数据库

1 数据库概念（了解）

1.1 什么是数据库

数据库就是用来**存储和管理**数据的仓库！
数据库存储数据的优先：
数据库的好处：

- 可存储大量数据；
- 方便检索；
- 保持数据的一致性、完整性；
- 安全，可共享；
- 通过组合分析，可产生新数据。

1.2 数据库的发展历程

- 没有数据库，使用磁盘文件存储数据；txt,word 文档
- 层次结构模型数据库，树形结构；
- 网状结构模型数据库；
- **关系结构模型数据库：使用二维表格来存储数据；excel 表格**
- 关系-对象模型数据库；

MySQL 就是关系型数据库！

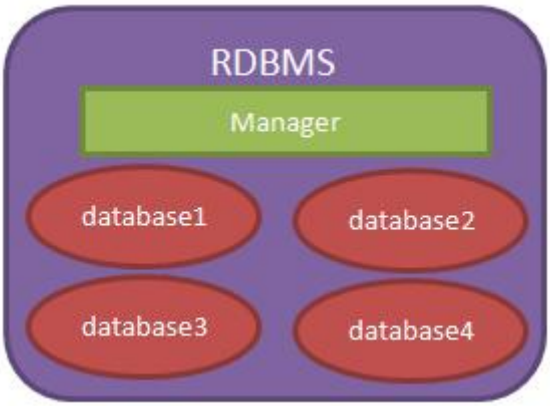
1.3 常见数据库

- Oracle（神喻）：甲骨文（最高！）；
- DB2：IBM；
- SQL Server：微软；
- Sybase：赛尔斯；
- MySQL：甲骨文；

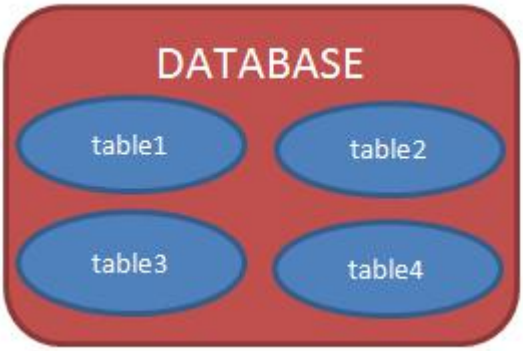
1.4 理解数据库

- RDBMS = 管理员（manager）+仓库（database）
- database = N 个 table
- table：
  - 表结构：定义表的列名和列类型！
  - 表记录：一行一行的记录！

我们现在所说的数据库泛指“关系型数据库管理系统（RDBMS - Relational database management system）”，即“数据库服务器”。



当我们安装了数据库服务器后，就可以在数据库服务器中创建数据库，每个数据库中还可以包含多张表。



数据库表就是一个多行多列的表格。在创建表时，需要指定表的列数，以及列名称，列类型等信息。而不用指定表格的行数，行数是没有上限的。下面是 tab_student 表的结构：

s_id	varchar(10)
s_name	varchar(20)
s_age	int
s_sex	varchar(10)

当把表格创建好了之后，就可以向表格中添加数据了。向表格添加数据是以行为单位的！下面是 s_student 表的记录：

s_id	s_name	s_age	s_sex
S_1001	zhangSan	23	male
S_1002	liSi	32	female
S_1003	wangWu	44	male

大家要学会区分什么是表结构，什么是表记录。

1.5 应用程序与数据库

应用程序使用数据库完成对数据的存储！

2 安装 MySQL 数据库

2.1 安装 MySQL

参考：MySQL 安装图解.doc

2.2 MySQL 目录结构

MySQL 的数据存储目录为 data，data 目录通常在 C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.5\data 位置。在 data 下的每个目录都代表一个数据库。

MySQL 的安装目录下：

- bin 目录中都是可执行文件；
- my.ini 文件是 MySQL 的配置文件；

3 基本命令

3.1 启动和关闭 mysql 服务器

- 启动：net start mysql；
- 关闭：net stop mysql；

在启动 mysql 服务后，打开 windows 任务管理器，会有一个名为 mysqld.exe 的进程运行，所以 mysqld.exe 才是 MySQL 服务器程序。

3.2 客户端登录退出 mysql

在启动 MySQL 服务器后，我们需要使用管理员用户登录 MySQL 服务器，然后来对服务器进行

操作。登录 MySQL 需要使用 MySQL 的客户端程序：mysql.exe

- 登录：mysql -u root -p 123 -h localhost；
  - -u：后面的 root 是用户名，这里使用的是超级管理员 root；
  - -p：后面的 123 是密码，这是在安装 MySQL 时就已经指定的密码；
  - -h：后面给出的 localhost 是服务器主机名，它是可以省略的，例如：mysql -u root -p 123；
  - -P: 端口号
- 退出：quit 或 exit；

在登录成功后，打开 windows 任务管理器，会有一个名为 mysql.exe 的进程运行，所以 mysql.exe 是客户端程序。

SQL 语句

1 SQL 概述

1.1 什么是 SQL

SQL（Structured Query Language）是“结构化查询语言”，它是对关系型数据库的操作语言。它可以应用到所有关系型数据库中，例如：MySQL、Oracle、SQL Server 等。SQ 标准（ANSI/ISO）有：

- SQL-92：1992 年发布的 SQL 语言标准；
- SQL:1999：1999 年发布的 SQL 语言标签；
- SQL:2003：2003 年发布的 SQL 语言标签；

这些标准就与 JDK 的版本一样，在新的版本中总要有一些语法的变化。不同时期的数据库对不同标准做了实现。

虽然 SQL 可以用在所有关系型数据库中，但很多数据库还都有标准之后的一些语法，我们可以称之为“方言”。例如 MySQL 中的 LIMIT 语句就是 MySQL 独有的方言，其它数据库都不支持！当然，Oracle 或 SQL Server 都有自己的方言。

1.2 语法要求

- SQL 语句可以单行或多行书写，以分号结尾；
- 可以用空格和缩进来来增强语句的可读性；
- 关键字不区别大小写，建议使用大写；

2 分类

- DDL（Data Definition Language）：数据定义语言，用来定义数据库对象：库、表、列等；
- DML（Data Manipulation Language）：数据操作语言，用来定义数据库记录（数据）；
- DCL（Data Control Language）：数据控制语言，用来定义访问权限和安全级别；
- DQL（Data Query Language）：数据查询语言，用来查询记录（数据）。

