------------+
| dname      | avgsal      |
+------------+-------------+
| ACCOUNTING | 2916.666667 |
+------------+-------------+
```

7 求平均薪水的等级最低的部门的部门名称

```
7、求平均薪水的等级最低的部门的部门名称
第一步：找出每个部门的平均薪水
select deptno,avg(sal) as avgsal from emp group by deptno;
+--------+-------------+
| deptno | avgsal      |
+--------+-------------+
|     10 | 2916.666667 |
```

```
|     20 | 2175.000000 |
|     30 | 1566.666667 |
+--------+-------------+

第二步：找出每个部门的平均薪水的等级
以上t表和salgrade表连接，条件：t.avgsal between s.losal and s.hisal
select
    t.*,s.grade
from
    (select d.dname,avg(sal) as avgsal from emp e join dept d on e.deptno = d.deptno group by d.dname) t
join
    salgrade s
on
    t.avgsal between s.losal and s.hisal;
+------------+-------------+-------+
| dname      | avgsal      | grade |
+------------+-------------+-------+
| SALES      | 1566.666667 |     3 |
| ACCOUNTING | 2916.666667 |     4 |
| RESEARCH   | 2175.000000 |     4 |
+------------+-------------+-------+

select
    t.*,s.grade
from
    (select d.dname,avg(sal) as avgsal from emp e join dept d on e.deptno = d.deptno group by d.dname) t
join
    salgrade s
on
    t.avgsal between s.losal and s.hisal
where
    s.grade = (select grade from salgrade where (select avg(sal) as avgsal from emp group by deptno order by avgsal asc limit 1) between
losal and hisal);
+-------+-------------+-------+
| dname | avgsal      | grade |
+-------+-------------+-------+
| SALES | 1566.666667 |     3 |
+-------+-------------+-------+

抛开之前的，最低等级你怎么着？
    平均薪水最低的对应的等级一定是最低的.
    select avg(sal) as avgsal from emp group by deptno order by avgsal asc limit 1;
    +-------------+
    | avgsal      |
    +-------------+
    | 1566.666667 |
    +-------------+

    select grade from salgrade where (select avg(sal) as avgsal from emp group by deptno order by avgsal asc limit 1) between losal and
hisal;
    +-------+
    | grade |
    +-------+
    |     3 |
    +-------+
```

8 取得比普通员工的最高薪水还要高的领导人姓名

```
8、取得比普通员工(员工代码没有在 mgr 字段上出现的) 的最高薪水还要高的领导人姓名
比“普通员工的最高薪水”还要高的一定是领导！
    没毛病！！！！
mysql> select distinct mgr from emp where mgr is not null;
+------+
| mgr  |
+------+
| 7902 |
| 7698 |
| 7839 |
| 7566 |
| 7788 |
| 7782 |
+------+
员工编号没有在以上范围内的都是普通员工。

第一步：找出普通员工的最高薪水！
not in在使用的时候，后面小括号中记得排除NULL。
select max(sal) from emp where empno not in(select distinct mgr from emp where mgr is not null);
+----------+
| max(sal) |
+----------+
|  1600.00 |
+----------+

第二步：找出高于1600的
select ename,sal from emp where sal > (select max(sal) from emp where empno not in(select distinct mgr from emp where mgr is not null));
+-------+---------+
| ename | sal     |
```

```
+-------+---------+
| JONES | 2975.00 |
| BLAKE | 2850.00 |
| CLARK | 2450.00 |
| SCOTT | 3000.00 |
| KING  | 5000.00 |
| FORD  | 3000.00 |
+-------+---------+
```

9 取得薪水最高的前五名员工

```
9、取得薪水最高的前五名员工
select ename,sal from emp order by sal desc limit 5;
+-------+---------+
| ename | sal     |
+-------+---------+
| KING  | 5000.00 |
| SCOTT | 3000.00 |
| FORD  | 3000.00 |
| JONES | 2975.00 |
| BLAKE | 2850.00 |
+-------+---------+
```

10 取得薪水最高的第六到第十名员工

