

Mysql

- 启动与停止
 - services.msc 启动Mysql
 - net start/stop mysql80
- 客户端链接
 - 直接使用自带的
 - 命令: mysql [-h 127.0.0.1][-p 3306] -u root -p
- 关系型数据库 多张二维表组成的数据库

图形化界面

- datagrip

Sql

通用语法

- 单行或多行书写 分号结尾
- 空格/缩进来增强语句的可读性
- 不区分大小写 关键字建议使用大写
- 注释 -- # /**/
- DDL 定义语言 DML操作语言 DQL 查询语言 DCL控制语言

DDL操作数据库表 表中字段

数据库

- SHOW DATABASES ==查询所有数据库
- SELECT DATABASE (=) =查询当前数据库
- CREATE DATABASE 数据库名 创建数据库 推荐使用 utf8mb4字符集
- DROP DATABASE 数据库名 删除数据库
- USE 数据库名 == 进入/使用数据库

表的查看和创建

- show tables; 查看当前数据库所有表
- desc 表名 查询表结构
- show create table 表名 查询指定表的建表语句
- create table 表名() 创建表
 - 注意: 最后一个属性不要加逗号

- ```

1 create table tb_user(--comment 备注
2 -> id int comment '编号',
3 -> name varchar(50) comment '姓名',
4 -> age int comment '年龄',
5 -> gender varchar(1) comment '性别'
6 ->) comment '用户表';
7
Query OK, 0 rows affected (0.03 sec)

```

## 数值类型

| 分类   | 类型          | 大小      | 有符号(SIGNED)范围                                         | 无符号(UNSIGNED)范围                                          | 描述         |
|------|-------------|---------|-------------------------------------------------------|----------------------------------------------------------|------------|
| 数值类型 | TINYINT     | 1 byte  | (-128, 127)                                           | (0, 255)                                                 | 小整数值       |
|      | SMALLINT    | 2 bytes | (-32768, 32767)                                       | (0, 65535)                                               | 大整数值       |
|      | MEDIUMINT   | 3 bytes | (-8388608, 8388607)                                   | (0, 16777215)                                            | 大整数值       |
|      | INT或INTEGER | 4 bytes | (-2147483648, 2147483647)                             | (0, 4294967295)                                          | 大整数值       |
|      | BIGINT      | 8 bytes | (-2^63, 2^63-1)                                       | (0, 2^64-1)                                              | 极大整数值      |
|      | FLOAT       | 4 bytes | (-3.402823466 E+38, 3.402823466351 E+38)              | 0 和 (1.175494351 E-38, 3.402823466 E+38)                 | 单精度浮点数值    |
|      | DOUBLE      | 8 bytes | (-1.7976931348623157 E+308, 1.7976931348623157 E+308) | 0 和 (2.2250738585072014 E-308, 1.7976931348623157 E+308) | 双精度浮点数值    |
|      | DECIMAL     |         | 依赖于M(精度)和D(标度)的值                                      | 依赖于M(精度)和D(标度)的值                                         | 小数值(精确定点数) |

- double (长度, 小数位) 如100.0 设置为 double(4,1)
  - age 取值 int/tinyint unsigned

## 字符串

| 分类    | 类型         | 大小                    | 描述              |
|-------|------------|-----------------------|-----------------|
| 字符串类型 | CHAR       | 0-255 bytes           | 定长字符串           |
|       | VARCHAR    | 0-65535 bytes         | 变长字符串           |
|       | TINYBLOB   | 0-255 bytes           | 不超过255个字符的二进制数据 |
|       | TINYTEXT   | 0-255 bytes           | 短文本字符串          |
|       | BLOB       | 0-65 535 bytes        | 二进制形式的长文本数据     |
|       | TEXT       | 0-65 535 bytes        | 长文本数据           |
|       | MEDIUMBLOB | 0-16 777 215 bytes    | 二进制形式的中等长度文本数据  |
|       | MEDIUMTEXT | 0-16 777 215 bytes    | 中等长度文本数据        |
|       | LONGBLOB   | 0-4 294 967 295 bytes | 二进制形式的极大文本数据    |
|       | LONGTEXT   | 0-4 294 967 295 bytes | 极大文本数据          |

- 文本和二进制一般不用
  - char(最大长度): 为占用的字符 则用空格 代替 定长长度 在不管什么情况 占用空间一样时 使用
  - varchar(最大长度): 根据所存取的计算空间 变长长度

