今日内容：MySQL多表查询、索引、事务

授课老师：习知_前吉

联系方式：QQ：3264935908

准备工作

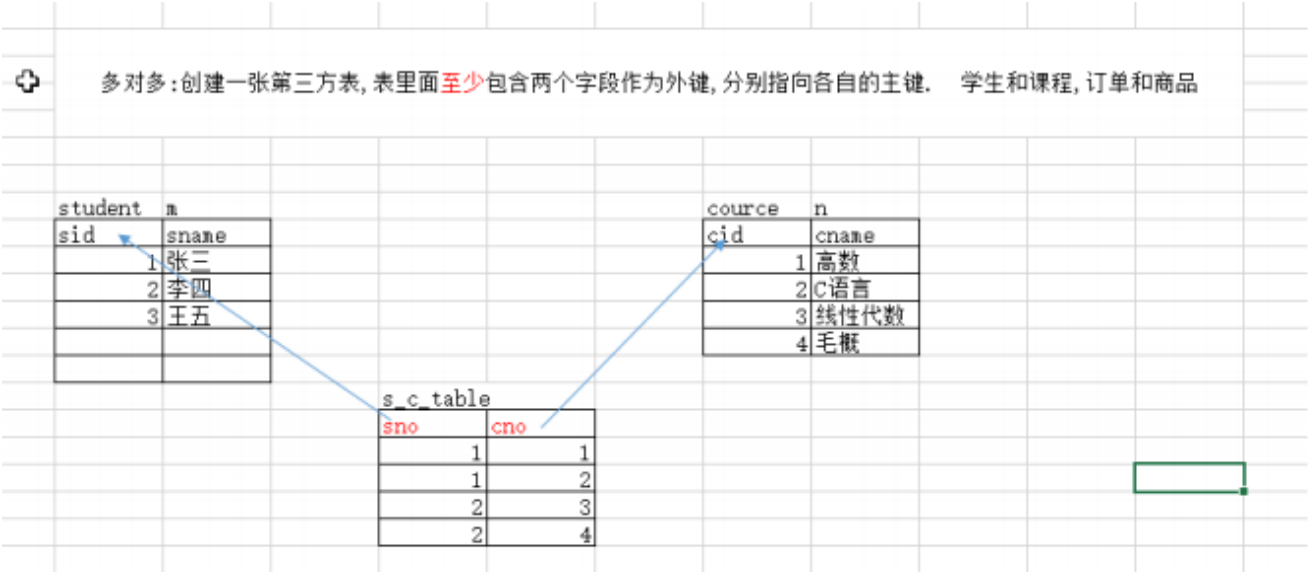
```sql
-- 部门表
create table dept(
    dno int primary key ,
    dname varchar(30),
    address varchar(30)
);

insert into dept values(10,'研发部','北京');
insert into dept values(20,'市场部','上海');
insert into dept values(30,'财务部','武汉');
insert into dept values(40,'运营部','深圳');

-- 员工表
CREATE TABLE emp (
    id INT PRIMARY KEY, -- 员工id
    ename VARCHAR(50), -- 员工姓名
    mgr INT , -- 上级领导
    joindate DATE, -- 入职日期
    salary DECIMAL(7,2), -- 工资,
    deptno int -- 部门id
);

-- 添加员工
INSERT INTO emp(id,ename,mgr,joindate,salary,deptno) VALUES
(1001,'孙悟空',1004,'2000-12-17','8000.00',10),
(1002,'卢俊义',1006,'2001-02-20','16000.00',10),
(1003,'林冲',1006,'2001-02-22','12500.00',10),
(1004,'唐僧',1009,'2001-04-02','29750.00',10),
(1005,'李逵',1006,'2001-09-28','12500.00',10),
(1006,'宋江',1009,'2001-05-01','28500.00',10),
(1007,'刘备',1009,'2001-09-01','24500.00',20),
(1008,'猪八戒',1004,'2007-04-19','30000.00',20),
(1009,'罗贯中',NULL,'2001-11-17','50000.00',20),
(1010,'吴用',1006,'2001-09-08','15000.00',20),
(1011,'沙僧',1004,'2007-05-23','11000.00',30),
(1012,'李逵',1006,'2001-12-03','9500.00',30),
(1013,'小白龙',1004,'2001-12-03','30000.00',30),
(1014,'关羽',1007,'2002-01-23','13000.00',null);
```