```
10、取得薪水最高的第六到第十名员工
select ename,sal from emp order by sal desc limit 5, 5;
+--------+---------+
| ename  | sal     |
+--------+---------+
| CLARK  | 2450.00 |
| ALLEN  | 1600.00 |
| TURNER | 1500.00 |
| MILLER | 1300.00 |
| MARTIN | 1250.00 |
+--------+---------+
```

11 取得最后入职的 5 名员工

```
11、取得最后入职的 5 名员工
日期也可以降序，升序。
    select ename,hiredate from emp order by hiredate desc limit 5;
    +--------+------------+
    | ename  | hiredate   |
    +--------+------------+
    | ADAMS  | 1987-05-23 |
    | SCOTT  | 1987-04-19 |
    | MILLER | 1982-01-23 |
    | FORD   | 1981-12-03 |
    | JAMES  | 1981-12-03 |
    +--------+------------+
```

12 取得每个薪水等级有多少员工

```
12、取得每个薪水等级有多少员工
分组count
第一步：找出每个员工的薪水等级
select
    e.ename,e.sal,s.grade
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal;
+--------+---------+-------+
| ename  | sal     | grade |
+--------+---------+-------+
| SMITH  |  800.00 |     1 |
| ALLEN  | 1600.00 |     3 |
| WARD   | 1250.00 |     2 |
| JONES  | 2975.00 |     4 |
| MARTIN | 1250.00 |     2 |
| BLAKE  | 2850.00 |     4 |
| CLARK  | 2450.00 |     4 |
| SCOTT  | 3000.00 |     4 |
| KING   | 5000.00 |     5 |
| TURNER | 1500.00 |     3 |
| ADAMS  | 1100.00 |     1 |
| JAMES  |  950.00 |     1 |
| FORD   | 3000.00 |     4 |
| MILLER | 1300.00 |     2 |
+--------+---------+-------+
```

```

第二步：继续按照grade分组统计数量
select
    s.grade ,count(*)
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal
group by
    s.grade;
+-------+----------+
| grade | count(*) |
+-------+----------+
|     1 |        3 |
|     2 |        3 |
|     3 |        2 |
|     4 |        5 |
|     5 |        1 |
+-------+----------+
```

13 面试题

```
13、面试题：
有 3 个表 S(学生表)，C（课程表），SC（学生选课表）
S（SNO，SNAME）代表（学号，姓名）
C（CNO，CNAME，CTEACHER）代表（课号，课名，教师）
SC（SNO，CNO，SCGRADE）代表（学号，课号，成绩）
问题：
1，找出没选过“黎明”老师的所有学生姓名。
2，列出 2 门以上（含2 门）不及格学生姓名及平均成绩。
3，即学过 1 号课程又学过 2 号课所有学生的姓名。
```

14 列出所有员工及领导的姓名

```
14、列出所有员工及领导的姓名
select
    a.ename '员工', b.ename '领导'
from
    emp a
left join  #左外连接
    emp b
on
    a.mgr = b.empno;
+--------+-------+
| 员工   | 领导    |
+--------+-------+
| SMITH  | FORD  |
| ALLEN  | BLAKE |
| WARD   | BLAKE |
| JONES  | KING  |
| MARTIN | BLAKE |
| BLAKE  | KING  |
| CLARK  | KING  |
| SCOTT  | JONES |
| KING   | NULL  |
| TURNER | BLAKE |
| ADAMS  | SCOTT |
| JAMES  | BLAKE |
| FORD   | JONES |
| MILLER | CLARK |
+--------+-------+
```

15 列出受雇日期早于其直接上级的所有员工的编号,姓名,部门名称

```
15、列出受雇日期早于其直接上级的所有员工的编号,姓名,部门名称
emp a 员工表
emp b 领导表
a.mgr = b.empno and a.hiredate < b.hiredate

select
    a.ename '员工', a.hiredate, b.ename '领导', b.hiredate, d.dname
from
    emp a
join
    emp b
on
    a.mgr = b.empno
join
    dept d
on
    a.deptno = d.deptno
where
```

