

### 1. 数据库概述

数据库一个存储数据的仓库。

数据库类型：层次式数据库、网络式数据库、关系型数据库。非关系型数据库

### 2. 关系型数据库

商业：

oracle

SQL Server

DB2

开源：

mysql

SQLite

### 3. MySql数据的安装

参照文档 --- 课前资料中图片

安装的路径不要有中文和空格

默认的端口3306不要去改, 保持默认即可

使用命令行窗口连接MYSQL数据库：mysql -u用户名 -p密码

登陆或退出MySql客户端命令

登录：**mysql -u root -p**

**回车之后写上密码：root**

-u：后面的root是用户名，这里使用的是超级管理员root；

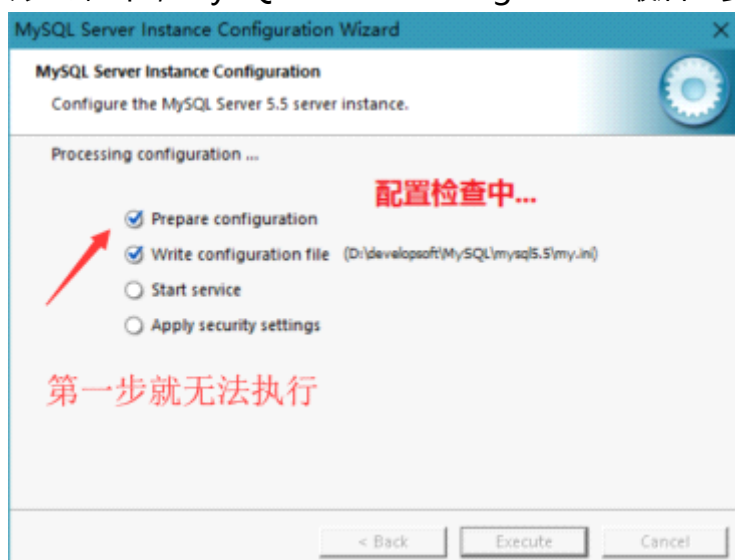
-p：后面的123是密码，这是在安装MySQL时就已经指定的密码；

-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；

退出：quit或exit；

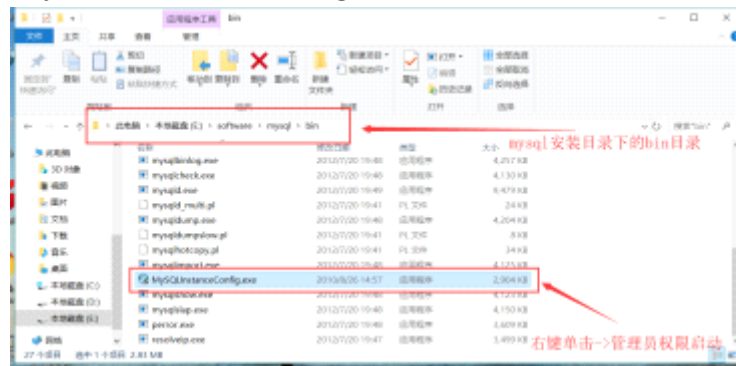
安装过程中出现的问题：

i. 在安装过程中，MySQL Server Configuration最后一步无法执行。



□ 解决方案：

结束当前界面，找到Mysql安装目录下的bin目录，右键单击MySQLInstanceConfig.exe文件->管理员权限执行。



ii. 10061错误：

报 “Can't connect to MySQL server on 'localhost' (10061) ”错误

解决：

在DOS下进入BIN目录

C:\Program Files\MySQL\MySQL Server 5.4\bin

然后，直接输入net start mysql

然后enter就可以了。