多表查询

多表间关系（了解）

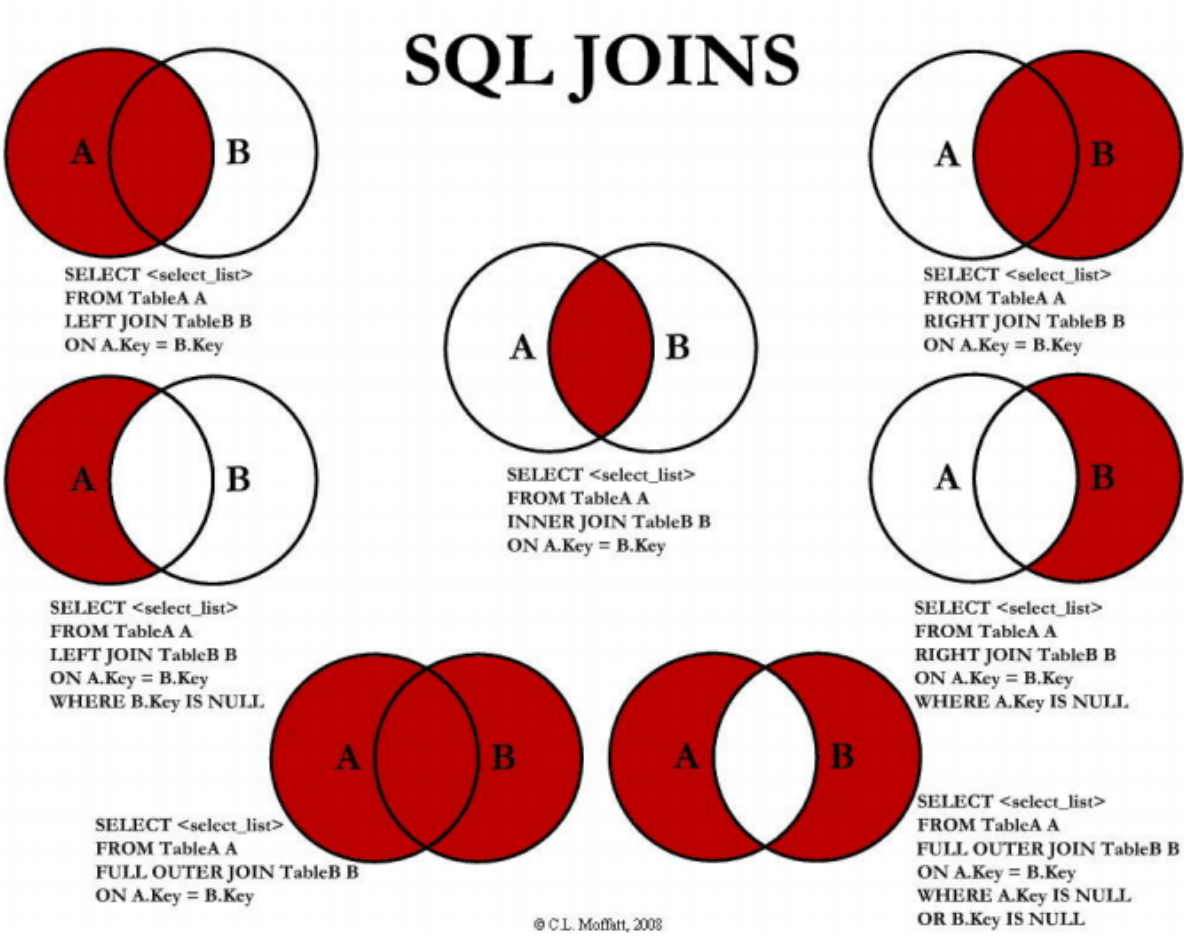
- 一对多关系
  - 主表的一行数据可以同时对应从表的多行数据，反过来就是从表的多行数据指向主表的同一行数 据。
  - 建表原则：将少的一方作为主表，多的一方作为从表，在从表中指定一个字段作为外键，指向主表的主键