3 DDL

3.1 基本操作

- 查看所有数据库名称：SHOW DATABASES；
- 切换数据库：USE mydb1，切换到 mydb1 数据库；

3.2 操作数据库

- 创建数据库：CREATE DATABASE [IF NOT EXISTS] mydb1；

创建数据库，例如：CREATE DATABASE mydb1，创建一个名为 mydb1 的数据库。如果这个数据已经存在，那么会报错。例如 CREATE DATABASE IF NOT EXISTS mydb1，在名为 mydb1 的数据库不存在时创建该库，这样可以避免报错。

- 删除数据库：DROP DATABASE [IF EXISTS] mydb1；

删除数据库，例如：DROP DATABASE mydb1，删除名为 mydb1 的数据库。如果这个数据库不存在，那么会报错。DROP DATABASE IF EXISTS mydb1，就算 mydb1 不存在，也不会的报错。

- 修改数据库编码：ALTER DATABASE mydb1 CHARACTER SET utf8

修改数据库 mydb1 的编码为 utf8。注意，在 MySQL 中所有的 UTF-8 编码都不能使用中间的“-”，即 UTF-8 要书写为 UTF8。

3.3 数据类型

MySQL 与 Java 一样，也有数据类型。MySQL 中数据类型主要应用在列上。

常用类型：

- int：整型(放的是整数，比如：1,2,3)
- flot
- double：浮点型，例如 double(5,2)表示最多 5 位，其中必须有 2 位小数，即最大值为 999.99；
  - 放的都是小数类型的数
- decimal：泛型型，在表单钱方面使用该类型，因为不会出现精度缺失问题；
  - 也是小数类型，不会出现精度丢失。
- char：固定长度字符串类型；
- varchar：可变长度字符串类型；
- text：字符串类型；
- blob：字节类型；
- date：日期类型，格式为：yyyy-MM-dd；
- time：时间类型，格式为：hh:mm:ss
- timestamp：时间戳类型；
- 1970-1-1

数值、字符、日期

3.4 操作表

- 创建表：

```
CREATE TABLE 表名(
    列名 列类型,
    列名 列类型,
    ......
);
```

例如：

```
CREATE TABLE stu(
    sid       CHAR(6),
    sname     VARCHAR(20),
    age       INT,
    gender    VARCHAR(10)
);
```

再例如：

```
CREATE TABLE emp(
    eid       CHAR(6),
    ename     VARCHAR(50),
    age       INT,
    gender    VARCHAR(6),
    birthday  DATE,
    hiredate  DATE,
    salary    DECIMAL(7,2),
    resume    VARCHAR(1000)
);
```

- 查看当前数据库中所有表名称：SHOW TABLES；
- 查看指定表的创建语句：SHOW CREATE TABLE emp，查看 emp 表的创建语句；
- select database(); 查看当前数据库
- 查看表结构：DESC emp，查看 emp 表结构；
- 删除表：DROP TABLE emp，删除 emp 表；
- 修改表：

1. 修改之添加列：给 stu 表添加 classname 列：
   ALTER TABLE stu ADD (classname varchar(100));
2. 修改之修改列类型：修改 stu 表的 gender 列类型为 CHAR(2)：
   ALTER TABLE stu MODIFY gender CHAR(2);
3. 修改之修改列名：修改 stu 表的 gender 列名为 sex：
   ALTER TABLE stu change gender sex CHAR(2);
4. 修改之删除列：删除 stu 表的 classname 列：
   ALTER TABLE stu DROP classname;
5. 修改之修改表名称：修改 stu 表名称为 student：
   ALTER TABLE stu RENAME TO student;

4 DML

4.1 插入数据

语法：

INSERT INTO 表名(列名 1,列名 2, …) VALUES(值 1, 值 2)

```
INSERT INTO stu(sid, sname,age,gender) VALUES('s_1001', 'zhangSan', 23, 'male');
INSERT INTO stu(sid, sname) VALUES('s_1001', 'zhangSan');
```

语法：

INSERT INTO 表名 VALUES(值 1,值 2,…)

因为没有指定要插入的列，表示按创建表时列的顺序插入所有列的值：

```
INSERT INTO stu VALUES('s_1002', 'liSi', 32, 'female');
```

注意：所有字符串数据必须使用单引用！

4.2 修改数据

语法：

UPDATE 表名 SET 列名 1=值 1, … 列名 n=值 n [WHERE 条件]

```
UPDATE stu SET sname='zhangSanSan', age='32', gender='female' WHERE sid='s_1001';
UPDATE stu SET sname='liSi', age='20' WHERE age>50 AND gender='male';
UPDATE stu SET sname='wangWu', age='30' WHERE age>60 OR gender='female';
UPDATE stu SET gender='female' WHERE gender IS NULL
UPDATE stu SET age=age+1 WHERE sname='zhaoLiu';
```

4.3 删除数据

语法：

DELETE FROM 表名 [WHERE 条件]

```
DELETE FROM stu WHERE sid='s_1001'003B
DELETE FROM stu WHERE sname='chenQi' OR age > 30;
DELETE FROM stu;
```

语法：

TRUNCATE TABLE 表名

```
TRUNCATE TABLE stu;
```

虽然 TRUNCATE 和 DELETE 都可以删除表的所有记录，但有原理不同。DELETE 的效率没有 TRUNCATE 高！

TRUNCATE 其实属性 DDL 语句，因为它是先 DROP TABLE，再 CREATE TABLE。而且 TRUNCATE 删除的记录是无法回滚的，但 DELETE 删除的记录是可以回滚的（回滚是事务的知识！）。

5 DCL

5.1 创建用户

语法：

CREATE USER 用户名@地址 IDENTIFIED BY '密码';

```
CREATE USER user1@localhost IDENTIFIED BY ‘123’;
CREATE USER user2@’%’ IDENTIFIED BY ‘123’;
```

5.2 给用户授权

语法：

GRANT 权限 1, … , 权限 n ON 数据库.* TO 用户名

```
GRANT CREATE,ALTER,DROP,INSERT,UPDATE,DELETE,SELECT ON mydb1.* TO user1@localhost;
GRANT ALL ON mydb1.* TO user2@localhost;
```

5.3 撤销授权

语法：

REVOKE 权限 1, … , 权限 n ON 数据库.* FORM 用户名

```
REVOKE CREATE,ALTER,DROP ON mydb1.* FROM user1@localhost;
```

5.4 查看用户权限

语法：

SHOW GRANTS FOR 用户名

```
SHOW GRANTS FOR user1@localhost;
```

5.5 删除用户

语法：

DROP USER 用户名

```
DROP USER user1@localhost;
```

5.6 修改用户密码

语法：

USE mysql;

UPDATE USER SET PASSWORD=PASSWORD(‘密码’) WHERE User=’用户名’ and Host=’IP’;