```
    a.hiredate < b.hiredate;
+-------+------------+-------+------------+------------+
| 员工  | hiredate   | 领导  | hiredate   | dname      |
+-------+------------+-------+------------+------------+
| CLARK | 1981-06-09 | KING  | 1981-11-17 | ACCOUNTING |
| SMITH | 1980-12-17 | FORD  | 1981-12-03 | RESEARCH   |
| JONES | 1981-04-02 | KING  | 1981-11-17 | RESEARCH   |
| ALLEN | 1981-02-20 | BLAKE | 1981-05-01 | SALES      |
| WARD  | 1981-02-22 | BLAKE | 1981-05-01 | SALES      |
| BLAKE | 1981-05-01 | KING  | 1981-11-17 | SALES      |
+-------+------------+-------+------------+------------+
```

16 列出部门名称和这些部门的员工信息, 同时列出那些没有员工的部门

```
16、 列出部门名称和这些部门的员工信息, 同时列出那些没有员工的部门
select
    e.*,d.dname
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno;
+-------+--------+-----------+------+------------+---------+---------+--------+------------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL     | COMM    | DEPTNO | dname      |
+-------+--------+-----------+------+------------+---------+---------+--------+------------+
|  7782 | CLARK  | MANAGER   | 7839 | 1981-06-09 | 2450.00 |    NULL |     10 | ACCOUNTING |
|  7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000.00 |    NULL |     10 | ACCOUNTING |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-23 | 1300.00 |    NULL |     10 | ACCOUNTING |
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-17 |  800.00 |    NULL |     20 | RESEARCH   |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-02 | 2975.00 |    NULL |     20 | RESEARCH   |
|  7788 | SCOTT  | ANALYST   | 7566 | 1987-04-19 | 3000.00 |    NULL |     20 | RESEARCH   |
|  7876 | ADAMS  | CLERK     | 7788 | 1987-05-23 | 1100.00 |    NULL |     20 | RESEARCH   |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-03 | 3000.00 |    NULL |     20 | RESEARCH   |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-20 | 1600.00 |  300.00 |     30 | SALES      |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-22 | 1250.00 |  500.00 |     30 | SALES      |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-28 | 1250.00 | 1400.00 |     30 | SALES      |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850.00 |    NULL |     30 | SALES      |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-09-08 | 1500.00 |    0.00 |     30 | SALES      |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-03 |  950.00 |    NULL |     30 | SALES      |
|  NULL | NULL   | NULL      | NULL | NULL       |    NULL |    NULL |   NULL | OPERATIONS |
+-------+--------+-----------+------+------------+---------+---------+--------+------------+
```

17 列出至少有 5 个员工的所有部门

```
17、列出至少有 5 个员工的所有部门
按照部门编号分组，计数，筛选出 >= 5
select
    deptno
from
    emp
group by
    deptno
having
    count(*) >= 5;
+--------+
| deptno |
+--------+
|     20 |
|     30 |
+--------+
```

18 列出薪金比"SMITH" 多的所有员工信息

```
18、列出薪金比"SMITH" 多的所有员工信息
select ename,sal from emp where sal > (select sal from emp where ename = 'SMITH');
+--------+---------+
| ename  | sal     |
+--------+---------+
| ALLEN  | 1600.00 |
| WARD   | 1250.00 |
| JONES  | 2975.00 |
| MARTIN | 1250.00 |
| BLAKE  | 2850.00 |
| CLARK  | 2450.00 |
| SCOTT  | 3000.00 |
| KING   | 5000.00 |
| TURNER | 1500.00 |
| ADAMS  | 1100.00 |
| JAMES  |  950.00 |
| FORD   | 3000.00 |
| MILLER | 1300.00 |
+--------+---------+
```

19 列出所有"CLERK"(办事员) 的姓名及其部门名称, 部门的人数

```
19、 列出所有"CLERK"( 办事员) 的姓名及其部门名称, 部门的人数
select ename,job from emp where job = 'CLERK';
+--------+-------+
| ename  | job   |
+--------+-------+
| SMITH  | CLERK |
| ADAMS  | CLERK |
| JAMES  | CLERK |
| MILLER | CLERK |
+--------+-------+

select
    e.ename,e.job,d.dname,d.deptno
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
where
    e.job = 'CLERK';
+--------+-------+------------+--------+
| ename  | job   | dname      | deptno |
+--------+-------+------------+--------+
| MILLER | CLERK | ACCOUNTING |     10 |
| SMITH  | CLERK | RESEARCH   |     20 |
| ADAMS  | CLERK | RESEARCH   |     20 |
| JAMES  | CLERK | SALES      |     30 |
+--------+-------+------------+--------+


