1. MySQL数据库

目录

[1 数据库概述 2](#_Toc533513240)

[1.1 什么是数据库? 2](#_Toc533513241)

[1.2 数据库的分类? 2](#_Toc533513242)

[1.3 什么是关系型数据库？ 3](#_Toc533513243)

[1.4 常见的关系型数据库 3](#_Toc533513244)

[2 MySQL建库、建表 3](#_Toc533513245)

[2.1 查看数据库、查看数据表 3](#_Toc533513246)

[2.2 创建数据库、创建数据表 4](#_Toc533513247)

[3 MySQL数据类型 4](#_Toc533513248)

[3.1 数据类型 - 数值类型 4](#_Toc533513249)

[3.2 数据类型 - 字符串类型 5](#_Toc533513250)

[3.3 数据类型 - 日期类型 6](#_Toc533513251)

[3.4 字段(列)约束 6](#_Toc533513252)

[4 SQL语言 7](#_Toc533513253)

[4.1 什么是SQL? 7](#_Toc533513254)

[5 更新表记录（insert/update/delete） 8](#_Toc533513255)

[5.1 insert--插入表记录 8](#_Toc533513256)

[5.2 update--修改表记录 9](#_Toc533513257)

[5.3 delete--删除表记录 9](#_Toc533513258)

[6 查询表记录（select） 9](#_Toc533513259)

[6.1 基础查询 10](#_Toc533513260)

[6.2 where子句查询 10](#_Toc533513261)

[6.3 排序查询 11](#_Toc533513262)

[6.4 分组查询 11](#_Toc533513263)

[6.5 聚合函数查询 12](#_Toc533513264)

[6.6 其他函数 13](#_Toc533513265)

[7 外键和表关系 14](#_Toc533513266)

[7.1 外键 14](#_Toc533513267)

[7.2 表关系 14](#_Toc533513268)

[7.2.1 一对多 14](#_Toc533513269)

[7.2.2 一对一 15](#_Toc533513270)

[7.2.3 多对多 16](#_Toc533513271)

[8 多表连接查询 17](#_Toc533513272)

[8.1 连接查询 17](#_Toc533513273)

[8.2 外连接查询 18](#_Toc533513274)

[8.2.1 左外连接查询 18](#_Toc533513275)

[8.2.2 右外连接查询 18](#_Toc533513276)

[8.3 子查询 19](#_Toc533513277)

[8.4 多表查询 20](#_Toc533513278)

[9 扩展内容 21](#_Toc533513279)

[9.1 使用CMD连接MySQL服务器 21](#_Toc533513280)

[9.2 使用Navicat连接MySQL服务器 22](#_Toc533513281)

# 数据库概述

## 什么是数据库

数据库：英文为Database，简称DB

数据库是按照数据结构来组织、存储和管理数据的仓库

简而言之，数据库就是存储数据的仓库。

## 数据库的分类

数据库根据存储采用的数据结构的不同可以分为许多种，其中包含早期的层次式数据库、网络式数据库。

目前占市场主流的是**关系型数据库**。当然还有**非关系(NoSQL)型数据库**（键值对数据库，例如：MongoDB、Redis）等其他类型的数据库

## 什么是关系型数据库？

底层是以二维表的及其之间的关系所组成的数据库，即是关系型数据库。例如：



## 常见的关系型数据库

SQL Server 微软提供（收费、Java中使用不多）

Oracle 甲骨文公司（收费、功能强大、性能优异，Java中使用者很多）

DB2 IBM（收费、中型/大型、银行/电信等企业）

MySQL 瑞典MySQL AB（免费开源、小型、性能也不差、适用于中小型项目、可集群）

SQLite 迷你数据库，嵌入式设备中

...

## 名称解释

1、数据库服务器

数据库服务器其实就是一个软件，比如我们安装的mysql软件（或者mariadb软件）

mysql服务器软件需要安装在服务器硬件上（就是一台计算机）才可以让外界来访问

2、数据库

在mysql服务器中，可以创建很多的数据库（database）

通常情况下，一个web站点对应一个数据库

3、数据表

在数据库（database）中，可以创建很多张表（table）

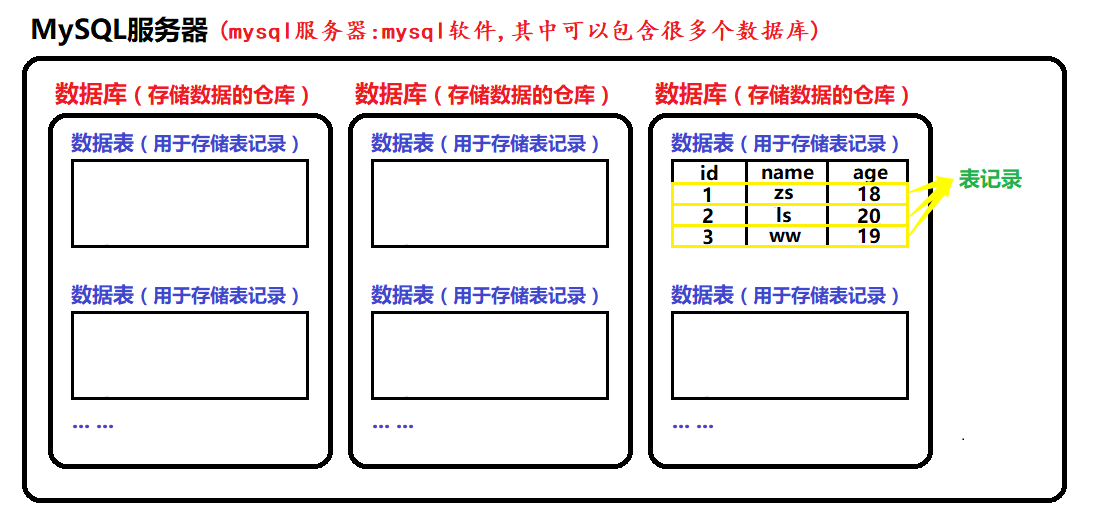
通常情况下，一张表用于保存一类数据，例如网站中的所有用户信息会保存在一张表中，所有商品信息会保存在另一张表中。

