

## 1. 数据库概述

数据库就是一个存储数据的仓库。

数据库的发展：层次式数据库, 网络式数据库, 关系型数据库

## 2. 数据库

- 使用关系模型来存储的数据的数据库叫做关系型数据库。

常见数据库

- 商业数据库

Oracle  
SQLServer  
DB2  
Sybase

- 开源数据库

MySQL  
SQLite

## 3. Mysql数据库的安装

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

`root password:root`

-u：后面的root是用户名，这里使用的是超级管理员root；

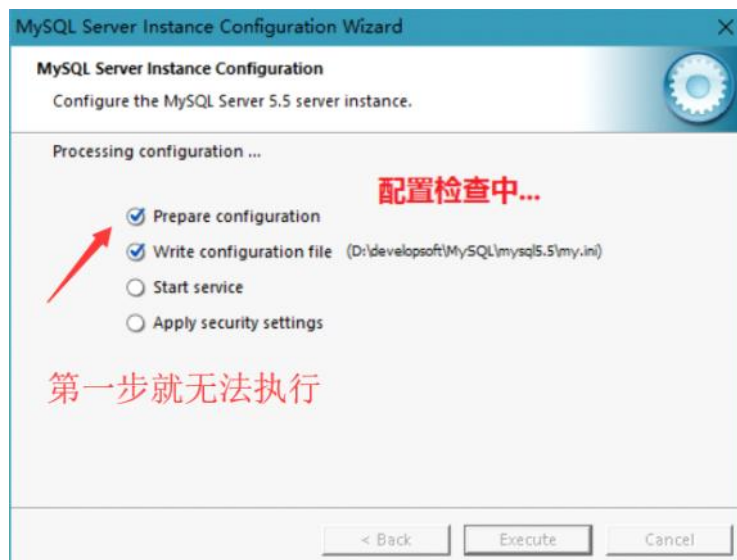
-p：后面的123是密码，这是在安装MySQL时就已经指定的密码；

退出：`quit`或`exit`；

`mysql -uroot -proot`

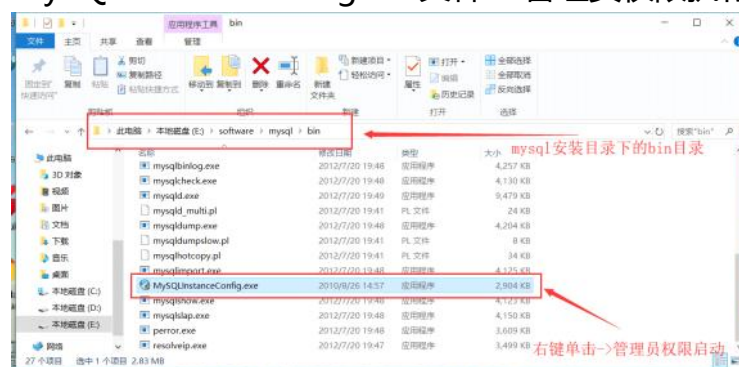
安装过程中出现的问题：

- 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

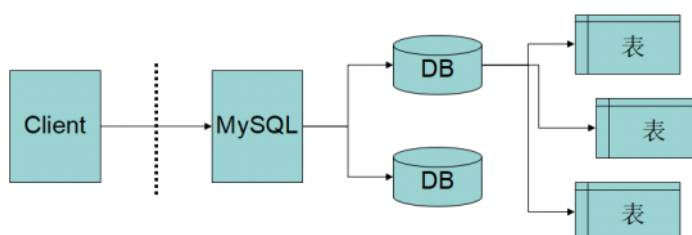
然后enter就可以了。

#### 4. MySQL数据库服务器、数据库和表的关系

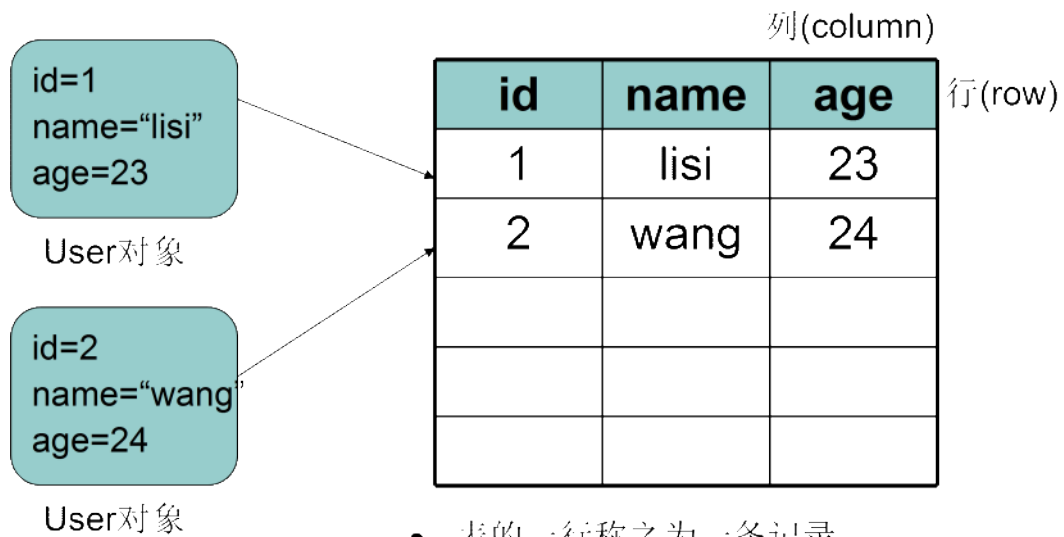
所谓安装数据库服务器，只是在机器上装了一个数据库管理程序，这个管理程序可以管理多个数据库，一般开发人员会针对每一个应用创建一个数据库。

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



- 表的一行称之为一条记录
- 表中一条记录对应一个java对象的数据

**注意：**数据库中的列也称之为**字段或域**

## 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。

## 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;
```

### 1. 创建表(基本语句) 创建表之前要先use database\_name;

```
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 在数据库中使用char或varchar保存
password	string
birthday	date

### 2. MySQL常用数据类型

#### ○ 字符串型

- VARCHAR(20) 0~255
- CHAR(20) 0~255

varchar在存储数据的时候数据的长度不是固定的，可以在指定的范围内存储任意长度的数据。varchar在读取数据的时候效率较低。

char在存储数据的时候数据的长度是固定的，尽管输入的数据没有达到指定的长度也会占用数据库中的指定的长度。char读取数据的时候效率较高。

#### ○ 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

### 3. 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
);
```

创建完毕之后利用desc employee 来查看表结构。

#### 4. 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 ;
  - 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

#### 5. 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;
```

## 6. 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 datatype [DEFAULT expr]
[, column datatype]..;
ALTER TABLE table_name MODIFY column datatype [DEFAULT expr]
[, column datatype]...;
ALTER TABLE table_name DROP column;
```

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