#每个部门的人数?
select deptno, count(*) as deptcount from emp group by deptno;
+--------+-----------+
| deptno | deptcount |
+--------+-----------+
|     10 |         3 |
|     20 |         5 |
|     30 |         6 |
+--------+-----------+

select
    t1.*,t2.deptcount
from
    (select
        e.ename,e.job,d.dname,d.deptno
    from
        emp e
    join
        dept d
    on
        e.deptno = d.deptno
    where
        e.job = 'CLERK') t1
join
    (select deptno, count(*) as deptcount from emp group by deptno) t2
on
    t1.deptno = t2.deptno;
+--------+-------+------------+--------+-----------+
| ename  | job   | dname      | deptno | deptcount |
+--------+-------+------------+--------+-----------+
| MILLER | CLERK | ACCOUNTING |     10 |         3 |
| SMITH  | CLERK | RESEARCH   |     20 |         5 |
| ADAMS  | CLERK | RESEARCH   |     20 |         5 |
| JAMES  | CLERK | SALES      |     30 |         6 |
+--------+-------+------------+--------+-----------+
```

20 列出最低薪金大于 1500 的各种工作及从事此工作的全部雇员人数

```
20、列出最低薪金大于 1500 的各种工作及从事此工作的全部雇员人数
按照工作岗位分组求最小值。
select job,count(*) from emp group by job having min(sal) > 1500;
+-----------+----------+
| job       | count(*) |
+-----------+----------+
| ANALYST   |        2 |
| MANAGER   |        3 |
| PRESIDENT |        1 |
+-----------+----------+
```

21 列出在部门"SALES"< 销售部> 工作的员工的姓名, 假定不知道销售部的部门编号.

```
21、列出在部门"SALES"< 销售部> 工作的员工的姓名, 假定不知道销售部的部门编号.
select ename from emp where deptno = (select deptno from dept where dname = 'SALES');
+--------+
| ename  |
+--------+
| ALLEN  |
| WARD   |
| MARTIN |
| BLAKE  |
| TURNER |
| JAMES  |
+--------+
```

22 列出薪金高于公司平均薪金的所有员工, 所在部门, 上级领导, 雇员的工资等级.

```
22、列出薪金高于公司平均薪金的所有员工, 所在部门, 上级领导, 雇员的工资等级.
select
    e.ename '员工',d.dname,l.ename '领导',s.grade
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
left join
    emp l
on
    e.mgr = l.empno
join
    salgrade s
on
    e.sal between s.losal and s.hisal
where
    e.sal > (select avg(sal) from emp);
+-------+------------+-------+-------+
| 员工     | dname      | 领导    | grade |
+-------+------------+-------+-------+
| JONES | RESEARCH   | KING  |     4 |
| BLAKE | SALES      | KING  |     4 |
| CLARK | ACCOUNTING | KING  |     4 |
| SCOTT | RESEARCH   | JONES |     4 |
| KING  | ACCOUNTING | NULL  |     5 |
| FORD  | RESEARCH   | JONES |     4 |
+-------+------------+-------+-------+
```

23 列出与"SCOTT" 从事相同工作的所有员工及部门名称

```
23、 列出与"SCOTT" 从事相同工作的所有员工及部门名称
select job from emp where ename = 'SCOTT';
+---------+
| job     |
+---------+
| ANALYST |
+---------+

select
    e.ename,e.job,d.dname
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
where
    e.job = (select job from emp where ename = 'SCOTT')
and
    e.ename <> 'SCOTT';
+-------+---------+----------+
| ename | job     | dname    |
+-------+---------+----------+
| FORD  | ANALYST | RESEARCH |
+-------+---------+----------+
```

24 列出薪金等于部门 30 中员工的薪金的其他员工的姓名和薪金.