## 日期时间类型

| 分类   | 类型        | 大小 | 范围                                        | 格式                  | 描述            |
|------|-----------|----|-------------------------------------------|---------------------|---------------|
| 日期类型 | DATE      | 3  | 1000-01-01 至 9999-12-31                   | YYYY-MM-DD          | 日期值           |
|      | TIME      | 3  | -838:59:59 至 838:59:59                    | HH:MM:SS            | 时间值或持续时间      |
|      | YEAR      | 1  | 1901 至 2155                               | YYYY                | 年份值           |
|      | DATETIME  | 8  | 1000-01-01 00:00:00 至 9999-12-31 23:59:59 | YYYY-MM-DD HH:MM:SS | 混合日期和时间值      |
|      | TIMESTAMP | 4  | 1970-01-01 00:00:01 至 2038-01-19 03:14:07 | YYYY-MM-DD HH:MM:SS | 混合日期和时间值, 时间戳 |

- 常用 data 年月日 time 时/分/秒 datetime 年月日时分秒

## 案例

- 建表：编号数字工号/姓名字符串10位 性别男/女 年龄 不为负数 身份证 18位 入职 年月日

```

1 create table emp(
2 -> id int,
3 -> gh varchar(10),
4 -> name varchar(10),
5 -> sex char(1),
6 -> age tinyint unsigned,
7 -> sfz char(18),
8 -> rz date
9 ->);
10 --添加字段名
11 alter table emp add nickname varchar(20);
12 --修改字段的 数据类型
13 alter table emp modify age int;
14 --修改字段名和类型
15 alter table emp change sex gender char(1);
16 --删除字段名
17 alter table emp drop nickname;
18 --修改表名
19 alter table emp rename to emp1;
20 --删除表
21 drop table td_user;
22 --删除指定表名 并重新创建改表
23 truncate table td_user;

```

## 表的操作

- alter table 表名 **add** 字段名 类型长度 --添加字段
- alter table 表名 **modify** 字段名 新类型 --修改字段类型
- alter table 表名 **change** 原字段名 新字段名 类型 --修改字段名和类型
- alter table 表名 **drop** 字段名 --删除字段
- alter table 原表名 **rename to** 新表名 --修改表名
- **drop table 表名;** **删除表** 两种删除语法 数据都会被删除
- truncate table 表名 删除指定表名 并重新创建改表

## DML 数据操作语言

### 增加数据 insert

- insert into 表名(字段1, 字段2, ....) values(值1, 值2, ....); 给指定字段添加数据
- insert into 表名 values(值1, 值2, ....); 给所有字段添加数据
- **insert into 表名(字段1, 字段2, ....) values(值1, 值2, ....), (值1, 值2, ....), (值1, 值2, ....); 批量添加**
- **insert into 表名 values(值1, 值2, ....), (值1, 值2, ....), (值1, 值2, ....);**

### 修改数据 update

- **update 表名 set 字段名1=值1,字段名2=值2,... [where 条件];**
- update 表名 set 字段名1=值1,字段名2=值2; 修改所有

### 删除数据 delete

- `delete from 表名 [where 条件];` --删除一条
- `delete from user` --删除所有数据

## 案例

- 注意:
  - 指定字段顺序需要和值一一对应
  - 字符串和日期数据包含在引号里
  - 插入数据大小应在字段范围内
  - 修改或删除语句如果没有where 则修改或删除整张表的所有数据
  - delete语句不能删除某一字段的值(可以使用update)

```

1 --插入数据
2 insert into user(id, name, age) values (001,'李慧亮',10); --指定字段插入
3 insert into user values (002, 'scdad',16); --给所有字段插入
4 --批量
5 insert into user(id, name, age values (004, 'scdad',16),(003, 'scwdad',16);
6 insert into user values (005,'scdad',16),(006, 'scwdad',16);
7 --修改数据
8 update user set name='王五',age=18 where id=1; --将id为1的人名字 改为王五 年龄改为18
9 update user set age=19; --修改所有行
10 --删除
11 delete from user where id=1; --删除id为1的人
12 delete from user --删除所有数据

```

## DQL数据查询语言

### 编写顺序

- `select 字段列表 from 表名 where 条件 group by 分组字段 having分组后条件 order by 排序查询 limit 分页参数`