4、表记录

在数据表（table）中，可以插入很多条表记录

数据表往往保存一类数据，对应java中的一个类。

而一条表记录往往对应java中的一个具体的实例



# MySQL建库、建表

## 查看数据库、查看数据表

-- 1、查看mysql服务器中所有数据库

show databases;

-- 2、进入数据库

use test;

-- 3、查看当前库中的所有表

show tables;

提示: mysql数据库不区分大小写

## 创建数据库、创建数据表

**-- 1、创建数据库**

-- 删除mydb1库(如果存在)

drop database if exists mydb1;

-- 重新创建mydb1库

create database mydb1 charset utf8;

-- 查看、进入mydb1库

show databases;

use mydb1;

**-- 2、创建数据表**

-- 删除stu学生表(如果存在)

drop table if exists stu;

-- 创建stu学生表

create table stu(

id int, -- 学生编号

name varchar(20), -- 学生姓名

gender char(1), -- 学生性别

birthday date, -- 出生年月

score double -- 考试成绩

);

-- 查看表结构

desc stu;

# MySQL数据类型及字段约束

## 数值类型

MySQL中支持多种整型，其实很大程度上是相同的，只是存储值的大小范围不同而已。

**tinyint**：占用1个字节，相对于java中的byte

**smallint**：占用2个字节，相对于java中的short

**int**：占用4个字节，相对于java中的int

**bigint**：占用8个字节，相对于java中的long

其次是浮点类型即：float和double类型

**float**：4字节单精度浮点类型，相对于java中的float

**double**：8字节双精度浮点类型，相对于java中的double

## 字符串类型

**1、char(n)**

定长字符串，最长255个字符。n表示字符数，例如：

-- 创建user表, 指定用户名为char类型，字符长度不超过10

create table user(

username **char(10)**,

...

);

所谓的定长，是当插入的值长度小于指定的长度**10**（上面指定的）时, 剩余的空间会用空格填充。（这样会浪费空间）

**2、varchar(n)**

变长字符串，最长不超过 65535个字节，n表示字符数，

一般超过255个字节，会使用text类型，例如：

-- 创建user表, 指定用户名为varchar类型，长度不超过10

create table user(

username **varchar(10)**

);

所谓的不定长，是当插入的值长度小于指定的长度**10**（上面指定的）时, 剩余的空间可以留给别的数据使用。（节省空间）

**3、大文本（长文本）类型**

最长65535个字节，一般超过255个字符列的会使用text。

-- 创建user表,

create table user(

resume text

);

另，text也分多种，其中bigtext存储数据的长度约为4GB。

* **总结**： char、varchar、text都可以表示字符串类型，其区别在于：

(1)char在保存数 据时, 如果存入的字符串长度小于指定的长度n,后面会用空格补全。

(2)varchar和text保存数据时, 按数据的真实长度存储, 剩余的空间可以留给别的数据用.

(3)char会造成空间浪费(不足指定长度的会用空格补全), 但是由于不需要计算数据的长度, 因此速度更快。（即**以空间换时间**）

(4)varchar和text是节省了空间，但是存储的速度不如char快，因为要计算数据的实际长度。（即**以时间换空间**）

## 日期类型

**1、date**：年月日

**2、time**：时分秒

**3、datetim**e：年月日 时分秒

**4、timestamp**：时间戳(实际存储的是一个时间毫秒值)，与datetime存储日期格式相同。

timestamp最大表示2038年，而datetime范围是1000~9999

timestamp在插入数、修改数据时，可以自动更新成系统当前时间(后面用到时再做讲解)

## 字段(列)约束

创建表时, 除了要给每个列指定对应的数据类型, 有时也需要给列添加约束。常见的约束有：主键约束、唯一约束、非空约束、外键约束。

**1、主键(primary key)**

主键是数据表中，一行记录的唯一标识。比如学生的编号，人的身份证号, 例如:

创建user表,指定id为主键, 作为用户的唯一标识

create table stu(

id int **primary key**,

...

);

主键的特点： 唯一且不能为空！！！

当主键为数值时，为了方便维护，可以设置主键为自增（auto\_increment）

示例：创建user表,指定id为主键, 并指定主键自增

create table user(

id int primary key **auto\_increament**,

...

);

自增的作用是, 后期往表中插入记录时, 自增的列可以不用插入值(插入null即可), 数据库会自动插入值(1,2,3…), 例如：

insert into stu values(null, '王海涛','男', '1993-1-10');

insert into stu values(null, '刘沛霞','男', '1995-6-15');

提示：如果建表时没有指定主键自增, 也可以后期修改id为主键自增。

**2、唯一(unique)**

保证所约束的列必须是唯一的，即不能重复出现，例如：用户注册时，保存的用户名不可以重复。

示例：创建user表, 指定用户名不能重复

create table user(

id int primary key auto\_increament,

username varchar(50) **unique**,

...