FLUSH PRIVILEGES;

```
UPDATE USER SET PASSWORD=PASSWORD('1234') WHERE User='user2' and Host=’localhost’;
FLUSH PRIVILEGES;
```

数据查询语法（DQL）

DQL 就是数据查询语言，数据库执行 DQL 语句不会对数据进行改变，而是让数据库发送结果集给客户端。

语法：

```
SELECT selection_list /*要查询的列名称*/
   FROM table_list /*要查询的表名称*/
   WHERE condition /*行条件*/
   GROUP BY grouping_columns /*对结果分组*/
   HAVING condition /*分组后的行条件*/
   ORDER BY sorting_columns /*对结果分组*/
   LIMIT offset_start, row_count /*结果限定*/
```

创建名：

- 学生表：stu

字段名称	字段类型	说明
sid	char(6)	学生学号
sname	varchar(50)	学生姓名
age	int	学生年龄
gender	varchar(50)	学生性别

```
CREATE TABLE stu (
    sid   CHAR(6),
    sname    VARCHAR(50),
    age      INT,
    gender   VARCHAR(50)
);
```

```
INSERT INTO stu VALUES('S_1001', 'liuYi', 35, 'male');
INSERT INTO stu VALUES('S_1002', 'chenEr', 15, 'female');
INSERT INTO stu VALUES('S_1003', 'zhangSan', 95, 'male');
INSERT INTO stu VALUES('S_1004', 'liSi', 65, 'female');
INSERT INTO stu VALUES('S_1005', 'wangWu', 55, 'male');
INSERT INTO stu VALUES('S_1006', 'zhaoLiu', 75, 'female');
INSERT INTO stu VALUES('S_1007', 'sunQi', 25, 'male');
INSERT INTO stu VALUES('S_1008', 'zhouBa', 45, 'female');
INSERT INTO stu VALUES('S_1009', 'wuJiu', 85, 'male');
INSERT INTO stu VALUES('S_1010', 'zhengShi', 5, 'female');
INSERT INTO stu VALUES('S_1011', 'xxx', NULL, NULL);
```

- 雇员表：emp

字段名称	字段类型	说明
empno	int	员工编号
ename	varchar(50)	员工姓名

job	varchar(50)	员工工作
mgr	int	领导编号
hiredate	date	入职日期
sal	decimal(7,2)	月薪
comm	decimal(7,2)	奖金
deptno	int	部分编号

```
CREATE TABLE emp(
        empno       INT,
        ename       VARCHAR(50),
        job     VARCHAR(50),
        mgr     INT,
        hiredate    DATE,
        sal     DECIMAL(7,2),
        comm        decimal(7,2),
        deptno      INT
) ;
```

```
INSERT INTO emp values(7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);
INSERT INTO emp values(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);
INSERT INTO emp values(7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);
INSERT INTO emp values(7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);
INSERT INTO emp values(7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250,1400,30);
INSERT INTO emp values(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);
INSERT INTO emp values(7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);
INSERT INTO emp values(7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20);
INSERT INTO emp values(7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);
INSERT INTO emp values(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);
INSERT INTO emp values(7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20);
INSERT INTO emp values(7900,'JAMES','CLERK',7698,'1981-12-03',950,NULL,30);
INSERT INTO emp values(7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20);
INSERT INTO emp values(7934,'MILLER','CLERK',7782,'1982-01-23',1300,NULL,10);
```

- 部分表：dept

字段名称	字段类型	说明
deptno	int	部分编码
dname	varchar(50)	部分名称
loc	varchar(50)	部分所在地点

```
CREATE TABLE dept(
        deptno      INT,
        dname       varchar(14),
        loc     varchar(13)
);
```

```
INSERT INTO dept values(10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO dept values(20, 'RESEARCH', 'DALLAS');
INSERT INTO dept values(30, 'SALES', 'CHICAGO');
INSERT INTO dept values(40, 'OPERATIONS', 'BOSTON');
```

1 基础查询

1.1 查询所有列

```
SELECT * FROM stu;
```

1.2 查询指定列

```
SELECT sid, sname, age FROM stu;
```

2 条件查询

2.1 条件查询介绍

条件查询就是在查询时给出 WHERE 子句，在 WHERE 子句中可以使用如下运算符及关键字：

- =、!=、<>、<、<=、>、>=；
- BETWEEN…AND；
- IN(set)；
- IS NULL；
- AND；
- OR；
- NOT；

2.2 查询性别为女，并且年龄 50 的记录

```
SELECT * FROM stu
WHERE gender='female' AND ge<50;
```

2.3 查询学号为 S_1001，或者姓名为 liSi 的记录

```
SELECT * FROM stu
WHERE sid ='S_1001' OR sname='liSi';
```

2.4 查询学号为 S_1001，S_1002，S_1003 的记录

```
SELECT * FROM stu
WHERE sid IN ('S_1001','S_1002','S_1003');
```

2.5 查询学号不是 S_1001，S_1002，S_1003 的记录

```
SELECT * FROM tab_student
WHERE s_number NOT IN ('S_1001','S_1002','S_1003');
```

2.6 查询年龄为 null 的记录

```
SELECT * FROM stu
WHERE age IS NULL;
```

2.7 查询年龄在 20 到 40 之间的学生记录

```
SELECT *
FROM stu
WHERE age>=20 AND age<=40;
```

或者

```
SELECT *
FROM stu
WHERE age BETWEEN 20 AND 40;
```

2.8 查询性别非男的学生记录

```
SELECT *
FROM stu
WHERE gender!='male';
```

或者

```
SELECT *
FROM stu
WHERE gender<>'male';
```

或者

```
SELECT *
FROM stu
WHERE NOT gender='male';
```

2.9 查询姓名不为 null 的学生记录

```
SELECT *
FROM stu
WHERE NOT sname IS NULL;
```

或者

```
SELECT *
FROM stu
```

```
WHERE sname IS NOT NULL;
```

3 模糊查询

当想查询姓名中包含 a 字母的学生时就需要使用模糊查询了。模糊查询需要使用关键字 LIKE。

3.1 查询姓名由 5 个字母构成的学生记录

```
SELECT *
FROM stu
WHERE sname LIKE '_____';
```

模糊查询必须使用 LIKE 关键字。其中 “_”匹配任意一个字母，5 个“_”表示 5 个任意字母。

3.2 查询姓名由 5 个字母构成，并且第 5 个字母为“i”的学生记录

```
SELECT *
FROM stu
WHERE sname LIKE '____i';
```