### 基本查询

- 查询多个字段 `select 字段1, 字段2, ... from 表名` 查询所有 `select * from 表名`
- 设置别名 `select 字段1[as别名], 字段2[as别名]... from 表名`
- 去重: `select distinct 字段列表 from 表名`

```

1 # --基本查询
2 -- 查询指定字段 name workno age
3 select name,workno,age from emp;
4 -- 查询所有
5 select * from emp;
6 -- 查询所有员工的工作地址,起别名
7 select workaddress as '工作地址' from emp;
8 select workaddress '工作地址' from emp;
9 -- 查询所有员工地址 不重复
10 select distinct workaddress as work from emp;

```

## 条件查询

- select 字段列表 from 表名 where 条件列表

- 

| 比较运算符               | 功能                      | 逻辑运算符    | 功能              |
|---------------------|-------------------------|----------|-----------------|
| >                   | 大于                      | AND 或 && | 并且 (多个条件同时成立)   |
| >=                  | 大于等于                    | OR 或     | 或者 (多个条件任意一个成立) |
| <                   | 小于                      | NOT 或 !  | 非 , 不是          |
| <=                  | 小于等于                    |          |                 |
| =                   | 等于                      |          |                 |
| <> 或 !=             | 不等于                     |          |                 |
| BETWEEN ... AND ... | 在某个范围之内(含最小、最大值)        |          |                 |
| IN(...)             | 在in之后的列表中的值, 多选一        |          |                 |
| LIKE 占位符            | 模糊匹配(_匹配单个字符, %匹配任意个字符) |          |                 |
| IS NULL             | 是NULL                   |          |                 |

- 

```
1 -- 条件查询
2 -- 年龄等于88
3 select * from emp where age=88;
4 -- 年龄 小于20
5 select * from emp where age<20;
6 -- 年龄 小于等于20
7 select * from emp where age<=20;
8 -- 没有省份证
9 select * from emp where idcard is null ;
10 -- 有身份证
11 select * from emp where idcard is not null ;
12 -- 年龄不等88
13 select * from emp where age!=88 ;
14 -- 15-20
15 select * from emp where age between 15 and 20;
16 -- 小于25 女
17 select * from emp where gender='女' and age<25 ;
18 -- 年龄 18 20 40
19 select * from emp where age=18 || age=20 || age=40;
20 select * from emp where age in (18,20,40);
21 -- 两个字 _单个字符 %任意个字符
22 select * from emp where name like '__';
23 -- 省份证最后为x
24 select * from emp where idcard like '%X';
25 select * from emp where idcard like '140%';
```

## 聚合函数

- 纵向计算 count 统计数量 max最大值 min 最小值 avg平均值 sum 求和
- select 聚合函数(字段列表) from 表名 不计算为null的值

- ```

1 -- 统计员工数量
2 select count(id) from emp;
3 select count(*) from emp;
4 -- 平均年龄
5 select avg(age) from emp;
6 -- 最大年龄
7 select max(age) from emp;
8 -- 最小年龄
9 select min(age) from emp;
10 -- 北京地区 所有员工之和
11 select sum(age) from emp where workaddress='北京';

```

分组查询

- select 字段列表 from 表名 where 条件 group by 基于的分组字段 having 分组过滤条件**
- where 分组之前过滤 不能对聚合函数进行判断(条件不可以写聚合函数)
- having 分组之后过滤 可以对聚合函数进行判断
- select 字段列表 from 表名 group by 分组字段名**

- ```

1 -- 性别 分组 统计员工 男和女的数量
2 select gender,COUNT(*) from emp group by gender;
3 -- 小于45 根据地址分组 获取员工数量大于3的工作地址
4 select workaddress,count(*) from emp where age <45 group by workaddress having
count(*)>3;

```

## 排序查询

- order by**
- select 字段列表 from 表名 order by 字段1 排序方式2, 字段1 排序方式2,**
- == asc 升序默认 desc 降序==**
- 如果是多字段排序, 第一个字段值相同时, 才会执行第二个字段**

- ```

1 -- 年龄升序
2 select * from emp order by age ;
3 -- 入职时间 降序
4 select * from emp order by entrydate desc ;
5 -- 年龄升序 如果年龄相同, 则入职时间降序
6 select * from emp order by age,entrydate desc ;
7 -- 分页查询
8 select * from emp limit 9,3;
9 -- 第一页 10
10 select * from emp limit 10;
11 -- 第二页 2-1 *10
12 select * from emp limit 10,10;

```

分页查询

- limit
- select 字段列表 from 表名 limit 起始索引, 查询记录;
- 从0开始 起始索引= (查询页码-1) *记录数
 - 如每页显示10跳 则查询第二页 = (2-1) *10 limit 10,10