);

唯一约束特点是不能重复！！(允许为空)

**3、非空(not null)**

保证所约束的列必须是不为空的，即在插入记录时，该列必须要赋值，例如：用户注册时，保存的密码不能为空。

创建user表, 指定密码不能为空

create table user(

id int primary key auto\_increament,

username varchar(50) unique,

password varchar(50) **not null**,

...

);

非空约束的特点是： 不能为空值， 即插入数据时该列必须得有值！！

**4、外键约束**

外键是用于表和表之间关系的列，后面在表关系中讲解

# SQL语言

## 什么是SQL语言?

**Structured Query Language：结构化的查询语言**

SQL是操作所有关系型数据库的通用的语言

SQL语言的分类：

a) DDL -- 数据库定义语言，指CREATE、ALTER、DROP等操作（ 即创建、删除、修改数据库和数据表 ）

b) DML -- 数据操作语言，指INSERT、UPDATE、DROP等操作（ 即数据表中数据的增、删、改操作 ）

c) **DQL** -- 数据查询语言（指SELECT操作，即数据表中数据的查询操作）

...

**其中DQL，也就是数据查询操作是 作！！也是我们学习的重点！！**

# 更新表记录（insert/update/delete）

## insert--插入表记录

插入记录的语法：

**insert** **into** 表名(列1,列2...) **values**(值1, 值2...);

-- insert用于向指定的表中插入哪些列, 以及给予哪些值

如果要为所有的列插入值, 可以省略表名后面的列列表，例如：

**insert** **into** 表名 **values**(值1, 值2...);

注意: (1)只能在给所有列插入值时, 才可以省略列列表, 否则会报错

(2)在插入值时, 如果没有省略列列表, 那么列列表和值列表要一一对应

(3)在插入值时, 如果省略了列列表, 那么值列表中值的顺序要和声明列的顺序一致

(4)在插入字符串和日期时, 字符串和日期要用单引号引起来。

-- 例如: 往学生表(stu)中插入记录

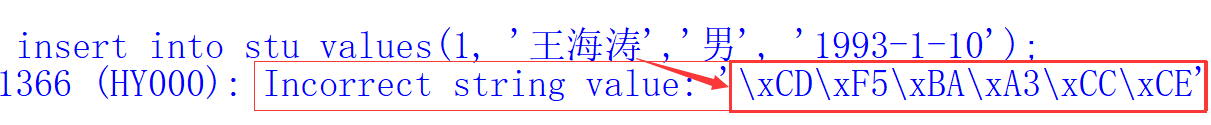
insert into stu(id,name,gender,birthday,score) values(1, '王海涛','男', '1993-1-10', 78);

insert into stu values(2, '齐雷','男', '1995-4-8', 78);

-- 或为指定列插入数据

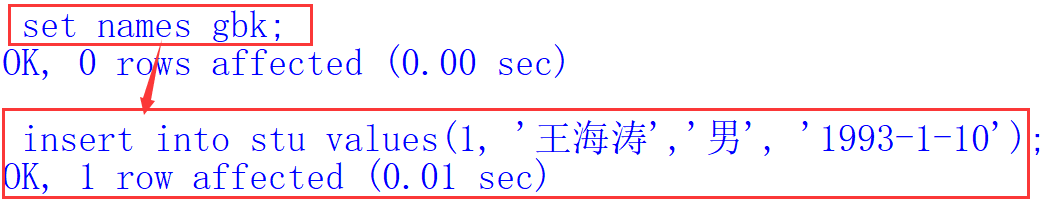
insert into stu(id, name, gender) values(3, '刘沛霞', '女');

**常见问题（FAQ）：在CMD中插入记录时如果报如下错误，**



因为这里存在中文的乱码问题。

解决方式：先设置编码GBK **"set names gbk"**, 再插入记录



因此，在通过**CMD**访问数据库时，最好一登录就设置 **"set names gbk"**，即可避免在插入数据时或者查询数据时数据乱码

## update--修改表记录

-- 修改stu表中王海涛的成绩，加10分特长分。

update stu set score=score+10 where name='王海涛';

-- 查询stu表中王海涛的信息

select \* from stu where name='王海涛';

若没有where子句, 则默认修改所有员工

-- 修改stu表中所有学生的成绩，加10分特长分。

update stu set score=score+10;

-- update stu set score=ifnull(score, 0)+10;

注意: null值和任何值计算结果还是null, 因此, 可以通过ifnull函数将null置为零对待.

## delete--删除表记录

-- 删除stu表中的王海涛信息

delete from stu where name='王海涛';

若没有where子句, 则默认删除所有记录

-- 删除emp表中的所有信息

delete from stu;

# 查询表记录（select）

\* 准备数据：执行《SQL脚本-db10库.txt》中的脚本，创建db10库，创建emp表，并往emp表中插入记录。

## 基础查询

-- 查询emp表中的所有员工，显示员工姓名、薪资、奖金

select name,sal,bonus from emp;

-- 查询emp表中的所有员工，显示所有列。

select \* from emp;

提示: distinct关键字，用于剔除指定列中的重复值，例如：

-- 查询emp表中的所有部门，剔除重复的部门

select distinct dept from emp;