- 多对多关系
  - 两张表都是多的一方，A表的一行数据可以同时对应B表的多行数据，反之B表的一行数据也可以同时对应A表的多行 数据
  - 建表原则：因为两张表都是多的一方，所以在两张表中都无法创建外键，所以需要新创建一张中间表，在中间表中定义两个字段，这俩字段分别作为外键指向两张表各自的主键



多表关联查询

多表关联查询是使用一条SQL语句，将关联的多张表的数据查询出来



内连接查询

主表和从表中的数据都是满足连接条件则能够查询出来，不满足连接条件则不会查询出来

- 隐式内连接查询

隐式内连接查询里面是没有inner join关键字

```
select [字段,字段,字段] from a,b where 连接条件 # b表里面的外键 = a表里面的主键
```

- 显示内连接查询

显式内连接查询里面是有inner join关键字

```
select [字段,字段,字段] from a [inner] join b on 连接条件 [ where 其它条件]
```

外连接查询

- 左外连接查询

以join左边的表为主表,展示主表的所有数据,根据条件查询连接右边表的数据,若满足条件则展示,若不满足则以null显示。可以理解为：**在内连接的基础上保证左边表的数据全部显示**

```
-- 基本语法
select 字段 from a left [outer] join b on 条件

-- 只要a表存在的 我都要
-- select * from a表 LEFT JOIN b表 on deot.KEY = EMP.KEY;deot.KEY = EMP.KEY 连接字段
select * from dept LEFT JOIN emp on dept.dno = emp.deptno;

-- 只要a表里面存在，b表里面不存在的数据
-- select * from a表 LEFT JOIN b表 on deot.KEY = EMP.KEY B表的外键 is null;
select * from dept LEFT JOIN emp on dept.dno = emp.DEPTNO WHERE emp.DEPTNO IS NULL;
```

- 右外连接查询

以join右边的表为主表,展示右边表的所有数据,根据条件查询join左边表的数据,若满足则展示,若不满足则以null显示。可以理解为：**在内连接的基础上保证右边表的数据全部显示**

```
-- 基本语法
select 字段 from a right [outer] join b on 条件

-- 只要b表存在，我就要
-- select * from a表 RIGHT JOIN b表 on deot.KEY = EMP.KEY;deot.KEY = EMP.KEY 连接字段
select * from dept RIGHT JOIN emp on dept.dno = emp.deptno;

-- 只要b表里面存在的数据
-- select * from a表 RIGHT JOIN b表 on deot.KEY = EMP.KEY a表的主键 is null;
select * from dept RIGHT JOIN emp on dept.dno = emp.DEPTNO WHERE emp.DEPTNO IS NULL;
```

- 别名

```
-- 别名
select e.ename '姓名',e.salary '工资',d.address '城市',d.DNAME '部门' from dept d
LEFT JOIN emp e on d.dno = e.deptno;
```

子查询

- 如果一个查询语句嵌套在另一个查询语句里面，那么这个查询语句就称之为子查询，根据位置不同，分为：where型，from型，exists型。注意：不管子查询在哪里，子查询必须使用()括起来。
- 将一个select语句的结果作为另一个select语句的条件

where型

```
select * from emp where salary = (select max(salary) from emp)
```

from型

```
select * from  (select语句) 别名
```

exists型（了解）

```
# EXISTS用于检查子查询是否至少会返回一行数据
SELECT dno,dname FROM dept WHERE EXISTS (SELECT * FROM emp WHERE emp.deptno = dept.dno);
select * from dept where not exists (select * from emp where deptno = dept.dno )
```