案例

```

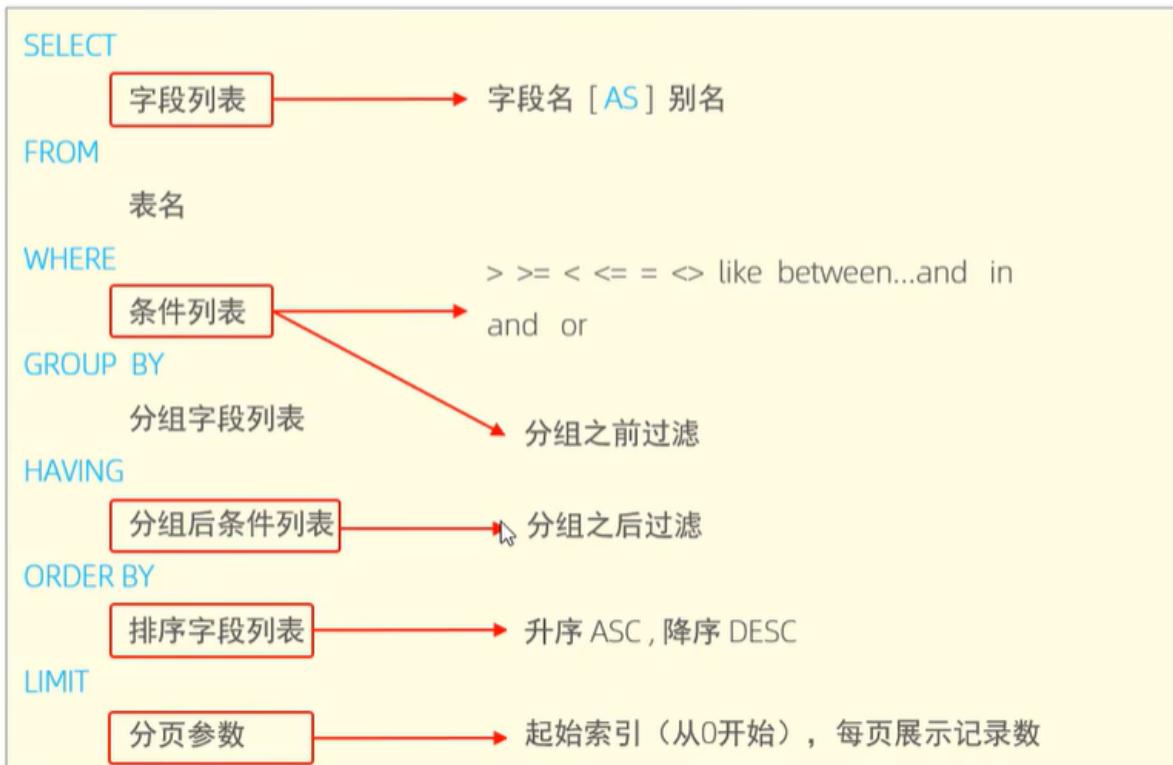
1 -- 查询 年龄 20 21 22 23员工信息
2 select * from emp where age in(20,21,22,23);
3 -- 性别 男 年龄20-40 名字 2个
4 select * from emp where gender='男' and age between 20 and 40 and name like '__';
5 -- 年龄小于 60 男性和女性的人数
6 select gender,count(*) from emp where age<60 group by gender;
7 -- 年龄 小于等于 35的姓名和年龄 对结果 年龄升序 相同则入职时间 降序
8 select age,name from emp where age<=35 order by age, entrydate desc ;
9 -- 性别为男 年龄20-40 前5个 年龄升序 入职升序
10 select * from emp where gender='男' and age between 20 and 40 order by age,entrydate limit
2;

```

执行顺序和编写

- from where group by select order by limit 执行
- select from where group by having order by limit 编写

1. DQL语句



DCL 控制数据语言

1. 用户管理

```
CREATE USER '用户名'@'主机名' IDENTIFIED BY '密码';  
ALTER USER '用户名'@'主机名' IDENTIFIED WITH mysql_native_password BY '密码';  
DROP USER '用户名'@'主机名'
```



2. 权限控制

```
GRANT 权限列表 ON 数据库名.表名 TO '用户名'@'主机名';  
REVOKE 权限列表 ON 数据库名.表名 FROM '用户名'@'主机名';
```