试一试: 查询emp表中的所有奖金, 剔除重复的值

## where子句查询

对表中的所有记录进行筛选、过滤使用where子句

下面的运算符可以在 WHERE 子句中使用：



SQL练习：

-- 查询emp表中薪资大于3000的所有员工，显示员工姓名、薪资

select name,sal from emp **where** sal>3000;

-- 查询emp表中总薪资(薪资+奖金)大于3500的所有员工，显示员工姓名、总薪资

select name, sal+bonus from emp **where** (sal+bonus)>3000;

-- 或

select name, sal+bonus **as** 总薪资 from emp **where** (sal+bonus)>3000;

**提示：(1)as用于定义别名**(仅在查询的结果中作为列的表头显示)，也可以省略as

**(2)where子句中不能使用列别名**（但是可以使用表别名）

-- 查询emp表中薪资在3000和4500之间的员工，显示员工姓名和薪资

select name,sal from emp **where** sal between 3000 and 4500;

-- 查询emp表中姓名中以"刘"开头的员工，显示员工姓名。

select name,sal from emp where name like '刘%';

-- 查询emp表中姓名以"刘"开头，字数为两个字的员工，显示员工姓名。

select \* from emp where name like '刘\_';

-- 查询emp表中姓名中包含"涛"字的员工，显示员工姓名。

select \* from emp where name like '%涛%';

提示："%" 表示任意0或多个字符。"\_" 表示任意一个字符

-- 查询emp表中薪资为1400、1600、1800的员工，显示员工姓名和薪资

select name,sal from emp **where** sal in(1400,1600,1800);

-- 查询emp表中薪资小于2000和薪资大于4000的员工，显示员工姓名、薪资。

select name,sal from emp where sal<2000 or sal >4000;

-- 查询emp表中薪资大于3000并且奖金小于600的员工，显示姓名、薪资、奖金。

select name,sal,bonus from emp where sal>3000 and bonus<600;

## 排序查询

对查询的结果进行排序使用 order by关键字。

order by 排序的列 asc 升序 ↑

order by 排序的列 desc 降序 ↓

-- 对emp表中所有员工的薪资进行升序(从低到高)排序，显示姓名、薪资。

select name,sal from emp **order by** **asc**;

-- 对emp表中所有员工的总薪资进行降序(从高到低)排序，显示姓名、总薪资。

select name, sal+bonus as 总薪资 from emp order by (sal+bonus) **desc**;

## 分组查询

对所查询的记录可以根据某一列进行分组, 分组使用group by。

-- 将员工按照部门进行分组

select \* from emp group by dept;

-- 对emp表按照部门进行分组, 并统计每个部门的人数, 显示部门和对应人数

select dept 部门名称, count(\*) 部门人数 from emp group by dept;

-- 对emp表按照部门进行分组, 求每个部门的最高薪资(不包含奖金)

select max(sal) 总薪资 from emp **group by** dept;

注意：分组之前使用聚合函数表示对查询的所有记录进行统计计算

分组之后使用聚合函数表示对每一个组中的所有记录进行统计计算。

## 聚合函数查询

**1、max()或min()** -- 返回某列的最大值或最小值

**2、count()** -- 返回某列的行数

**3、sum()** -- 返回某列值之和

**4、avg()** -- 返回某列的平均值

-- 查询emp表中最高薪资

select **max**(sal) as 最高薪资 from emp;

-- 查询emp表中最高总薪资(薪资加奖金)

select **max**(sal+bonus) as 最高薪资 from emp;

-- 统计emp表中薪资大于3000的员工人数

select **count**(\*) from emp where sal>3000;

-- 统计emp表中所有员工的总薪资(不包含奖金)

select **sum**(sal) as 员工总薪资 from emp;

-- 统计emp表员工的平均薪资(不包含奖金)

select **avg**(sal) as 员工总薪资 from emp;

**!!重要提示：**

a) 可以使用count(\*)统计记录行数

b) 多个聚合函数可以一起查询

-- 例如:根据部门进行分组，统计每个部门员工人数和平均薪资

select dept, count(\*) 员工人数, avg(sal) 平均薪资 from emp group by dept;

c) 聚合函数不能用在where子句中

d) 在没有分组的情况下，聚合函数不能和其他普通字段一起查询

-- 例如: 查询emp表中薪资最高的员工姓名, 下面的写法是错的:

select name, max(sal) from emp;--结果是错的

-- 正确的查询：

select name, sal from emp where sal=(

select max(sal) from emp

);-- 子查询

## 其他函数

**1、数值函数**

(1)ceil(数值) -- 向上取整

(2)floor(数值) -- 向下取整

(3)round(数值) -- 向下取整

(4)rand(数值) -- 随机数

-- emp表中所有员工薪资上涨15.47%, 向上取整。

select name,sal, ceil(sal\*1.1547) from emp;

**2、日期函数**

(1)curdate() -- 返回当前日期(年月日)

(2)curtime() -- 返回当前时间(时分秒)

(3)now() -- 返回当前日期+时间(年月日 时分秒)

(4)date\_add()、date\_sub() -- 增加/减少日期

(5)year()、month()、day()、hour()、minute()、second()，分别用来获取日期中的年、月、日、时、分、秒

-- 查询系统当前时间。

select now();

-- 查询emp表中所有员工的年龄，显示姓名、年龄。

select name,year(curdate()) - year(birthday) 年龄 from emp;