## 7. 删除表

```
drop table tabName;
```

## 1. 插入数据 create

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

### 3.1 插入数据

#### 3.1.1

```
insert into table_name values(column_value,column_value...);
```

```
insert into employee
```

```
values(null,'caoyang','ma','1890-09-09','1990-09-09','dance',5000.0,'like dance');
```

```
insert into employee
```

```
values(null,'lishuai','ma','1790-01-09','1910-01-09','sing',15000.0,'like sing');
```

```
insert into employee
```

```
values(null,'piaopian','ma','1000-09-09','1992-02-08','basketball',25000.0,'like basketball');
```

#### 3.1.2

```
insert into table_name(id,name) values(null,'mxh');
```

```
insert into employee(id,name) values(null,'mxh');
```

```
insert into employee(name,id) values('ysq',null);
```

### 3.2 插入中文出现乱码

```
insert into employee values(null,'曹洋','男','1000-09-09','1992-02-08','唱歌',25000.0,'爱唱歌');
```

---语句会报错，产生乱码。

解决：

乱码产生的原因是cmd窗口使用gbk字符集，数据库服务器使用utf-8字符集，两者不统一产生的。由于修改cmd窗口字符集十分繁琐，所以选择修改数据库的字符集。

修改方式：

1.在mysql客户端中输入 set names gbk;--临时修改，只在当前窗口内有效。

2.修改mysql根目录下的my.ini文件，将default-character-set=utf8改为default-character-set=gbk.重新启动mysql服务。--永久修改，修改过后，每一打开的客户端都是gbk字符集。

insert into语句特点：



- 1.向自动增长的字段插入数据时，可以预留为null。这个字段的值会自动添加，并增长。
- 2.字符串类型和日期类型的数据需要放在单引号或者双引号中间。
- 3.在插入数据的时候可以在表名之后指定字段，这样在填写插入的数据内容时，只需要指定对应字段的值即可。
- 4.在插入数据时不指定插入的字段名称，则需要在values后按照顺序填写所有字段对应值。
- 5.对应字段只能插入对应字段类型的数据。
- 6.插入数据的长度不能超过字段指定的长度。

## 2. 更新数据 update

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。

- 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。

update employee set salary = 5000;

将姓名为'zs'的员工薪水修改为3000元。

update employee set salary = 3000 where name = 'caoyang';

将姓名为'lisi'的员工薪水修改为4000元,job改为ccc。

update employee set salary = 4000,job='ccc' where name = 'lishuai';

将'wu'的薪水在原有基础上增加1000元。

update employee set salary = salary + 1000 where name = 'piaoqian';

## 3. 删除数据 delete

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。

### 4. 查询数据 read

#### a. 基本查询语句