函数

- mysql内置函数

字符串函数

函数	功能
CONCAT(S1,S2,...Sn)	字符串拼接，将S1, S2, ... Sn拼接成一个字符串
LOWER(str)	将字符串str全部转为小写
• UPPER(str)	将字符串str全部转为大写
LPAD(str,n,pad) ↳	左填充，用字符串pad对str的左边进行填充，达到n个字符串长度
RPAD(str,n,pad)	右填充，用字符串pad对str的右边进行填充，达到n个字符串长度
TRIM(str)	去掉字符串头部和尾部的空格
SUBSTRING(str,start,len)	返回从字符串str从start位置起的len个长度的字符串

数值函数

函数	功能
CEIL(x)	向上取整
FLOOR(x) ↳	向下取整
MOD(x,y)	返回x/y的模
RAND()	返回0~1内的随机数
ROUND(x,y)	求参数x的四舍五入的值，保留y位小数

日期函数

函数	功能
CURDATE()	返回当前日期
CURTIME()	返回当前时间
NOW()	返回当前日期和时间
YEAR(date)	获取指定date的年份
MONTH(date)	获取指定date的月份
DAY(date)	获取指定date的日期
DATE_ADD(date, INTERVAL expr type)	返回一个日期/时间值加上一个时间间隔expr后的时间值
DATEDIFF(date1,date2)	返回起始时间date1 和 结束时间date2之间的天数

流程函数

- ifnull('123') '' ifnull(null,'123') '123'
- if(true,1,2) 1 if(false,1,2) 2

流程函数也是很常用的一类函数，可以在SQL语句中实现条件筛选，从而提高语句的效率。

函数	功能
IF(value , t , f)	如果value为true，则返回t，否则返回f
IFNULL(value1 , value2)	如果value1不为空，返回value1，否则返回value2
CASE WHEN [val1] THEN [res1] ... ELSE [default] END	如果val1为true，返回res1，... 否则返回default默认值
CASE [expr] WHEN [val1] THEN [res1] ... ELSE [default] END	如果expr的值等于val1，返回res1，... 否则返回default默认值

```

select
    id,
    name,
    (case when math >= 85 then '优秀' when math >=60 then '及格' else '不及格' end ) '数学',
    (case when english >= 85 then '优秀' when english >=60 then '及格' else '不及格' end ) '英语',
    (case when chinese >= 85 then '优秀' when chinese >=60 then '及格' else '不及格' end ) '语文'
from score;

```

```

1 -- 为工号补0
2 update emp set workno= lpad(workno,5,'0');
3 --生成六位验证码 由于第一位 0不显示 需要补0
4 select rpad(ceil(rand()*1000000),6,'0');
5 -- 查询 所有员工 入职天数(当前时间 距离 入职时间的天数) 并根据入职天数 进行降序
6 select name, datediff(curdate(),entrydate) as 'data' from emp order by data desc;
7 -- 北京显示 一线 阳泉显示 五线 其他 为工作地址
8 select
9     name,
10    (case workaddress when '北京' then '一线' when '阳泉' then '五线' else workaddress end)
11   '工作地址'
12  from emp;

```

约束

约束	描述	关键字
非空约束	限制该字段的数据不能为null	NOT NULL
唯一约束	保证该字段的所有数据都是唯一、不重复的	UNIQUE
主键约束	主键是一行数据的唯一标识，要求非空且唯一	PRIMARY KEY
默认约束	保存数据时，如果未指定该字段的值，则采用默认值	DEFAULT
检查约束(8.0.16版本之后)	保证字段值满足某一个条件	CHECK
外键约束	用来让两张表的数据之间建立连接，保证数据的一致性和完整性	FOREIGN KEY

约束是作用与表中字段上的

字段名	字段含义	字段类型	约束条件	约束关键字
id	ID唯一标识	int	主键，并且自动增长	PRIMARY KEY, AUTO_INCREMENT
name	姓名	varchar(10)	不能为空，并且唯一	NOT NULL, UNIQUE
age	年龄	int	大于0，并且小于等于120	CHECK
status	状态	char(1)	如果没有指定该值，默认为1	DEFAULT
gender	性别	char(1)	无	

```

1 create table users(
2     id int primary key auto_increment comment '编号',
3     name varchar(10) not null unique comment '姓名',
4     age int check (age>0&&age<20) comment '年龄',
5     status char(1) default 1 comment '状态',
6     gender char(1) comment '性别'
7 ) comment '用户表';