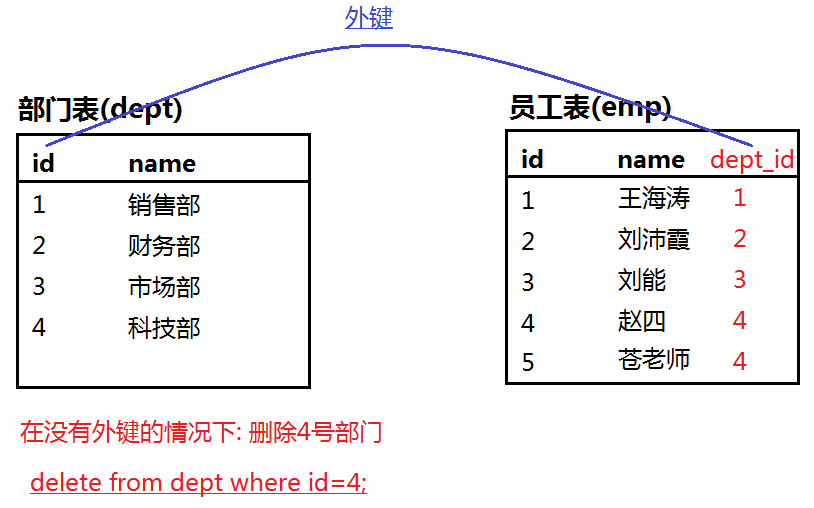
-- 查询emp表中所有在1993和1995年出生的，显示姓名、出生日期。

select name,birthday from emp where year(birthday) between 1993 and 1995;

# 外键和表关系

## 外键

外键：唯一标识其他表中的一条记录，用来通知数据库两张表列与列之间的对应关系, 并让数据库帮我们维护这样关系的键就叫做外键。



例如：员工表的部门id列（dept\_id）和部门表的id列具有一 一对应的关系, 其中dept\_id就是外键。

外键作用: 确保数据库数据的完整性和一致性

添加外键: 例如：foreign key(dept\_id) references dept(id)

\* 准备数据：执行《SQL脚本-db20库.txt》中的脚本，创建db20库，创建dept和emp表，并往两表中插入记录。

**练习：**

(1)执行上面的SQL语句，创建两张表（dept和emp），在创建时先不指定dept\_id为外键，尝试删除部门表中的某一个部门。（删除具有员工的部门会失败）

(2)将dept和emp表删除重建，在创建时指定dept\_id为外键，再次尝试删除部门表中的某一个部门。

## 表关系

### 一对多

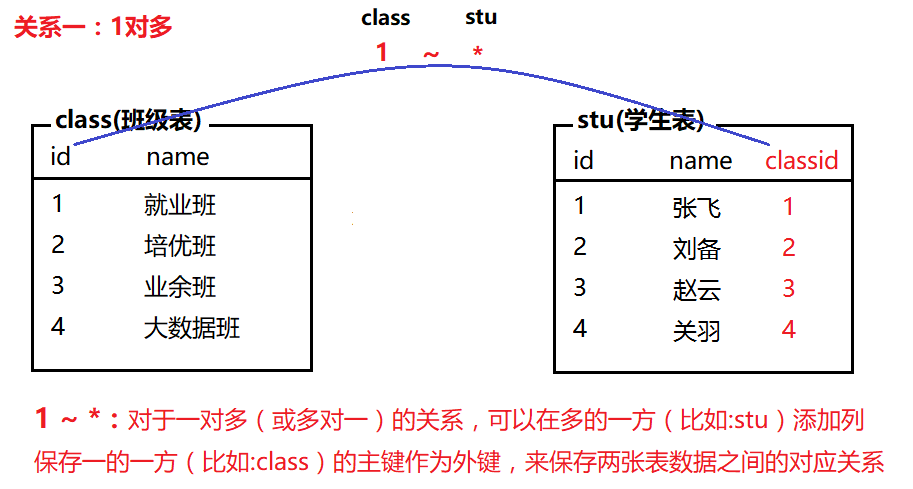
一对多，反过来就是多对一，以班级和学生为例：

(1)一个班级中可能会有多个学生(1~\*)

(2)一个学生只能属于一个班级(1~1)，两者合并结果还是1~\*

因此，班级表和学生表是一对多的关系

对于一对多的两张表，可以在多的一方添加列，保存一的一方的主键，从而保存两张表之间的关系



### 一对一

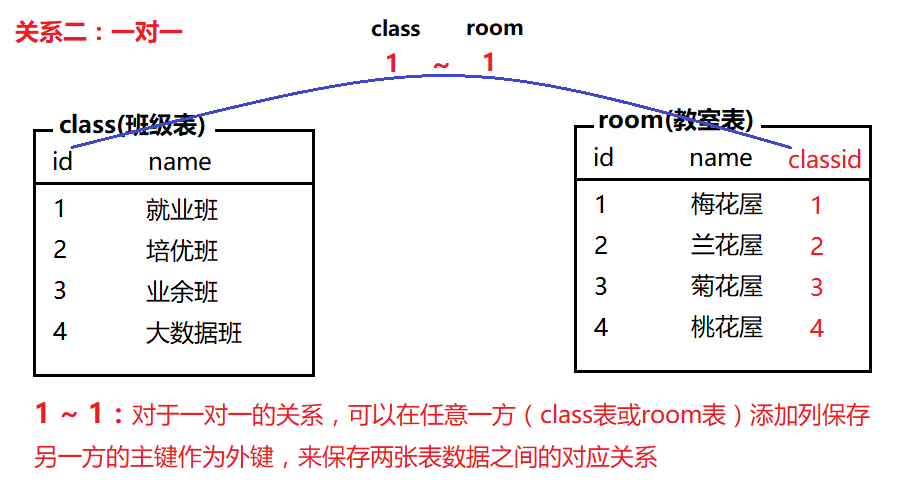
以班级和教室为例：

(1)一个班级对应一个教室(1~1)