3.3 查询姓名以“z”开头的学生记录

```
SELECT *
FROM stu
WHERE sname LIKE 'z%';
```

其中“%”匹配 0~n 个任何字母。

3.4 查询姓名中第 2 个字母为“i”的学生记录

```
SELECT *
FROM stu
WHERE sname LIKE '_i%';
```

3.5 查询姓名中包含“a”字母的学生记录

```
SELECT *
FROM stu
WHERE sname LIKE '%a%';
```

4 字段控制查询

4.1 去除重复记录

去除重复记录（两行或两行以上记录中系列的上的数据都相同），例如 emp 表中 sal 字段就存在

相同的记录。当只查询 emp 表的 sal 字段时，那么会出现重复记录，那么想去除重复记录，需要使用 DISTINCT：

```
SELECT DISTINCT sal FROM emp;
```

4.2 查看雇员的月薪与佣金之和

因为 sal 和 comm 两列的类型都是数值类型，所以可以做加运算。如果 sal 或 comm 中有一个字段不是数值类型，那么会出错。

```
SELECT *,sal+comm FROM emp;
```

comm 列有很多记录的值为 NULL，因为任何东西与 NULL 相加结果还是 NULL，所以结算结果可能会出现 NULL。下面使用了把 NULL 转换成数值 0 的函数 IFNULL：

```
SELECT *,sal+IFNULL(comm,0) FROM emp;
```

4.3 给列名添加别名

在上面查询中出现列名为 sal+IFNULL(comm,0)，这很不美观，现在我们给这一列给出一个别名，为 total：

```
SELECT *, sal+IFNULL(comm,0) AS total FROM emp;
```

给列起别名时，是可以省略 AS 关键字的：

```
SELECT *,sal+IFNULL(comm,0) total FROM emp;
```

5 排序

5.1 查询所有学生记录，按年龄升序排序

```
SELECT *
FROM stu
ORDER BY sage ASC;
```

或者

```
SELECT *
FROM stu
ORDER BY sage;
```

5.2 查询所有学生记录，按年龄降序排序

```
SELECT *
FROM stu
ORDER BY age DESC;
```

5.3 查询所有雇员，按月薪降序排序，如果月薪相同时，按编号升序排序

```
SELECT * FROM emp
```

```
ORDER BY sal DESC,empno ASC;
```

6 聚合函数

聚合函数是用来做纵向运算的函数：

- COUNT()：统计指定列不为 NULL 的记录行数；
- MAX()：计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；
- MIN()：计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；
- SUM()：计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为 0；
- AVG()：计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为 0；

6.1 COUNT

当需要纵向统计时可以使用 COUNT()。

- 查询 emp 表中记录数：

```
SELECT COUNT(*) AS cnt FROM emp;
```

- 查询 emp 表中有佣金的人数：

```
SELECT COUNT(comm) cnt FROM emp;
```

注意，因为 count()函数中给出的是 comm 列，那么只统计 comm 列非 NULL 的行数。

- 查询 emp 表中月薪大于 2500 的人数：

```
SELECT COUNT(*) FROM emp
WHERE sal > 2500;
```

- 统计月薪与佣金之和大于 2500 元的人数：

```
SELECT COUNT(*) AS cnt FROM emp WHERE sal+IFNULL(comm,0) > 2500;
```

- 查询有佣金的人数，以及有领导的人数：

```
SELECT COUNT(comm), COUNT(mgr) FROM emp;
```

6.2 SUM 和 AVG

当需要纵向求和时使用 sum()函数。

- 查询所有雇员月薪和：

```
SELECT SUM(sal) FROM emp;
```

- 查询所有雇员月薪和，以及所有雇员佣金和：

```
SELECT SUM(sal), SUM(comm) FROM emp;
```

- 查询所有雇员月薪+佣金和：

```
SELECT SUM(sal+IFNULL(comm,0)) FROM emp;
```

- 统计所有员工平均工资：

```
SELECT SUM(sal), COUNT(sal) FROM emp;
```

或者

```
SELECT AVG(sal) FROM emp;
```

6.3 MAX 和 MIN

- 查询最高工资和最低工资：

```
SELECT MAX(sal), MIN(sal) FROM emp;
```

7 分组查询

当需要分组查询时需要使用 GROUP BY 子句，例如查询每个部门的工资和，这说明要使用部分来分组。

7.1 分组查询

- 查询每个部门的部门编号和每个部门的工资和：

```
SELECT deptno, SUM(sal)
FROM emp
GROUP BY deptno;
```

- 查询每个部门的部门编号以及每个部门的人数：

```
SELECT deptno,COUNT(*)
FROM emp
GROUP BY deptno;
```

- 查询每个部门的部门编号以及每个部门工资大于 1500 的人数：

```
SELECT deptno,COUNT(*)
FROM emp
WHERE sal>1500
GROUP BY deptno;
```

7.2 HAVING 子句

- 查询工资总和大于 9000 的部门编号以及工资和：

```
SELECT deptno, SUM(sal)
FROM emp
GROUP BY deptno
HAVING SUM(sal) > 9000;
```

注意，WHERE 是对分组前记录的条件，如果某行记录没有满足 WHERE 子句的条件，那么这行

记录不会参加分组；而 HAVING 是对分组后数据的约束。

8 LIMIT

LIMIT 用来限定查询结果的起始行，以及总行数。

8.1 查询 5 行记录，起始行从 0 开始

```
SELECT * FROM emp LIMIT 0, 5;
```

注意，起始行从 0 开始，即第一行开始！

8.2 查询 10 行记录，起始行从 3 开始

```
SELECT * FROM emp LIMIT 3, 10;
```

8.3 分页查询

如果一页记录为 10 条，希望查看第 3 页记录应该怎么查呢？

- 第一页记录起始行为 0，一共查询 10 行；
- 第二页记录起始行为 10，一共查询 10 行；
- 第三页记录起始行为 20，一共查询 10 行；

完整性约束

完整性约束是为了表的数据的正确性！如果数据不正确，那么一开始就不能添加到表中。

1 主键

当某一列添加了主键约束后，那么这一列的数据就不能重复出现。这样每行记录中其主键列的值就是这一行的唯一标识。例如学生的学号可以用来做唯一标识，而学生的姓名是不能做唯一标识的，因为学习有可能同名。

主键列的值不能为 NULL，也不能重复！

指定主键约束使用 PRIMARY KEY 关键字

- 创建表：定义列时指定主键：

```
CREATE TABLE stu(
    sid     CHAR(6) PRIMARY KEY,
    sname   VARCHAR(20),
    age     INT,
    gender  VARCHAR(10)
```

```
);
```

- 创建表：定义列之后独立指定主键：

```
CREATE TABLE stu(
    sid         CHAR(6),
    sname       VARCHAR(20),
    age         INT,
    gender      VARCHAR(10),
    PRIMARY KEY(sid)
);
```