```

外键约束

- 让两张表的数据之间建立连接

- 关联表的数据 无法直接删除
- 添加外键 创建时 最后 constraint fk_xbuser_dept_id foreign key (dept_id) references dept(id)
- alter table 表名 add constraint 外键名称fk_表名_字段名 foreign key(字段名) references 那个表(那列);
- 删除外键
- alter table 表名 drop foreing key 外键名
-

WHERE							ORDER BY						
	id	name	age	job	salary	entrydate	managerid	dept_id					
1	1	金庸	55	总裁	20000	2000-01-01	<null>	4					
2	2	张三	30	开发	20000	2000-01-01	1	2					
3	3	李四	40	研发经理	20000	2000-01-01	1	1					
4	4	王五	35	市场	20000	2000-01-01	1	3					

empd

•

	id	name
1	1	研发部
2	2	开发部
3	3	市场部
4	4	总经办

dept

- ```

1 -- 添加外键
2 -- alter table 表名 add constraint 外键名称fk_表名_字段名 foreign key(字段名)
 references 那个表(那列);
3 alter table empd add constraint fk_empd_dept_id foreign key(dept_id) references
 dept(id);
4 -- 创建时 添加外键
5 create table xbuser(
6 id int auto_increment primary key ,
7 name varchar(50) not null ,
8 dept_id int comment '部门ID',
9 constraint fk_xbuser_dept_id foreign key (dept_id) references dept(id)
10);
11 insert into xbuser(id,name,dept_id) values (1,'李四'),(2,'王五',1);
12 -- 删除外键
13 alter table xbuser drop foreign key fk_xbuser_dept_id;

```

## 外键 删除/更新行为

- 加在最后 on update xxx on delete xxx

| 行为          | 说明                                                              |
|-------------|-----------------------------------------------------------------|
| NO ACTION   | 当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除/更新。(与 RESTRICT 一致)     |
| RESTRICT    | 当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除/更新。(与 NO ACTION 一致)    |
| CASCADE     | 当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有，则也删除/更新外键在子表中的记录。            |
| SET NULL    | 当在父表中删除对应记录时，首先检查该记录是否有对应外键，如果有则设置子表中该外键值为null(这就要求该外键允许取null)。 |
| SET DEFAULT | 父表有变更时，子表将外键列设置成一个默认的值(Innodb不支持)                               |

## 总结

1. 非空约束: **NOT NULL**
2. 唯一约束: **UNIQUE**
3. 主键约束: **PRIMARY KEY (自增: AUTO\_INCREMENT)**
4. 默认约束: **DEFAULT**
5. 检查约束: **CHECK**
6. 外键约束: **FOREIGN KEY**



## 多表查询

### 多表关系

- 多的一方建立外键，指向唯一的方的主键
-

## ● 一对多(多对一)

- 案例: 部门与员工的关系
- 关系: 一个部门对应多个员工, 一个员工对应一个部门
- 实现: 在多的一方建立外键, 指向一的一方的主键

| 员工表(emp) |     |      |
|----------|-----|------|
|          | id  | name |
| 1        | 张无忌 | 20   |
| 2        | 杨逍  | 33   |
| 3        | 赵敏  | 18   |
| 4        | 常遇春 | 43   |

| 部门表(dept) |     |      |
|-----------|-----|------|
|           | id  | name |
| 1         | 研发部 |      |
| 2         | 市场部 |      |
| 3         | 财务部 |      |
| 4         | 销售部 |      |

## ● 多对多

- 案例: 学生与课程的关系
- 关系: 一个学生可以选修多门课程, 一门课程也可以供多个学生选择
- 实现: 建立第三张中间表, 中间表至少包含两个外键, 分别关联两方主键

| 学生表(student) |     |            |
|--------------|-----|------------|
|              | id  | name       |
| 1            | 黛绮丝 | 2000100101 |
| 2            | 谢逊  | 2000100102 |
| 3            | 殷天正 | 2000100103 |
| 4            | 韦一笑 | 2000100104 |

| 学生课程关系表(student_course) |    |           |          |
|-------------------------|----|-----------|----------|
|                         | id | studentid | courseid |
| 1                       | 1  | 1         | 1        |
| 2                       | 1  | 1         | 2        |
| 3                       | 1  | 1         | 3        |
| 4                       | 2  | 2         | 1        |
| 5                       | 2  | 2         | 4        |

| 课程表(course) |        |      |
|-------------|--------|------|
|             | id     | name |
| 1           | Java   |      |
| 2           | PHP    |      |
| 3           | MySQL  |      |
| 4           | Hadoop |      |