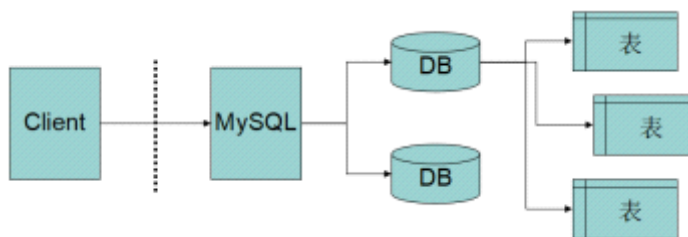
c:/programdata

#### 4. MySQL数据库服务器、数据库和表的关系

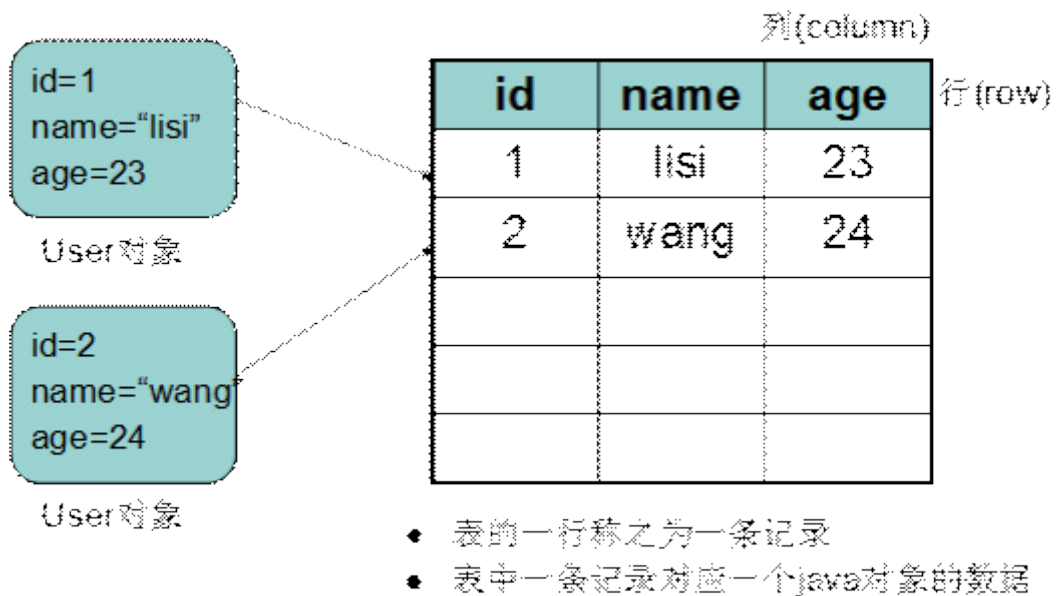
所谓安装数据库服务器，只是在机器上装了一个数据库管理程序，这个管理程序可以管理多个数据库，一般开发人员会针对每一个应用创建一个数据库。

为保存应用中实体的数据，一般会在数据库创建多个表，以保存程序中实体的数据。

数据库服务器、数据库和表的关系如图所示：



#### 5. 数据在数据库中的存储方式



列也称之为字段或者域。

## 6. SQL语言

- Structured Query Language, 结构化查询语言
- 非过程性语言
- 美国国家标准局(ANSI ) 与国际标准化组织 ( ISO ) 已经制定了SQL标准
- 为加强SQL的语言能力，各厂商增强了过程性语言的特征
  - 如Oracle的PL/SQL 过程性处理能力
  - SQL Server、Sybase的T-SQL
- SQL是用来存取关系数据库的语言，具有查询、操纵、定义和控制关系型数据库的四方面功能。

## 7.

## 1. 创建数据库

```
CREATE DATABASE [IF NOT EXISTS] db_name
[create_specification[,create_specification] ...]
create_specification:
    [DEFAULT] CHARACTER SET charset_name | [DEFAULT] COLLATE
collation_name
```

CHARACTER SET：指定数据库采用的字符集

COLLATE：指定数据库字符集的比较方式

( 查看mysql存储位置 :show global variables like "%datadir%"; )