- 修改表时指定主键：

```
ALTER TABLE stu
ADD PRIMARY KEY(sid);
```

- 删除主键（只是删除主键约束，而不会删除主键列）：

```
ALTER TABLE stu DROP PRIMARY KEY;
```

2 主键自增长

MySQL 提供了主键自动增长的功能！这样用户就不用再为是否有主键是否重复而烦恼了。当主键设置为自动增长后，在没有给出主键值时，主键的值会自动生成，而且是最大主键值+1，也就不会出现重复主键的可能了。

- 创建表时设置主键自增长（主键必须是整型才可以自增长）：

```
CREATE TABLE stu(
    sid INT PRIMARY KEY AUTO_INCREMENT,
    sname       VARCHAR(20),
    age         INT,
    gender      VARCHAR(10)
);
```

- 修改表时设置主键自增长：

```
ALTER TABLE stu CHANGE sid sid INT AUTO_INCREMENT;
```

- 修改表时删除主键自增长：

```
ALTER TABLE stu CHANGE sid sid INT;
```

3 非空

指定非空约束的列不能没有值，也就是说在插入记录时，对添加了非空约束的列一定要给值；在修改记录时，不能把非空列的值设置为 NULL。

- 指定非空约束：

```
CREATE TABLE stu(
    sid INT PRIMARY KEY AUTO_INCREMENT,
```

```
    sname VARCHAR(10) NOT NULL,
    age       INT,
    gender    VARCHAR(10)
);
```

当为 sname 字段指定为非空后，在向 stu 表中插入记录时，必须给 sname 字段指定值，否则会报错：

```
INSERT INTO stu(sid) VALUES(1);
```

插入的记录中 sname 没有指定值，所以会报错！

4 唯一

还可以为字段指定唯一约束！当为字段指定唯一约束后，那么字段的值必须是唯一的。这一点与主键相似！例如给 stu 表的 sname 字段指定唯一约束：

```
CREATE TABLE tab_ab(
    sid INT PRIMARY KEY AUTO_INCREMENT,
    sname VARCHAR(10) UNIQUE
);

INSERT INTO sname(sid, sname) VALUES(1001, 'zs');
INSERT INTO sname(sid, sname) VALUES(1002, 'zs');
```

当两次插入相同的名字时，MySQL 会报错！

5 外键

主外键是构成表与表关联的唯一途径！

外键是另一张表的主键！例如员工表与部门表之间就存在关联关系，其中员工表中的部门编号字段就是外键，是相对部门表的外键。

我们再来看 BBS 系统中：用户表（t_user）、分类表（t_section）、帖子表（t_topic）三者之间的关系。

例如在 t_section 表中 sid 为 1 的记录说明有一个分类叫 java，版主是 t_user 表中 uid 为 1 的用户，即 zs！

例如在 t_topic 表中 tid 为 2 的记录是名字为“Java 是咖啡”的帖子，它是 java 版块的帖子，它的作者是 ww。

外键就是用来约束这一列的值必须是另一张表的主键值！！！

- 创建 t_user 表，指定 uid 为主键列：

```
CREATE TABLE t_user(
    uid   INT PRIMARY KEY AUTO_INCREMENT,
    uname    VARCHAR(20) UNIQUE NOT NULL
);
```

- 创建 t_section 表，指定 sid 为主键列，u_id 为相对 t_user 表的 uid 列的外键：

```
CREATE TABLE t_section(
    sid   INT PRIMARY KEY AUTO_INCREMENT,
    sname    VARCHAR(30),
    u_id INT,
    CONSTRAINT fk_t_user FOREIGN KEY(u_id) REFERENCES t_user(uid)
);
```

- 修改 t_section 表，指定 u_id 为相对 t_user 表的 uid 列的外键：

```
ALTER TABLE t_section
ADD CONSTRAINT fk_t_user
FOREIGN KEY(u_id)
REFERENCES t_user(uid);
```

- 修改 t_section 表，删除 u_id 的外键约束：

```
ALTER TABLE t_section
DROP FOREIGN KEY fk_t_user;
```

6 表与表之间的关系

- 一对一：例如 t_person 表和 t_card 表，即人和身份证。这种情况需要找出主从关系，即谁是主表，谁是从表。人可以没有身份证，但身份证必须要有人才行，所以人是主表，而身份证是从表。设计从表可以有两种方案：
  - 在 t_card 表中添加外键列（相对 t_user 表），并且给外键添加唯一约束；
  - 给 t_card 表的主键添加外键约束（相对 t_user 表），即 t_card 表的主键也是外键。
- 一对多（多对一）：最为常见的就是一对多！一对多和多对一，这是从哪个角度去看得出来的。t_user 和 t_section 的关系，从 t_user 来看就是一对多，而从 t_section 的角度来看就是多对一！这种情况都是在多方创建外键！
- 多对多：例如 t_stu 和 t_teacher 表，即一个学生可以有多个老师，而一个老师也可以有多个学生。这种情况通常需要创建中间表来处理多对多关系。例如再创建一张表 t_stu_tea 表，给出两个外键，一个相对 t_stu 表的外键，另一个相对 t_teacher 表的外键。

编码

1 查看 MySQL 编码

SHOW VARIABLES LIKE 'char%';

```
mysql> show variables like 'char%';
+--------------------------+-------------
| Variable_name            | Value
+--------------------------+-------------
| character_set_client     | utf8
| character_set_connection | utf8
| character_set_database   | utf8
| character_set_filesystem | binary
| character_set_results    | utf8
| character_set_server     | utf8
| character_set_system     | utf8
| character_sets_dir       | D:\Program
+--------------------------+-------------
```

因为当初安装时指定了字符集为 UTF8，所以所有的编码都是 UTF8。

- character_set_client：你发送的数据必须与 client 指定的编码一致！！！服务器会使用该编码来解读客户端发送过来的数据；
- character_set_connection：通过该编码与 client 一致！该编码不会导致乱码！当执行的是查询语句时，客户端发送过来的数据会先转换成 connection 指定的编码。但只要客户端发送过来的数据与 client 指定的编码一致，那么转换就不会出现问题；
- character_set_database：数据库默认编码，在创建数据库时，如果没有指定编码，那么默认使用 database 编码；
- character_set_server：MySQL 服务器默认编码；
- character_set_results：响应的编码，即查询结果返回给客户端的编码。这说明客户端必须使

用 result 指定的编码来解码；

2 控制台编码

修改 character_set_client、character_set_results、character_set_connection 为 GBK，就不会出现乱码了。但其实只需要修改 character_set_client 和 character_set_results。