(2)一个教室也只对应一个班级(1~1)，两者合并结果还是1~1

因此，班级表和教室表是一对一的关系

对于一对一关系的两张表，可以在任意一张表中添加列，保存另一张表的主键，从而保存两张表之间的关系



### 多对多

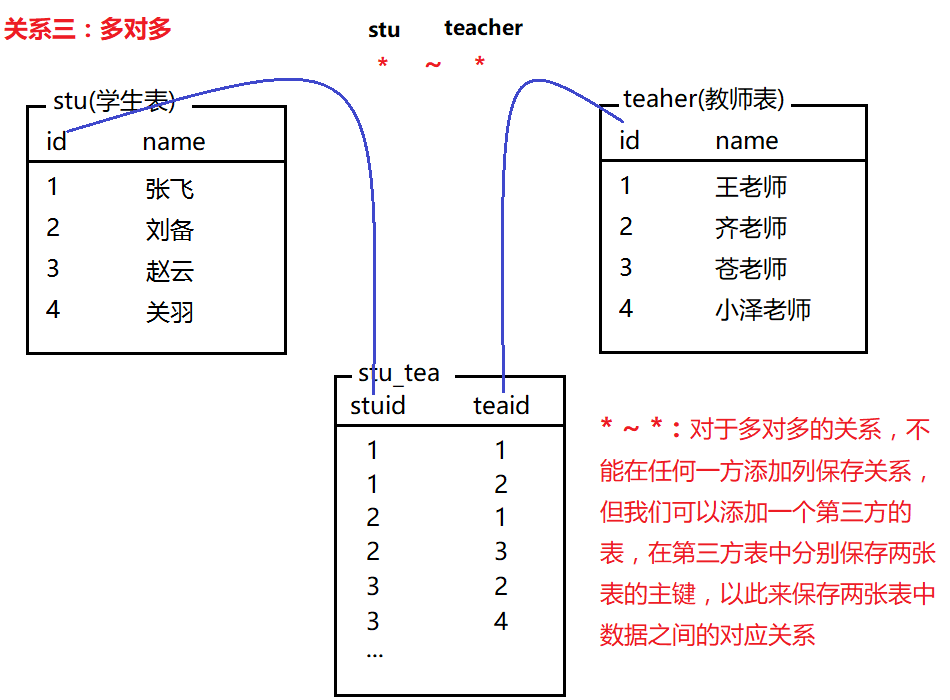
以学生和老师为例：

(1)一个学生对应多个老师(1~\*)

(2)一个老师也对应多个学生(1~\*)，两者合并结果是\*~\*

因此，学生表和老师表是多对多的关系

对于多对多的关系，可以拆分成两张一对多的关系，无法在两张表中添加列保存关系，但我们可以添加一张第三方的表（专门保存两张表的关系），保存两张表的主键，从而保存两张表的关系。

`

# 多表连接查询

## 连接查询

\* 准备数据：执行《SQL脚本-db30库.txt》中的脚本，创建db30库，创建dept和emp表，并往两表中插入记录。

连接查询：将两张或者两张以上的表，按照指定条件查询，将结果显示在一张表中。

多张表查询的语法：

select...

from **A, B...**

where...

如果表名过长，可以为表添加别名以方便书写

select...

from **A a, B b...**

where...

上面小写的a和b就是A和B表的别名:

-- 查询部门和员工两张表

select \*

from dept,emp;

上面查询的结果中存在大量错误的数据, 如果想正确显示部门及部门对应的员工，可以通过where子句从中筛选正确的数据.

-- 查询部门和部门下的员工。

select \*

from dept d,emp e

where d.id=e.dept\_id;

或

select \*

from dept d **inner** **join** emp e **on** d.id=e.dept\_id;

上面的查询(inner join...on...)方式也叫做内连接查询

## 外连接查询

### 左外连接查询

显示左侧表中的所有记录，如果在右侧表中没有对应的记录，则显示为null

语法：

select ...

from a **left** **join** b **on** (a.id=b.xid)

-- 查询所有部门和部门下的员工，如果部门下没有员工，显示null

select \*

from dept d left join emp e on d.id=e.dept\_id;

以上结果会显示（左侧表）所有部门，如果某部门下没有员工，（右侧表）则显示为null

### 右外连接查询

显示右侧表中的所有记录，如果在左侧表中没有对应的记录，则显示为null

语法：

select ...

from a **right** **join** b **on** (a.id=b.xid)

-- 查询部门和所有员工，如果员工没有所属部门，显示null

select \*

from dept d right join emp e on d.id=e.dept\_id;

以上结果会显示（右侧表）所有员工，如果员工没有所属部门，（左侧表）则显示为null

## 子查询

\* 准备数据：执行《SQL脚本-db40库.txt》中的脚本，创建db40库，创建dept和emp表，并往两表中插入记录

所谓的子查询，其实就是将一个查询得出的结果，作为另外一个查询的条件。

格式：

select...

from...

where...(select...from...)