练习：

创建一个名称为mydb1的数据库。

```
create database mydb1;
```

创建一个使用utf8字符集的mydb2数据库。

```
create database mydb2 character set gbk;
```

创建一个使用utf8字符集，并带校对规则的mydb3数据库。

```
create database mydb3 character set utf8 collate utf8_bin
```

## 2. 查看、删除数据库

显示数据库语句：

```
1 SHOW DATABASES
```

显示数据库创建语句：

```
1 SHOW CREATE DATABASE db_name
```

数据库删除语句：

```
1 DROP DATABASE [IF EXISTS] db_name
```

练习：

查看当前数据库服务器中的所有数据库 show databases;

查看前面创建的mydb2数据库的定义信息show create database mydb2;

删除前面创建的mydb1数据库 drop database mydb1;

## 3. 修改数据库

```
1 ALTER DATABASE [IF NOT EXISTS] db_name
2 [alter_specification [, alter_specification] ...]
3 alter_specification:
4     [DEFAULT] CHARACTER SET charset_name | [DEFAULT]
5     COLLATE collation_name
```

## 4. 选择数据库

```
1 use db_name;
```

查看当前使用的数据库:

```
1 select database();
```

- 练习

查看服务器中的数据库，并把其中某一个库的字符集修改为utf8;

```
alter database mydb3 character set gbk;
```

## 5. 创建表(基本语句)

```
1 CREATE TABLE table_name
2 (
3     field1    datatype,
4     field2    datatype,
5     field3    datatype
6 ) [character set 字符集] [collate 校对规则]
```

character set 字符集 collate 校对规则

field : 指定列名 datatype : 指定列类型

- 注意：创建表时，要根据需保存的数据创建相应的列，并根据数据的类型定义相应的列类型。例：user对象

id	int
name	string
password	string
birthday	date

## 6. MySQL常用数据类型

- 字符串型
  - VARCHAR、CHAR name varchar(20)存储的数据内容长度的可变的。  
name char(20)存储的数据内容长度是固定的。  
查询效率上char比varchar稍高一些。因为char类型每次读取固定长度，而varchar需要先判断数据的长度然后再读取。  
varchar(255);
- 大数据类型
  - BLOB、TEXT
- 数值型
  - TINYINT、SMALLINT、INT、BIGINT、FLOAT、DOUBLE
- 逻辑型
  - BIT
- 日期型
  - DATE、TIME、DATETIME、TIMESTAMP

## 7. 创建表练习

创建一个员工表employee ---- 查看表结构: desc 表名;

字段	属性
id	整形
name	字符型

gender	字符型
birthday	日期型
entry_date	日期型
job	字符型
salary	小数型
resume	大文本型

**\*创建一个员工表employee**

```
create table employee(
    id int primary key auto_increment ,
    name varchar(20),
    gender varchar(2) ,
    birthday date,
    entry_date date,
    job varchar(20),
    salary double,
    resume text
);
```

- 设置主键可以提升查询效率，依赖主键可以利用其身上的索引进行查询，利用索引查询速度较快。
- 一般情况情况下一张表都会都有一个主键，且这个主键一般都是int类型。
- 如果一个主键设置上auto\_increment属性则这个字段一定会是一个主键字段，且这个字段的数据不需要数据库管理员来维护，这个字段的值会自动添加，并增长。

创建完毕之后利用desc employee 来查看表结构。

## 8. 定义单表字段的约束

- 定义主键约束
  - primary key:不允许为空，不允许重复
  - 删除主键：alter table tablename drop primary key ;  
如果主键包含自动增长属性，需要先将自动增长去掉，再删除主键。
  - 主键自动增长：auto\_increment
- 定义唯一约束
  - unique
  - 例如：name varchar(20) unique
- 定义非空约束
  - not null
  - 例如：salary double not null
- 外键约束