控制台的编码只能是 GBK，而不能修改为 UTF8，这就出现一个问题。客户端发送的数据是 GBK，而 character_set_client 为 UTF8，这就说明客户端数据到了服务器端后一定会出现乱码。既然不能修改控制台的编码，那么只能修改 character_set_client 为 GBK 了。

服务器发送给客户端的数据编码为 character_set_result，它如果是 UTF8，那么控制台使用 GBK 解码也一定会出现乱码。因为无法修改控制台编码，所以只能把 character_set_result 修改为 GBK。

- 修改 character_set_client 变量：**set character_set_client=gbk;**
- 修改 character_set_results 变量：**set character_set_results=gbk;**

设置编码只对当前连接有效，这说明每次登录 MySQL 提示符后都要去修改这两个编码，但可以通过修改配置文件来处理这一问题：配置文件路径：D:\Program Files\MySQL\MySQL Server 5.1\ my.ini

```
# CLIENT SECTION
# ----------
#
# The following options will k
# Note that only client applic
# to read this section. If you
# honor these values, you need
# MySQL client library initial
#
[client]

port=3306

[mysql]

default-character-set=gbk
```

认清楚这个，不要修改错了

这个修改是一劳永逸的

3 MySQL 工具

使用 MySQL 工具是不会出现乱码的，因为它们会每次连接时都修改 character_set_client、character_set_results、character_set_connection 的编码。这样对 my.ini 上的配置覆盖了，也就不会出现乱码了。

MySQL 数据库备份与还原

备份和恢复数据

1 生成 SQL 脚本

在控制台使用 mysqldump 命令可以用来生成指定数据库的脚本文本，但要注意，脚本文本中只包含数据库的内容，而不会存在创建数据库的语句！所以在恢复数据时，还需要自己手动创建一个数据库之后再去恢复数据。

mysqldump –u 用户名 –p 密码 数据库名>生成的脚本文件路径

```
C:\>mysqldump -uroot -p123 mydb1>C:\mydb1.sql

C:\>
```

现在可以在 C 盘下找到 mydb1.sql 文件了！

注意，mysqldump 命令是在 Windows 控制台下执行，无需登录 mysql！！！

2 执行 SQL 脚本

执行 SQL 脚本需要登录 mysql，然后进入指定数据库，才可以执行 SQL 脚本！！！

执行 SQL 脚本不只是用来恢复数据库，也可以在平时编写 SQL 脚本，然后使用执行 SQL 脚本来操作数据库！大家都知道，在黑屏下编写 SQL 语句时，就算发现了错误，可能也不能修改了。所以我建议大家使用脚本文件来编写 SQL 代码，然后执行之！

SOURCE C:\mydb1.sql

```
mysql> source c:\mydb1.sql
```

注意，在执行脚本时需要先行核查当前数据库中的表是否与脚本文件中的语句有冲突！例如在脚本文件中存在 create table a 的语句，而当前数据库中已经存在了 a 表，那么就会出错！

还可以通过下面的方式来执行脚本文件：

mysql -uroot -p123 mydb1<c:\mydb1.sql

mysql –u 用户名 –p 密码 数据库<要执行脚本文件路径

```
C:\>mysql -uroot -p123 mydb1<c:\mydb1.sql
```

这种方式无需登录 mysql！

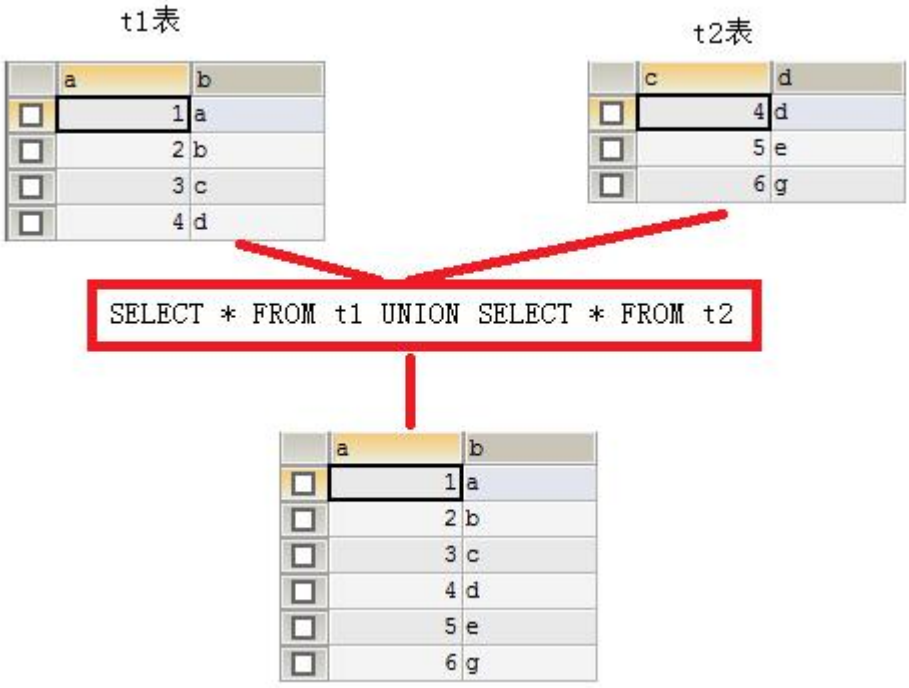
多表查询

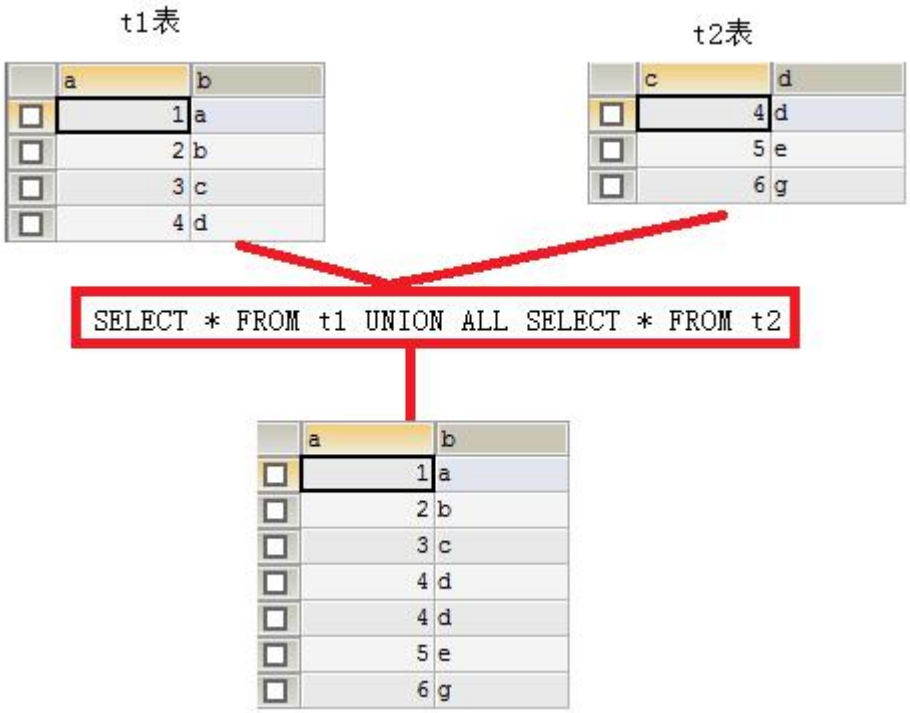
多表查询有如下几种：

- 合并结果集；
- 连接查询
  - 内连接
  - 外连接
    - 左外连接
    - 右外连接
    - 全外连接（MySQL 不支持）
  - 自然连接
- 子查询