```
24、列出薪金等于部门 30 中员工的薪金的其他员工的姓名和薪金.
select distinct sal from emp where deptno = 30;
+---------+
| sal     |
+---------+
| 1600.00 |
| 1250.00 |
```

```
| 2850.00 |
| 1500.00 |
|  950.00 |
+---------+

select
    ename,sal
from
    emp
where
    sal in(select distinct sal from emp where deptno = 30)
and
    deptno <> 30;

Empty set (0.00 sec)
```

25 列出薪金高于在部门 30 工作的所有员工的薪金的员工姓名和薪金. 部门名称

```
25、列出薪金高于在部门 30 工作的所有员工的薪金的员工姓名和薪金. 部门名称
select max(sal) from emp where deptno = 30;
+----------+
| max(sal) |
+----------+
|  2850.00 |
+----------+

select
    e.ename,e.sal,d.dname
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
where
    e.sal > (select max(sal) from emp where deptno = 30);
+-------+---------+------------+
| ename | sal     | dname      |
+-------+---------+------------+
| KING  | 5000.00 | ACCOUNTING |
| JONES | 2975.00 | RESEARCH   |
| SCOTT | 3000.00 | RESEARCH   |
| FORD  | 3000.00 | RESEARCH   |
+-------+---------+------------+
```

26 列出在每个部门工作的员工数量, 平均工资和平均服务期限

```
26、列出在每个部门工作的员工数量, 平均工资和平均服务期限
没有员工的部门，部门人数是0
select
    d.deptno, count(e.ename) ecount,ifnull(avg(e.sal),0) as avgsal, ifnull(avg(timestampdiff(YEAR, hiredate, now())), 0) as avgservicetime
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno
group by
    d.deptno;
+--------+--------+-------------+----------------+
| deptno | ecount | avgsal      | avgservicetime |
+--------+--------+-------------+----------------+
|     10 |      3 | 2916.666667 |        38.0000 |
|     20 |      5 | 2175.000000 |        35.8000 |
|     30 |      6 | 1566.666667 |        38.3333 |
|     40 |      0 |    0.000000 |         0.0000 |
+--------+--------+-------------+----------------+

在mysql当中怎么计算两个日期的“年差”，差了多少年？
    TimeStampDiff(间隔类型, 前一个日期, 后一个日期)
    timestampdiff(YEAR, hiredate, now())

    间隔类型：
        SECOND   秒，
        MINUTE   分钟，
        HOUR   小时，
        DAY   天，
        WEEK   星期
        MONTH   月，
        QUARTER   季度，
        YEAR   年
```

27 列出所有员工的姓名、部门名称和工资。

```
27、 列出所有员工的姓名、部门名称和工资。
select
    e.ename,d.dname,e.sal
from
    emp e
join
    dept d
on
    e.deptno = d.deptno;
+--------+------------+---------+
| ename  | dname      | sal     |
+--------+------------+---------+
| CLARK  | ACCOUNTING | 2450.00 |
| KING   | ACCOUNTING | 5000.00 |
| MILLER | ACCOUNTING | 1300.00 |
| SMITH  | RESEARCH   |  800.00 |
| JONES  | RESEARCH   | 2975.00 |
| SCOTT  | RESEARCH   | 3000.00 |
| ADAMS  | RESEARCH   | 1100.00 |
| FORD   | RESEARCH   | 3000.00 |
| ALLEN  | SALES      | 1600.00 |
| WARD   | SALES      | 1250.00 |
| MARTIN | SALES      | 1250.00 |
| BLAKE  | SALES      | 2850.00 |
| TURNER | SALES      | 1500.00 |
| JAMES  | SALES      |  950.00 |
+--------+------------+---------+
```

28 列出所有部门的详细信息和人数

```
28、列出所有部门的详细信息和人数
select
    d.deptno,d.dname,d.loc,count(e.ename)
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno
group by
    d.deptno,d.dname,d.loc;
+--------+------------+----------+----------------+
| deptno | dname      | loc      | count(e.ename) |
+--------+------------+----------+----------------+
|     10 | ACCOUNTING | NEW YORK |              3 |
|     20 | RESEARCH   | DALLAS   |              5 |
|     30 | SALES      | CHICAGO  |              6 |
|     40 | OPERATIONS | BOSTON   |              0 |
+--------+------------+----------+----------------+
```

29 列出各种工作的最低工资及从事此工作的雇员姓名