## 9. 查看表信息

查看表结构：

```
desc tableName
```

查看当前所有表：

```
show tables
```

查看当前数据库表建表语句

```
show create table tabName;
```

## 10. 修改表


使用 ALTER TABLE 语句追加, 修改, 或删除列的语法.

```
ALTER TABLE table_name ADD column_name datatype
[DEFAULT expr] [, column_name datatype]..;
ALTER TABLE table_name MODIFY column_name datatype
[DEFAULT expr] [, column datatype]...;
ALTER TABLE table_name DROP column_name;
```

修改表的名称：

```
rename table 表名 to 新表名;
```

修改列的名称：

```
 ALTER TABLE table change old_column new_column typefiled;
```

修改表的字符集：

```
alter table user character set utf8;
```

### a. 练习

在上面员工表的基本上增加一个image列。

修改job列，使其长度为60。

删除gender列。

表名改为user。

修改表的字符集为utf8

列名name修改为username

## 11. 删除表

```
drop table tabName;
```

## 1. 数据库表记录CRUD语句

- Insert语句 (增加数据)
- Update语句 (更新数据)
- Delete语句 (删除数据)
- Select语句(查找数据)

## 2. Insert语句

### a. 使用 INSERT 语句向表中插入数据。

```
INSERT INTO table_name [(column [, column...])]
VALUES (value [, value...]);
```

- 插入的数据应与字段的数据类型相同。
- 数据的大小应在列的规定范围内，例如：不能将一个长度为80的字符串加入到长度为40的列中。
- 在values中列出的数据位置必须与被加入的列的排列位置相对应。
- 字符和日期型数据应包含在单引号中。
- 插入空值：不指定或insert into table value(null)
- 如果要插入所有字段可以省写列列表，直接按表中字段顺序写值列表

### • Insert语句练习

练习：使用insert语句向employee表中插入三个员工的信息。

字段名	字段类型
id	整形
name	字符串型
gender	字符串型
birthday	日期型
salary	浮点型
entry_date	日期型
resume	大文本型



### Tip：mysql中文乱码

- mysql有六处使用了字符集，分别为：  
client、connection、database、results、server、system。
- **client**是客户端使用的字符集。



- **connection**是连接数据库的字符集设置类型，如果程序没有指明连接数据库使用的字符集类型就按照服务器端默认的字符集设置。
- **database**是数据库服务器中某个库使用的字符集设定，如果建库时没有指明，将使用服务器安装时指定的字符集设置。
- **results**是数据库给客户端返回时使用的字符集设定，如果没有指明，使用服务器默认的字符集。
- **server**是服务器安装时指定的默认字符集设定。
- **system**是数据库系统使用的字符集设定。（utf-8不可修改）  
**show variables like 'character%';**  
 set names gbk;临时修改当前CMD窗口和mysql的通信编码字符集
- 通过修改my.ini 修改字符集编码  
 请到mysql安装目录下面找到 my.ini文件  
 修改default-character-set=utf8 为 default-character-set=gbk  
 有两个地方都要改  
 修改文件前，先停止mysql服务，等修改后再重新启动  
 使用dos命令：net stop mysql 来停止服务 net start mysql 来启动

### 3. Update语句

使用 update语句修改表中数据。

```
UPDATE      tbl_name
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]
```

UPDATE语法可以用新值更新原有表行中的各列。

SET子句指示要修改哪些列和要给予哪些值。

WHERE子句指定应更新哪些行。如没有WHERE子句，则更新所有的行。

#### Update语句练习

- 练习：在上面创建的employee表中修改表中的纪录。

要求:

将所有员工薪水修改为5000元。

将姓名为'张三丰'的员工薪水修改为3000元。

将姓名为'lisi'的员工薪水修改为4000元,job改为ccc。

将'wu'的薪水在原有基础上增加1000元。

### 4. Delete语句

使用 delete语句删除表中数据。

```
delete from tbl_name [WHERE where_definition]
```

- 如果不使用where子句，将删除表中所有数据。
- Delete语句不能删除某一列的值（可使用update）
  - update table\_name set 字段名="";
- 使用delete语句仅删除记录，不删除表本身。如要删除表，使用drop table语句。
  - drop table table\_name;
- 同insert和update一样，从一个表中删除记录将引起其它表的参照完整性问题，在修改数据库数据时，头脑中应该始终不要忘记这个潜在的问题。

### 外键约束

- 💡 • 删除表中数据也可使用TRUNCATE TABLE 语句，它和delete有所不同，参看mysql文档。

#### a. Delete语句练习

删除表中名称为'zs'的记录。

删除表中所有记录。

使用truncate删除表中记录。

## 5. Select语句(1)

基本select语句

```
SELECT [DISTINCT] * | {column1, column2, ..., column3...} FROM table;
```

select 指定查询哪些列的数据。

column指定列名。

\*号代表查询所有列。

from指定查询哪张表。

DISTINCT可选，指显示结果时，是否剔除重复数据

### 练习：

查询表中所有学生的信息。

查询表中所有学生的姓名和对应的英语成绩。

过滤表中重复数据。distinct去重

## 6. Select语句(2)

- 在select语句中可使用表达式对查询的列进行运算

```
SELECT * | {column1 | expression, column2 | expression, ...} FROM table;
```

- 在select语句中可使用as语句

SELECT column as 别名 from 表名;

### 练习

在所有学生分数上加10分特长分显示。

统计每个学生的总分。

使用别名表示学生总分。

## 7. Select语句(3)

使用where子句，进行过滤查询。练习：

查询姓名为xxx的学生成绩

查询英语成绩大于90分的同学

查询总分大于200分的所有同学

from--where--group by--having--select--order by。

## 8. Select语句(4)

在where子句中经常使用的运算符

比较运算符	> < <= >= = <>	大于、小于、大于(小于)等于、不等于
	between ...and...	显示在某一区间的值
	in(set)	显示在in列表中的值，例： in(100,200)
	like '张pattern'	模糊查询%_ % 张%
	is null	判断是否为空 select * from user where id is null
	ifnull(原值,替代值)	如果原值为null，则使用代替值 select ifnull(score,0) from exam;
逻辑运算符	and	多个条件同时成立
	or	多个条件任一成立
	not	不成立，例： where not(salary>100);

Like语句中，% 代表零个或多个任意字符，\_ 代表一个字符，例  
first\_name like '\_a%' ;

## Select语句(4)练习

查询英语分数在 80 - 100之间的同学。

查询数学分数为75,76,77的同学。

查询所有姓张的学生成绩。

查询数学分>70，语文分>80的同学。

## 9. Select语句(5)

使用order by 子句排序查询结果。

```
SELECT column1, column2, column3..  
FROM table  
order by column_name asc|desc;
```

Order by 指定排序的列，排序的列既可是表中的列名，也可以是select 语句后指定的列名。

Asc 升序（默认）、Desc 降序

ORDER BY 子句应位于SELECT语句的结尾。

练习：

对语文成绩排序后输出。

对总分排序按从高到低的顺序输出

对姓李的学生成绩排序输出

## 10. 聚集函数 - count

Count(列名)返回某一列，行的总数

```
Select count(*) | count(列名) from tablename [WHERE where_definition]
```

练习：

统计一个班级共有多少学生？

统计数学成绩大于90的学生有多少个？

统计总分大于250的人数有多少？

## 11. 聚集函数 - SUM

Sum函数返回满足where条件的行的和

```
Select sum(列名) { , sum(列名) ... } from  
tablename [WHERE where_definition]
```