1 合并结果集

1. 作用：合并结果集就是把两个 select 语句的查询结果合并到一起！
2. 合并结果集有两种方式：
   - UNION：去除重复记录，例如：SELECT * FROM t1 UNION SELECT * FROM t2；
   - UNION ALL：不去除重复记录，例如：SELECT * FROM t1 UNION ALL SELECT * FROM t2。

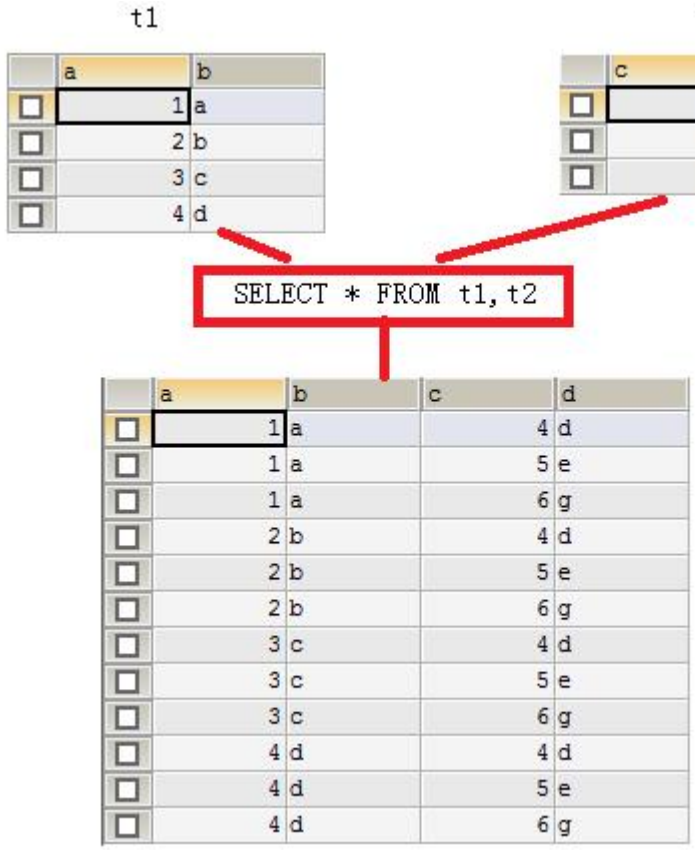




3. 要求：被合并的两个结果：列数、列类型必须相同。

2 连接查询

连接查询就是求出多个表的乘积，例如 t1 连接 t2，那么查询出的结果就是 t1*t2。



连接查询会产生笛卡尔积，假设集合 A={a,b}，集合 B={0,1,2}，则两个集合的笛卡尔积为{(a,0),(a,1),(a,2),(b,0),(b,1),(b,2)}。可以扩展到多个集合的情况。

那么多表查询产生这样的结果并不是我们想要的，那么怎么去除重复的，不想要的记录呢，当然是通过条件过滤。通常要查询的多个表之间都存在关联关系，那么就通过关联关系去除笛卡尔积。

你能想像到 emp 和 dept 表连接查询的结果么？emp 一共 14 行记录，dept 表一共 4 行记录，那么连接后查询出的结果是 56 行记录。

也就你只是想在查询 emp 表的同时，把每个员工的所在部门信息显示出来，那么就需要使用主外键来去除无用信息了。

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	10	教研部	北京
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	30	销售部	广州
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	40	财务部	武汉
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	10	教研部	北京
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	40	财务部	武汉
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	10	教研部	北京
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	20	学工部	上海
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	40	财务部	武汉
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	10	教研部	北京
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	30	销售部	广州
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	40	财务部	武汉
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	10	教研部	北京
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	20	学工部	上海
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	40	财务部	武汉
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	10	教研部	北京
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	20	学工部	上海

使用主外键关系做为条件来去除无用信息

```
SELECT * FROM emp,dept WHERE emp.deptno=dept.deptno;
```

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州

上面查询结果会把两张表的所有列都查询出来，也许你不需要那么多列，这时就可以指定要查询的列了。

```
SELECT emp.ename,emp.sal,emp.comm,dept.dname
FROM emp,dept
WHERE emp.deptno=dept.deptno;
```

	ename	sal	comm	dname
☐	张飞	24500.00	(NULL)	教研部
☐	曾阿牛	50000.00	(NULL)	教研部
☐	黄盖	13000.00	(NULL)	教研部
☐	甘宁	8000.00	(NULL)	学工部
☐	刘备	29750.00	(NULL)	学工部
☐	诸葛亮	30000.00	(NULL)	学工部
☐	周泰	11000.00	(NULL)	学工部
☐	庞统	30000.00	(NULL)	学工部
☐	黛绮丝	16000.00	3000.00	销售部
☐	殷天正	12500.00	5000.00	销售部
☐	谢逊	12500.00	14000.00	销售部
☐	关羽	28500.00	(NULL)	销售部
☐	韦一笑	15000.00	0.00	销售部
☐	程普	9500.00	(NULL)	销售部

还可以为表指定别名，然后在引用列时使用别名即可。

```
SELECT e.ename,e.sal,e.comm,d.dname
FROM emp AS e,dept AS d
WHERE e.deptno=d.deptno;
```

2.1 内连接

上面的连接语句就是内连接，但它不是 SQL 标准中的查询方式，可以理解为方言！SQL 标准的内连接为：

```
SELECT *
FROM emp e
INNER JOIN dept d
ON e.deptno=d.deptno;
```

内连接的特点：查询结果必须满足条件。例如我们向 emp 表中插入一条记录：

☐	1015	张三	保洁员	1009	1999-12-31	80000.00	20000.00	50

其中 deptno 为 50，而在 dept 表中只有 10、20、30、40 部门，那么上面的查询结果中就不会出现“张三”这条记录，因为它不能满足 e.deptno=d.deptno 这个条件。