1、列出薪资比'王海涛'高的所有员工，显示姓名、薪资

-- 先查询出'王海涛'的薪资

select sal from emp where name='王海涛';

-- 再查询比王海涛薪资(2450)高的员工

select name, sal

from emp

where sal>( select sal from emp where name='王海涛');

2、列出与'刘沛霞'从事相同职位的所有员工，显示姓名、职位、部门。

-- 先关联, 查询员工及员工对应的部门

select e.name, e.job, d.name from emp e, dept d where e.dept\_id=d.id;

-- 再查询'刘沛霞'的职位

select job from emp where name='刘沛霞';

-- 最后筛选, 筛选出和'刘沛霞'相同职位的员工

select e.name, e.job, d.name from emp e, dept d where e.dept\_id=d.id and job=(select job from emp where name='刘沛霞');

3、列出薪资高于在'大数据部'(已知部门编号为30)就职的所有员工的薪资的员工姓名和薪资、部门名称。

-- 查询出'大数据部'的最高薪资

select max(sal) from emp where dept\_id=30;

-- 关联查询, 查询员工的姓名,薪资, 部门名称

select e.name, e.sal, d.name from emp e, dept d where e.dept\_id=d.id and sal>(select max(sal) from emp where dept\_id=30);

## 多表查询

1、（左外连接）列出所有部门和部门下的员工，如果部门下没有员工, 显示为null。

select d.id, d.name, e.name, e.dept\_id

from dept d left join emp e on e.dept\_id=d.id;

2、（关联查询）列出在'培优部'任职的员工，假定不知道'培优部'的部门编号。

-- 先查询员工及员工所属部门

select e.name, e.dept\_id, d.id, d.name

from emp e, dept d

where e.dept\_id=d.id;

-- 再筛选过滤，查询部门名称为'培优部'的员工

select e.name, d.id, d.name

from emp e, dept d

where e.dept\_id=d.id and d.name='培优部';

3、（自连接查询）列出所有员工及其直接上级，显示员工姓名、上级编号，上级姓名

-- 先查询员工表（emp e1）

select name, topid from emp e1;

-- 再查询上级表（还是员工表，emp e2）

select id, name from emp e2;

-- 最后查询员工及其员工的直接上级

select e1.name,e1.topid, e2.id, e2.name

from emp e1, emp e2

where e1.topid = e2.id;

4、（分组、聚合函数）列出最低薪资大于1500的各种职位，显示职位和该职位最低薪资

-- 先查询出各种职位的最低薪资

select job, min(sal) 最低薪资

from emp group by job;

提示：对分组后的记录筛选过滤请使用having替换where，并且having书写在最后

-- 再查询出最低薪资>1500的职位

select job, min(sal) 最低薪资

from emp group by job

having min(sal)>1500;

5、（分组、聚合函数查询）列出在每个部门就职的员工数量、平均工资。显示部门编号、员工数量，平均薪资。

select dept\_id, count(\*) 员工数量, avg(sal) 平均薪资

from emp group by dept\_id;

6、（分组、关联、聚合函数查询）查出至少有一个员工的部门。显示部门编号、部门名称、部门位置、部门人数。

-- 先关联查询, 查询出员工和员工对应的部门

select d.id, d.name, d.loc, e.name

from emp e, dept d

where e.dept\_id=d.id;

-- 再根据部门进行分组, 统计每个部门的员工数量

select d.id, d.name, d.loc, count(\*) 员工数量

from emp e, dept d

where e.dept\_id=d.id group by e.dept\_id;

7、（自连接查询）列出受雇日期早于直接上级的所有员工的编号、姓名、部门名称。

-- 关联查询

select e1.id, e1.name, d.name

from emp e1, emp e2, dept d

where e1.dept\_id=d.id and e1.topid=e2.id and e1.hdate < e2.hdate;

# 扩展内容

## 使用CMD连接MySQL服务器

通过命令行工具可以登录MySQL客户端，连接MySQL服务器，从而访问服务器中的数据。

**1、登录mysql客户端命令**：**mysql -uroot -proot**

**-u：**后面的root是用户名，这里使用的是超级管理员root；

**-p：(小写的p)**后面的root是密码，这是在安装MySQL时就已经指定的密码；

**2、指定主机和端口连接**: mysql -uroot -proot **-h**127.0.0.1 **-P**3306

**-h：**后面给出的127.0.0.1是服务器主机名或ip地址，可以省略的，默认连接本机；

**-P：(大写的P)**后面的3306是连接端口，可以省略，默认连接3306端口；

**3、退出客户端命令：quit或exit或 \q**

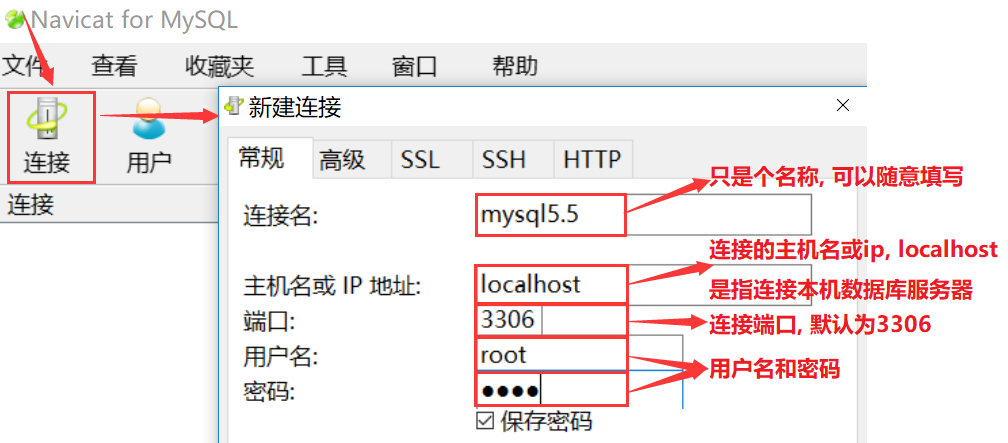
FAQ：常见问题:



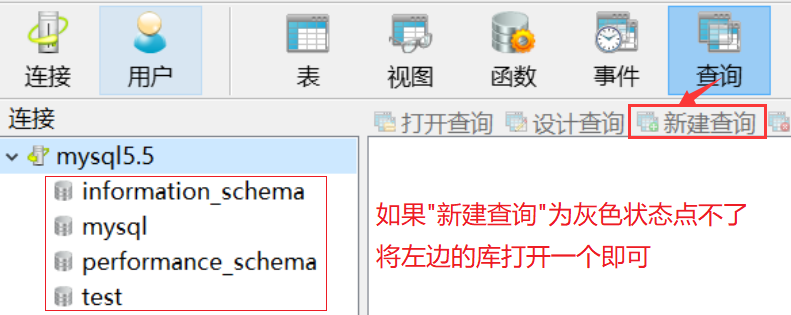
解决方法：复制mysql安装目录下的bin目录, 将bin目录的路径添加到path环境变量中!!

## 使用Navicat连接MySQL服务器

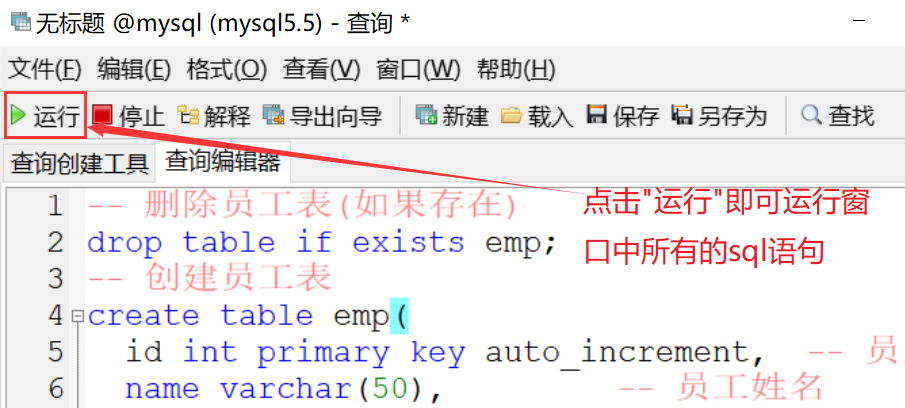
1、运行navicat客户端，按照下面的图示连接mysql服务器



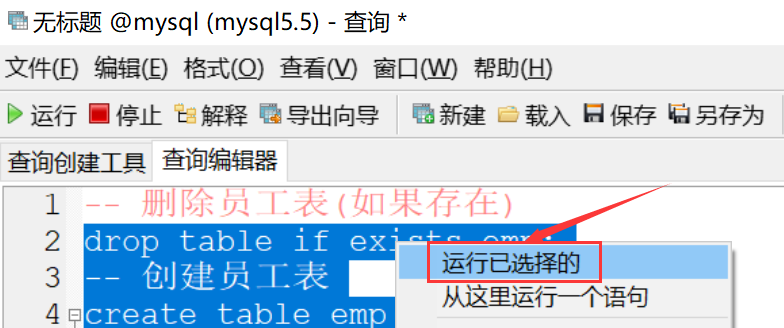
2、通过navicat发送SQL到mysql，打开所要操作的库（或任意打开一个库），再点击"查询" ---> "新建查询"



3、在弹出的窗口中，直接书写sql运行。



4、如果要运行指定的SQL，使用鼠标选中执行的SQL语句，右键-->"运行已选择的"



## 备份和恢复数据库

1、打开CMD窗口(不要登录)，通过命令备份mydb1数据库

备份命令: mysqldump -u用户名 -p 数据库名字 > 数据文件的位置

例如: mysqldump -uroot -p mydb1 > d:/mydb1.sql

输入密码, 如果没有提示错误, 即备份成功, 查询d盘的mydb1.sql文件

提示: (1)备份数据库的命令后面不要添加分号(;)

(2)备份数据库只是备份数据库中的表, 不会备份数据库本身

2、登录到MySQL客户端，删除mydb1库

-- 删除mydb1库

drop database mydb1;

-- 查询所有库, 是否还存在mydb1库

show databases;

由于备份时, 没有备份数据库本身, 所以在恢复库中的数据前, 需要先创建好要恢复的库

-- 创建mydb1数据库

create database mydb1 charset utf8;

提示: 此时的mydb1库是空的, 没有任何表

3、在CMD窗口中(不要登录)，通过命令恢复mydb1数据库

-- 在CMD窗口中(不要登录)

恢复命令: mysql -u用户名 -p 数据库名字 < 数据文件的位置

例如: mysql -uroot -p mydb1 < d:/mydb1.sql

输入密码, 如果没有提示错误, 即恢复成功, 下面进行验证

-- 在登录状态下, 选择mydb1库, 查询其中的表是否恢复了回来

use mydb1;

show tables;