练习：

统计一个班级数学总成绩？

统计一个班级语文、英语、数学各科的总成绩

统计一个班级语文、英语、数学的成绩总和

统计一个班级语文成绩平均分

注意：sum仅对数值起作用，否则会报错。

注意：对多列求和，“，”号不能少。

## 12. 聚集函数 - AVG

AVG函数返回满足where条件的一列的平均值

```
Select avg(列名) { , avg(列名) ... } from tablename  
[WHERE where_definition]
```

```
[WHERE where_definition]
```

练习：

求一个班级数学平均分？

求一个班级总分平均分？

### 13. 聚集函数 - MAX/MIN

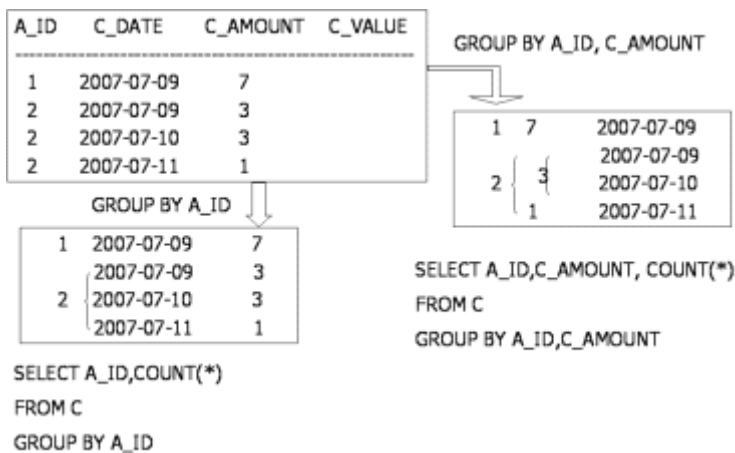
Max/min函数返回满足where条件的一列的最大/最小值

```
Select max(列名) from tablename [WHERE  
where_definition]
```

练习：

求班级最高分和最低分（数值范围在统计中特别有用）

### 14. Select语句(6)分组操作



使用group by 子句对列进行分组

```
SELECT column1, column2, column3..  
FROM table  
group by column having ...
```

练习：对订单表中商品归类后，显示每一类商品的总价

- 使用having 子句 对分组结果进行过滤

练习：查询购买了几类商品，并且每类总价大于100的商品

- where和having区别：where在分组前进行条件过滤，having在分组后进行条件过滤。使用where的地方都可以用having替换。但是having可以使用分组函数，而where后不可以使用。

# 数据库的备份、还原

2019年5月11日 星期六

10:22

## 1. 备份数据库表中的数据

```
cmd> mysqldump -u 用户名 -p 数据库名 > 文件名.sql
```

```
mysqldump -uroot -p db_name > d:/1.sql
```

## 2. 恢复数据库

( 注意：如果数据库已经删除，先创建数据库再恢复数据。 )

方式一: 在cmd中:

```
mysql -u 用户名 -p 数据库名 < 文件名.sql
```

```
mysql -uroot -p db_name < d:/1.sql
```

```
mysql -uroot -p mydb3 < d:/1.sql
```