事务[了解一下]

定义

- 事务（Transaction）：一个最小的不可再分的工作单元；通常一个事务对应一个完整的业务
- 一个完整的业务需要批量DML(数据操纵语言 insert、update、delete) 语句共同联合完成
- 事务只和DML语句有关，或者说DML语句才有事务。这个和业务逻辑有关，业务逻辑不同，DML语句的个数不同

事务的四大特性（ACID）

- Automicity:原子性,事务是最小的执行单位，不允许分割。
- Consistency:一致性，数据库从一个正确的状态变化到另一个正确的状态；
- Isolation:隔离性，并发访问数据库时，一个用户的事务不被其他事务所干扰，各并发事务之间数据库是独立的
- Durability:持久性，一个事务被提交之后。它对数据库中数据的改变是持久的，即使数据库发生故障也不应该对其有任何影响

事务开启的标志？事务结束的标志？

```
开启事务 begin
    任何一条DML语句(insert、update、delete)执行
结束标志(提交回滚) commit 或 回滚 rollback
```

> 提交：成功的结束，将所有的DML语句操作历史记录和底层硬盘数据来一次同步
>
> 回滚：失败的结束，将所有DML语句操作历史记录全部清空

事务产生的问题和事务的隔离界别

- 事务同时执行出现的问题
  - 脏读:事务A读取了B更新的数据，然后B回滚事务A读取到的就是脏数据
  - 不可重复读:事务A多次读取同一数据，事务B在事务A多次读取的过程中，对数据作了更新并提交，导致前后结果不一致。
  - 幻读:A管理员将学生按照分数分为ABCDE,B管理员在A操作完全部数据之后发现有一条记录没操作，好像出现幻觉一样。

> 总结：不可重复读侧重修改，幻读侧重新增和删除。不可重复读要锁符合条件的行数据，幻读要锁表

- 事务的隔离级别
  - 读未提交:一个事务还没提交时，它做的变更就能被别的事务看到。
  - 读提交(Orlace默认级别):一个事务提交之后，它做的变更才会被其他事务看到。解决脏读。
  - 可重复读(mysql默认级别):事务不会读到其它事务提交的修改（一开始读到什么，最后也读到什么）解决脏读、不可重复读

- 串行化：解决了脏读、不可重复读、幻读，相当于单进程，效率低下。

> truncate table 清空表， 事务不生效

索引

索引是什么

- 加索引，加快查询速度，避免慢sql
- 为什么where最好根据id来操作，id是主键索引，查询快。根据主键索引来查询 特别快

索引分类

- 普通索引
- 复合索引 -- 也叫多列索引

查看索引

```
show index from emp;
```

创建索引

```
# 创建索引的SQL， create index 索引名字(index_字段名) on 表名(字段);
CREATE INDEX index_ename on emp(ename);
```

创建索引的原则：

- where后面 用的比较多字段，都要加索引
- sql join 的字段，要加索引，多表查询很慢，最好用主键关键外键，外键加个索引
- 慢 sql （select * from 表名 like'%XX'）实际工作中，多了 会堆积，服务器承受不住了，就会宕机，全时报错，这个期间 用户用不了数据收集不到，钱飞了.

索引失效的原则

- select 不要用 * 为什么不要用 * 会全表扫描， 比如 你这个表有1亿条数据，全扫描一遍
- select查询的时候，使用like时 禁止% 在前面 %xx、%xx%
- select * 还有运算符号，字段计算都不走索引
- select 禁止用 or
- 尽量避免使用临时表 ， 大量的读写 会影响速度

> 字符串 时间 查询的时候 必须加上''
>
> 为什么： mysql底层有优化器，不加 这个sql会影响性能