```
select * from table_name;
```

select 查询

\* 全部字段名称

from 选择表

```
select id,name from table_name;
```

```
select id,name from employee;
```

select \* 表示查询全部字段，还可以使用select 具体的字段名称 来查询指定的字符安。

思考： select \*

```
select id,name,gender,birthday,entry_date,job,salary,resume
```

两种查询方式谁的查询效率更高？

答：

第二种方式效率更高。因为第一个种查询方式无法使用索引。第二种在字段名称中有索引字段存在，所以可以使用索引查询，使用索引查询效率会高于不使用索引。

#### ~3.5.2 distinct 去重

```
select distinct * |column_name from table_name;
```

employee表中原有6条数据，去重结果查询如下：

```
+-----+
| name   |
+-----+
| caoyang |
| lishuai |
| piaoqian |
+-----+
```

#### b. 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。

#### c. 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

#### d. 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("计算机系","英语系")
	like '张pattern'	模糊查询%_
	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。

#### e. 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 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

#### f. 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？

#### g. 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。

#### h. 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]
```

练习：

求一个班级数学平均分？

求一个班级总分平均分？

#### i. 聚集函数 - MAX/MIN

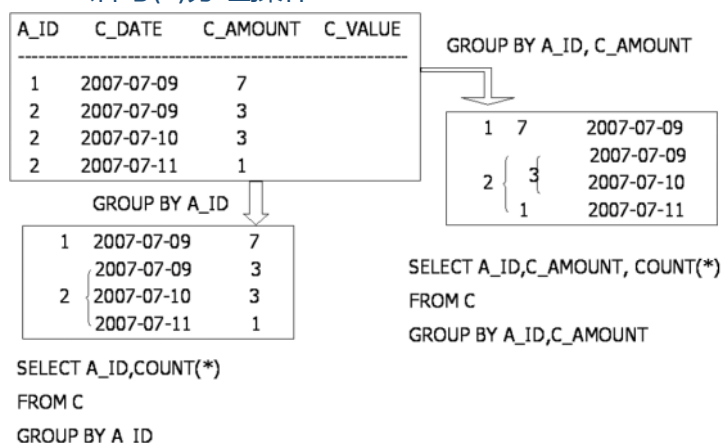
Max/min函数返回满足where条件的一列的最大/最小值

```
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）

#### j. 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

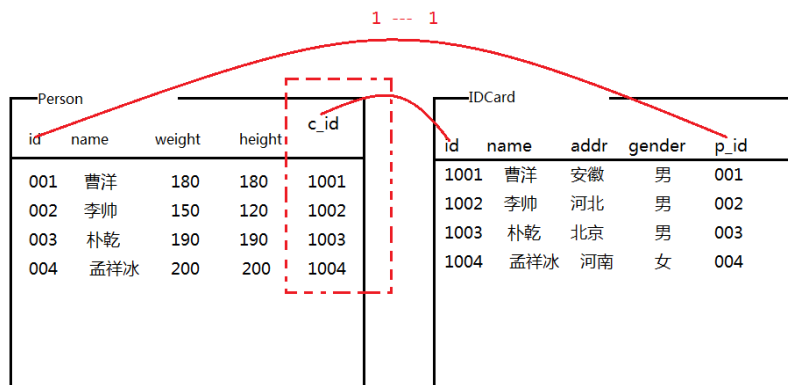
练习：查询购买了几类商品，并且每类总价大于100的商品

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

## 1. 表间关系

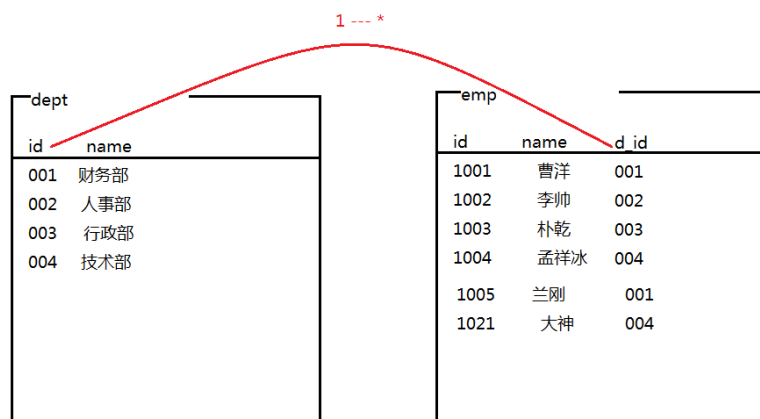