方式二: 在mysql客户端中

```
source 文件名.sql
```

```
source d:/1.sql
```

## 3. 练习

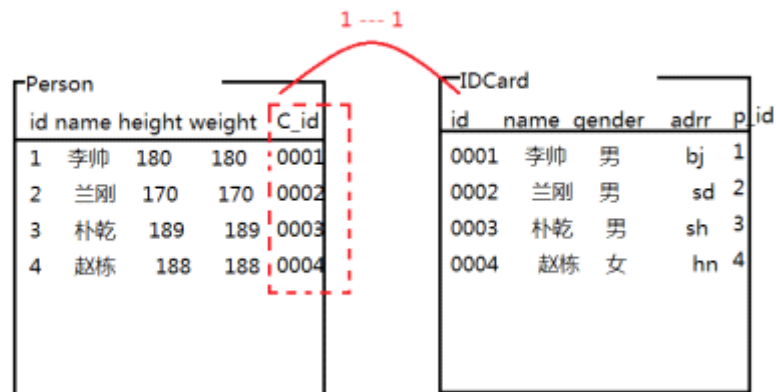
备份test库中的数据，并恢复

# 多表设计

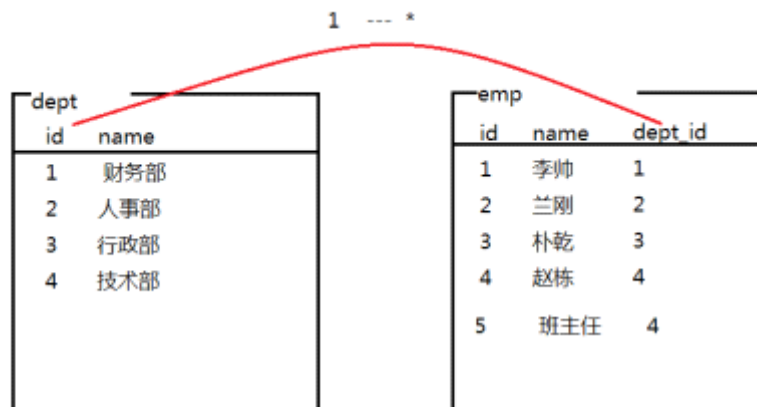
2019年5月11日 星期六

10:42

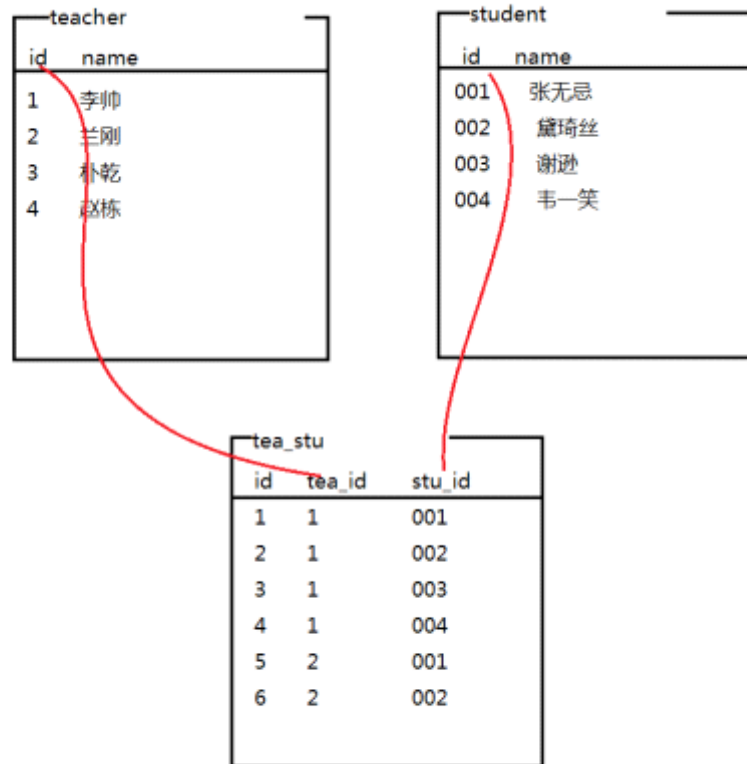
## 1. 表与表之间的关系



在1对1的表关系中，维护两张表的关系可以是在任意一张表的内部添加另外一张表的id字段，这样两张表之间就通过id建立的联系。



在一对多的表关系看中，需要在多的一张表中维护一个1的表的id字段，这样就可以维护两张表之间的关系。



在多对多的表关系时，可以建立一张中间表，在中间表中记录两张表的id字段，这两个id字段就是多对多关系的维护方式。



## 1. 外键:

- 用来通知数据库表与表字段之间的对应关系, 并让数据库帮我们维护这样关系的键就叫做外键
- 外键作用: 维护数据的完整性 一致性
- 定义外键约束

```
foreign key (ordersid) references orders(id)
```