2.2 外连接（左连接、右连接）

外连接的特点：查询出的结果存在不满足条件的可能。

左连接：

```
SELECT * FROM emp e
LEFT OUTER JOIN dept d
ON e.deptno=d.deptno;
```

左连接是先查询出左表（即以左表为主），然后查询右表，右表中满足条件的显示出来，不满足条件的显示 NULL。

这么说你可能不太明白，我们还是用上面的例子来说明。其中 emp 表中“张三”这条记录中，

部门编号为 50，而 dept 表中不存在部门编号为 50 的记录，所以“张三”这条记录，不能满足 e.deptno=d.deptno 这条件。但在左连接中，因为 emp 表是左表，所以左表中的记录都会查询出来，即“张三”这条记录也会查出，但相应的右表部分显示 NULL。

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1015	张三	保洁员	1009	1999-12-31	80000.00	20000.00	50	(NULL)	(NULL)	(NULL)

2.3 右连接

右连接就是先把右表中所有记录都查询出来，然后左表满足条件的显示，不满足显示 NULL。例如在 dept 表中的 40 部门并不存在员工，但在右连接中，如果 dept 表为右表，那么还是会查出 40 部门，但相应的员工信息为 NULL。

```
SELECT * FROM emp e
RIGHT OUTER JOIN dept d
ON e.deptno=d.deptno;
```

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	40	财务部	武汉

连接查询心得：

连接不限与两张表，连接查询也可以是三张、四张，甚至 N 张表的连接查询。通常连接查询不可能需要整个笛卡尔积，而只是需要其中一部分，那么这时就需要使用条件来去除不需要的记录。这个条件大多数情况下都是使用主外键关系去除。

两张表的连接查询一定有一个主外键关系，三张表的连接查询就一定有两个主外键关系，所以在大家不是很熟悉连接查询时，首先要学会去除无用笛卡尔积，那么就是用主外键关系作为条件来处理。如果两张表的查询，那么至少有一个主外键条件，三张表连接至少有两个主外键条件。

3 自然连接

大家也都知道，连接查询会产生无用笛卡尔积，我们通常使用主外键关系等式来去除它。而自然连接无需你去给出主外键等式，它会自动找到这一等式：

- 两张连接的表中名称和类型完成一致的列作为条件，例如 emp 和 dept 表都存在 deptno 列，并且类型一致，所以会被自然连接找到！

当然自然连接还有其他的查找条件的方式，但其他方式都可能存在问题！

```
SELECT * FROM emp NATURAL JOIN dept;
SELECT * FROM emp NATURAL LEFT JOIN dept;
SELECT * FROM emp NATURAL RIGHT JOIN dept;
```

4 子查询

子查询就是嵌套查询，即 SELECT 中包含 SELECT，如果一条语句中存在两个，或两个以上 SELECT，那么就是子查询语句了。

- 子查询出现的位置：
  - where 后，作为条件的一部分；
  - from 后，作为被查询的一条表；
- 当子查询出现在 where 后作为条件时，还可以使用如下关键字：
  - any
  - all
- 子查询结果集的形式：
  - 单行单列（用于条件）
  - 单行多列（用于条件）
  - 多行单列（用于条件）
  - 多行多列（用于表）

练习：

1. 工资高于甘宁的员工。

分析：

查询条件：工资>甘宁工资，其中甘宁工资需要一条子查询。

第一步：查询甘宁的工资

```
SELECT sal FROM emp WHERE ename='甘宁'
```

第二步：查询高于甘宁工资的员工

```
SELECT * FROM emp WHERE sal > (${第一步})
```

结果：

```
SELECT * FROM emp WHERE sal > (SELECT sal FROM emp WHERE ename='甘宁')
```

- 子查询作为条件
- 子查询形式为单行单列

2. 工资高于 30 部门所有人的员工信息

分析：
查询条件：工资高于 30 部门所有人工资，其中 30 部门所有人工资是子查询。高于所有需要使用 all 关键字。

第一步：查询 30 部门所有人工资

```
SELECT sal FROM emp WHERE deptno=30;
```

第二步：查询高于 30 部门所有人工资的员工信息

```
SELECT * FROM emp WHERE sal > ALL (${第一步})
```

结果：

```
SELECT * FROM emp WHERE sal > ALL (SELECT sal FROM emp WHERE deptno=30)
```

- 子查询作为条件
- 子查询形式为多行单列（当子查询结果集形式为多行单列时可以使用 ALL 或 ANY 关键字）

3. 查询工作和工资与殷天正完全相同的员工信息

分析：
查询条件：工作和工资与殷天正完全相同，这是子查询

第一步：查询出殷天正的工作和工资

```
SELECT job,sal FROM emp WHERE ename='殷天正'
```

第二步：查询出与殷天正工作和工资相同的人

```
SELECT * FROM emp WHERE (job,sal) IN (${第一步})
```

结果：

```
SELECT * FROM emp WHERE (job,sal) IN (SELECT job,sal FROM emp WHERE ename='殷天正')
```

- 子查询作为条件
- 子查询形式为单行多列

4. 查询员工编号为 1006 的员工名称、员工工资、部门名称、部门地址

分析：
查询列：员工名称、员工工资、部门名称、部门地址
查询表：emp 和 dept，分析得出，不需要外连接（外连接的特性：某一行（或某些行）记录上会出现一半有值，一半为 NULL 值）
条件：员工编号为 1006

第一步：去除多表，只查一张表，这里去除部门表，只查员工表

```
SELECT ename, sal FROM emp e WHERE empno=1006
```

第二步：让第一步与 dept 做内连接查询，添加主外键条件去除无用笛卡尔积

```
SELECT e.ename, e.sal, d.dname, d.loc
```

```
FROM emp e, dept d
WHERE e.deptno=d.deptno AND empno=1006
```

第二步中的 dept 表表示所有行所有列的一张完整的表，这里可以把 dept 替换成所有行，但只有 dname 和 loc 列的表，这需要子查询。

第三步：查询 dept 表中 dname 和 loc 两列，因为 deptno 会被作为条件，用来去除无用笛卡尔积，所以需要查询它。

```
SELECT dname,loc,deptno FROM dept;
```

第四步：替换第二步中的 dept

```
SELECT e.ename, e.sal, d.dname, d.loc
FROM emp e, (SELECT dname,loc,deptno FROM dept) d
WHERE e.deptno=d.deptno AND e.empno=1006
```

- 子查询作为表
- 子查询形式为多行多列