在数据库中可能会涉及到两张及两个以上表的操作，多个表之间存储一定关系，关系模型包括1对1, 1对多，多对多三种形式。

### 2. 1对1



在一对一中，可以在任意一张表中建立另一张表的id字段，这个字段就是用来维护两张表之间的关系字段。

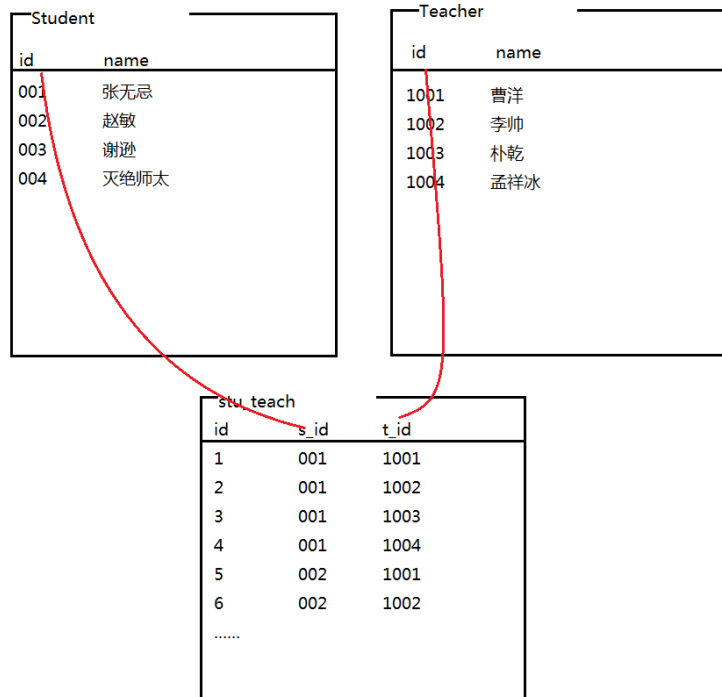
### 3. 1对多



在一对多中，可以在多的一张表中，建立一个1的id字段，这个字段可以用来维护两张表之间的关系。

### 4. 多对多

\* --- \*



在多对多的关系中，可以通过一个中间表来维护表间关系。  
中间表中需要填写两张表的id字段，通过这两个id字段维护表间关系。

### 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. 笛卡尔积查询

在查询语句中，选择两张表，表结果数据会是左边表的行数乘以右边表的行数（左边表m行，右边表n行，结果行数  $m*n$ ）

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

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

### 2. 内连接查询

在笛卡尔积的查询基础之上，获取左边表有且右边表也有的数据，这种查询方式称之为内连接查询。

- a. 从笛卡尔积中选择满足条件的结果数据：

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

```
where
```

```
dept.id = emp.dept_id;
```

inner join...on 关键字 表示内连接查询

```
select * from dept
```

```
inner join emp
```

```
on
```

```
dept.id = emp.dept_id;
```

### 3. 外连接查询

- a. 向表中插入测试数据：

```
insert into dept values(null,'小卖部');
```

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

- b. 左外连接查询：

在内连接查询的基础之上，获取左边表有且右边表没有的数据，这种查询称之为左外链接查询。

left join ... on关键字 表示外连接查询

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

★ 表使用别名查询:

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

#### 4. 右外连接查询:

在内连接查询的基础之上, 获取右边表有且左边表没有的数据, 这种查询称之为右外连接查询。

```
right join ... on关键字 表示外连接查询
select * from dept
right join emp
on
dept.id = emp.dept_id;
```

#### 5. 全外连接查询:

在内连接查询的基础之上, 获取左边表有且右边表没有的数据, 和右边表有且左边表没有的数据, 这种查询称之为全外连接查询。

full join --- mysql中没有这个关键字。

union --- 联合 将两个查询结果合并到一张表中展示, 重复的数据会去掉。

```
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;
```