## 2. 案例:

新建部门表dept(id,name)

通过外键约束建立与员工表emp关系

```
create table dept(  
  id int primary key auto_increment,  
  name varchar(20)  
);
```

```
insert into dept values (null, '财务部');  
insert into dept values (null, '人事部');  
insert into dept values (null, '科技部');  
insert into dept values (null, '销售部');
```

```
create table emp(  
  id int primary key auto_increment,  
  name varchar(20),  
  dept_id int,  
  foreign key (dept_id) references dept(id)  
);
```

```
insert into emp values (null, '张三', 1);  
insert into emp values (null, '李四', 2);  
insert into emp values (null, '老王', 3);  
insert into emp values (null, '赵四', 4);  
insert into emp values (null, '刘能', 4);
```

## 1. 如何确定一个员工在哪一个部门中？

将两张表的数据联合起来查询。观察是否会自动匹配。

## 2. 笛卡尔积查询

- a. 直接书写两张表的名称进行查询即可获取笛卡尔积查询结果。

```
select * from dept, emp;
```

- b. 笛卡尔积的查询结果是两张表数据相乘。如果表1有m条数据，表2有n条数据，则笛卡尔积的查询结果数量就是m\*n

id	name	id	name	dept_id
1	财务部	1	张飞	1
2	人事部	1	张飞	1
3	科技部	1	张飞	1
4	销售部	1	张飞	1
1	财务部	2	关羽	2
2	人事部	2	关羽	2
3	科技部	2	关羽	2
4	销售部	2	关羽	2
1	财务部	3	刘备	3
2	人事部	3	刘备	3
3	科技部	3	刘备	3
4	销售部	3	刘备	3
1	财务部	4	赵云	4
2	人事部	4	赵云	4
3	科技部	4	赵云	4
4	销售部	4	赵云	4

- c. 在笛卡尔积查询结果中有错误和正确的数据，只需要筛选出正确的数据即可。所以添加上判断条件。

```
select * from dept, emp where dept.id = emp.dept_id;
```

## 3. 内连接查询

关键字 inner join

- a. 写在innerjoin左侧的表称之为左边表，右侧的表称之为右边表。  
b. 在笛卡尔积查询的结果之上，获取左边表有且右边表也有的记录，这样的操作称之为内连接查询。

```
select * from dept inner join emp
on dept.id = emp.dept_id;
```

## 4. 外连接查询：

## a. 左外连接查询：

在内连接查询的基础之上获取左边表有而右边表没有的数据。

```
select * from dept
left join emp
on dept.id = emp.dept_id;
```

表名使用别名：

```
select * from dept d
left join emp e
```

```
on d.id = e.dept_id;
```

添加数据:

```
insert into dept values(null, '市场部');
```

结果:

id	name	id	name	dept_id
1	财务部	1	张飞	1
2	人事部	2	关羽	2
3	科技部	3	刘备	3
4	销售部	4	赵云	4
5	市场部	NULL	NULL	NULL

a. 右外连接查询:

在内连接查询的基础之上获取右边表有而左边表没有的数据。

```
select * from dept
right join emp
on dept.id = emp.dept_id;
```

表名使用别名:

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

添加数据:

```
insert into emp values(null, '路飞', 6);
```

b. 全外连接查询:

在内连接的查询基础之上, 获取左边表有而右边表没有的数据和右边表有而左边表没有的数据。

mysql中没有全外连接的关键字full join, 所以我们只能通过一个union关键字实现全外连接的查询效果。

union是一个联合查询的关键字, 在这个关键字结果中, 如果有相同的结果数据, 则只会保留一份相同的数据。

```
select * from dept
full join emp
on dept.id = emp.dept_id;
```

----- 没有full join

```
select * from dept left join emp on dept.id=emp.dept_id
```

```
union
```

```
select * from dept right join emp on dept.id = emp.dept_id;
```