```
29、列出各种工作的最低工资及从事此工作的雇员姓名
select
    job,min(sal) as minsal
from
    emp
group by
    job;
+-----------+----------+
| job       | minsal   |
+-----------+----------+
| ANALYST   |  3000.00 |
| CLERK     |   800.00 |
| MANAGER   |  2450.00 |
| PRESIDENT |  5000.00 |
| SALESMAN  |  1250.00 |
+-----------+----------+

emp e和以上t连接

select
    e.ename,t.*
from
    emp e
join
    (select
        job,min(sal) as minsal
    from
        emp
    group by
        job) t
on
```

```
    e.job = t.job and e.sal = t.minsal;
+--------+-----------+---------+
| ename  | job       | minsal  |
+--------+-----------+---------+
| SMITH  | CLERK     |  800.00 |
| WARD   | SALESMAN  | 1250.00 |
| MARTIN | SALESMAN  | 1250.00 |
| CLARK  | MANAGER   | 2450.00 |
| SCOTT  | ANALYST   | 3000.00 |
| KING   | PRESIDENT | 5000.00 |
| FORD   | ANALYST   | 3000.00 |
+--------+-----------+---------+
```

30 列出各个部门的 MANAGER(领导) 的最低薪金

```
30、列出各个部门的 MANAGER( 领导) 的最低薪金
select
    deptno, min(sal)
from
    emp
where
    job = 'MANAGER'
group by
    deptno;
+--------+----------+
| deptno | min(sal) |
+--------+----------+
|     10 |  2450.00 |
|     20 |  2975.00 |
|     30 |  2850.00 |
+--------+----------+
```

31 列出所有员工的 年工资, 按 年薪从低到高排序

```
31、列出所有员工的 年工资, 按 年薪从低到高排序
select
    ename,(sal + ifnull(comm,0)) * 12 as yearsal
from
    emp
order by
    yearsal asc;
+--------+----------+
| ename  | yearsal  |
+--------+----------+
| SMITH  |  9600.00 |
| JAMES  | 11400.00 |
| ADAMS  | 13200.00 |
| MILLER | 15600.00 |
| TURNER | 18000.00 |
| WARD   | 21000.00 |
| ALLEN  | 22800.00 |
| CLARK  | 29400.00 |
| MARTIN | 31800.00 |
| BLAKE  | 34200.00 |
| JONES  | 35700.00 |
| FORD   | 36000.00 |
| SCOTT  | 36000.00 |
| KING   | 60000.00 |
+--------+----------+
```

32 求出员工领导的薪水超过3000的员工名称与领导

```
32、求出员工领导的薪水超过3000的员工名称与领导
select
    a.ename '员工',b.ename '领导'
from
    emp a
join
    emp b
on
    a.mgr = b.empno
where
    b.sal > 3000;
+-------+------+
| 员工  | 领导  |
+-------+------+
| JONES | KING |
| BLAKE | KING |
| CLARK | KING |
+-------+------+
```

33 求出部门名称中, 带'S'字符的部门员工的工资合计、部门人数

```
33、求出部门名称中, 带'S'字符的部门员工的工资合计、部门人数
select
    d.deptno,d.dname,d.loc,count(e.ename),ifnull(sum(e.sal),0) as sumsal
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno
where
    d.dname like '%S%'
group by
    d.deptno,d.dname,d.loc;
+--------+------------+---------+----------------+----------+
| deptno | dname      | loc     | count(e.ename) | sumsal   |
+--------+------------+---------+----------------+----------+
|     20 | RESEARCH   | DALLAS  |              5 | 10875.00 |
|     30 | SALES      | CHICAGO |              6 |  9400.00 |
|     40 | OPERATIONS | BOSTON  |              0 |     0.00 |
+--------+------------+---------+----------------+----------+
```

34 给任职日期超过 30 年的员工加薪 10%.

```
34、给任职日期超过 30 年的员工加薪 10%.
update emp set sal = sal * 1.1 where timestampdiff(YEAR, hiredate, now()) > 30;
```