## ● 一对一

- 案例: 用户与用户详情的关系
- 关系: 一对关系, 多用于单表拆分, 将一张表的基础字段放在一张表中, 其他详情字段放在另一张表中, 以提升操作效率
- 实现: 在任意一方加入外键, 关联另外一方的主键, 并且设置外键为唯一的(UNIQUE)

| 用户基本信息表(tb_user) |     |      |     |             |
|------------------|-----|------|-----|-------------|
|                  | id  | name | age | gender      |
| 1                | 黄渤  | 45   | 1   | 18800001111 |
| 2                | 冰冰  | 35   | 2   | 18800002222 |
| 3                | 码云  | 55   | 1   | 18800008888 |
| 4                | 李彦宏 | 50   | 1   | 18800009999 |

| 用户教育信息表(tb_user_edu) |    |        |         |               |              |            |
|----------------------|----|--------|---------|---------------|--------------|------------|
|                      | id | degree | major   | primaryschool | middleschool | university |
| 1                    | 本科 | 舞蹈     | 静安区第一小学 | 静安区第一中学       | 北京舞蹈学院       | 1          |
| 2                    | 硕士 | 表演     | 朝阳区第一小学 | 朝阳区第一中学       | 北京电影学院       | 2          |
| 3                    | 本科 | 英语     | 杭州市第一小学 | 杭州市第一中学       | 杭州师范大学       | 3          |
| 4                    | 本科 | 应用数学   | 阳泉第一小学  | 阳泉区第一中学       | 清华大学         | 4          |

## 多表查询

- select \* from 表一, 表二 where 当前.id=外键表.id; 会出去两个表中的数据的组合 加入条件消除笛卡尔积

## 连接查询

## ● 多表查询分类

### ➤ 连接查询

内连接：相当于查询A、B交集部分数据

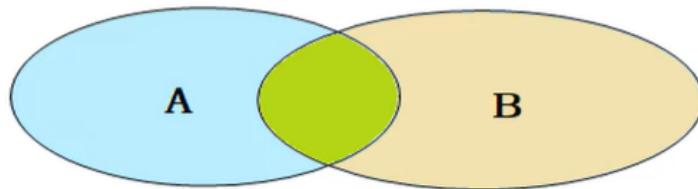
外连接：

左外连接：查询**左表**所有数据，以及两张表**交集部分数据**

右外连接：查询**右表**所有数据，以及两张表**交集部分数据**

自连接：当前表与自身的连接查询，**自连接必须使用表别名**

### ➤ 子查询



#### 内连接

- 绿色部分的数据
- 隐式内连接：select 字段列表 from 表1, 表2 where 条件...;

-- 内连接演示

I

-- 1. 查询每一个员工的姓名，及关联的部门的名称（隐式内连接实现）

-- 表结构: emp , dept

-- 连接条件: emp.dept\_id = dept.id

```
select emp.name , dept.name from emp , dept where emp.dept_id = dept.id
```

```
select e.name,d.name from emp e , dept d where e.dept_id = d.id;
```

- 显示: select 字段列表 from 表1 [inner] join 表2 on where 条件...;

```
-- 2. 查询每一个员工的姓名 , 及关联的部门的名称 (显式内连接实现) --- INNER JOIN ... ON ...
-- 表结构: emp , dept
-- 连接条件: emp.dept_id = dept.id
```

```
select e.name, d.name from emp e inner join dept d on e.dept_id = d.id;
```

```
select e.name, d.name from emp e join dept d on e.dept_id = d.id;
```

## 外连接

- 自己和绿色
- 左 select 字段列表 from 表1 left join 表2 on 条件...;
- 右 select 字段列表 from 表1 right join 表2 on 条件...;

### 左外表

```
select e.*, d.name from emp e left outer join dept d on e.dept_id = d.id;
```

```
select e.*, d.name from emp e left join dept d on e.dept_id = d.id;
```

```
-- 2. 查询dept表的所有数据, 和对应的员工信息(右外连接)
```

```
select d.*, e.* from emp e right outer join dept d on e.dept_id = d.id;
```

```
select d.*, e.* from dept d left outer join emp e on e.dept_id = d.id;
```

## 自连接

- select 字段列表 from 表1 别名1 join 表1 别名 表2 on 条件
- 可以是内连接查询 也可以时外连接查询 必须起别名

-- 自连接

-- 1. 查询员工 及其 所属领导的名字

-- 表结构: emp

```
select a.name , b.name from emp a , emp b where a.managerid = b.id;
```

-- 2. 查询所有员工 emp 及其领导的名字 emp , 如果员工没有领导, 也需要查询出来

-- 表结构: emp a , emp b

```
select a.name '员工' , b.name '领导' from emp a left join emp b on a.managerid = b.id;
```

## 联合查询

- select 字段列表 from 表a ... union all select 字段列表 from 表b ...;
- 若出现重复 将 all 省略

### 联合查询-union , union all

对于union查询，就是把多次查询的结果合并起来，形成一个新的查询结果集。

- ```
SELECT 字段列表 FROM 表A ...
UNION [ALL]
SELECT 字段列表 FROM 表B ....;
```

对于联合查询的多张表的列数必须保持一致，字段类型也需要保持一致。

子查询

- sql 嵌套select语句 嵌套语句 又称子查询
- 根据查询位置分为：where之后 from之后 select之后

子查询

- 概念：SQL语句中嵌套SELECT语句，称为**嵌套查询**，又称**子查询**。

```
SELECT * FROM t1 WHERE column1 = ( SELECT column1 FROM t2 );
```

子查询外部的语句可以是INSERT / UPDATE / DELETE / SELECT 的任何一个。

- 根据子查询结果不同，分为：

- 标量子查询（子查询结果为单个值）
- 列子查询(子查询结果为一列)
- 行子查询(子查询结果为一行)
- 表子查询(子查询结果为多行多列)

标量子查询

- 子查询返回结果为 单个值(数字、字符串、日期) 常见操作符 = <> > >= < <=
-

Output x ithema.empd x

|< < 1 row <> >>| ↻ | + - ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉

entrydate	
1	2010-01-01

|< < 1 row <> >>| ↻ | + -

id	
1	
	2

- ```

1 -- 子查询
2 -- 查询 开发部 所有员工信息 empd 员工表 dept 部门表 条件： 员工对应的部门中的开发部
3 -- a.查询开发部 部门 id
4 select id from dept where name='开发部';
5 -- 查询 部门id 对应的员工的所有信息
6 select * from empd where dept_id=(select id from dept where name ='开发部');
7 -- 查询 ‘李四’入职之后的员工信息
8 select * from empd where entrydate>(select entrydate from empd where name='李四')

```

## 例子查询

- 子查询返回一列 可以是多行 in not in any 任意满足一个 some all 全部满足

Output x ithema.empd x

|< < 2 rows <> >>| ↻ | + - ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉

| salary |       |
|--------|-------|
| 1      | 20000 |
| 2      | 26000 |

- ```

1 -- 查询 开发部 和 研发部的 所有员工信息
2 select * from empd where dept_id in(select id from dept where name='开发部'or name = '研发部');
3 -- 查询 比开发部所有人工资高的员工
4 select * from empd where salary > all (select salary from empd where dept_id =(select
id from dept where name='开发部')));
5 -- 查询 比开发部任意一人工资高的员工
6 select * from empd where salary > any (select salary from empd where dept_id =(select
id from dept where name='开发部')));
7 select * from empd where salary > some (select salary from empd where dept_id =(select
id from dept where name='开发部')));

```

行子查询

- 子查询返回一行 多列 = <>、 in 、 not in
-

	salary	managerid
1	20000	1

- ```

1 -- 查询与张三 薪资和直属领导 相同的
2 select * from empd where (salary,managerid) = (select salary,managerid from empd where
name='张三');

```

## 表子查询

- 子查询返回多行多列 IN from 之后

```

-- 表子查询
-- 1. 查询与 "鹿杖客" , "宋远桥" 的职位和薪资相同的员工信息
-- a. 查询 "鹿杖客" , "宋远桥" 的职位和薪资
-- b. 查询与 "鹿杖客" , "宋远桥" 的职位和薪资相同的员工信息
-- 2. 查询入职日期是 "2006-01-01" 之后的员工信息 , 及其部门信息
-- a. 入职日期是 "2006-01-01" 之后的员工信息
-- b. 查询这部分员工, 对应的部门信息;

```

1. 

```
select job, salary from emp where name = '鹿杖客' or name = '宋远桥';
```
2. 

```
select * from emp where (job,salary) in (select job, salary from emp where name = '鹿杖客' or name = '宋远桥');
```
3. 

```
select * from emp where entrydate > '2006-01-01';
```
4. 

```
select e.*, d.* from (select * from emp where entrydate > '2006-01-01') e left join dept d on e.dept_id = d.id ;
```