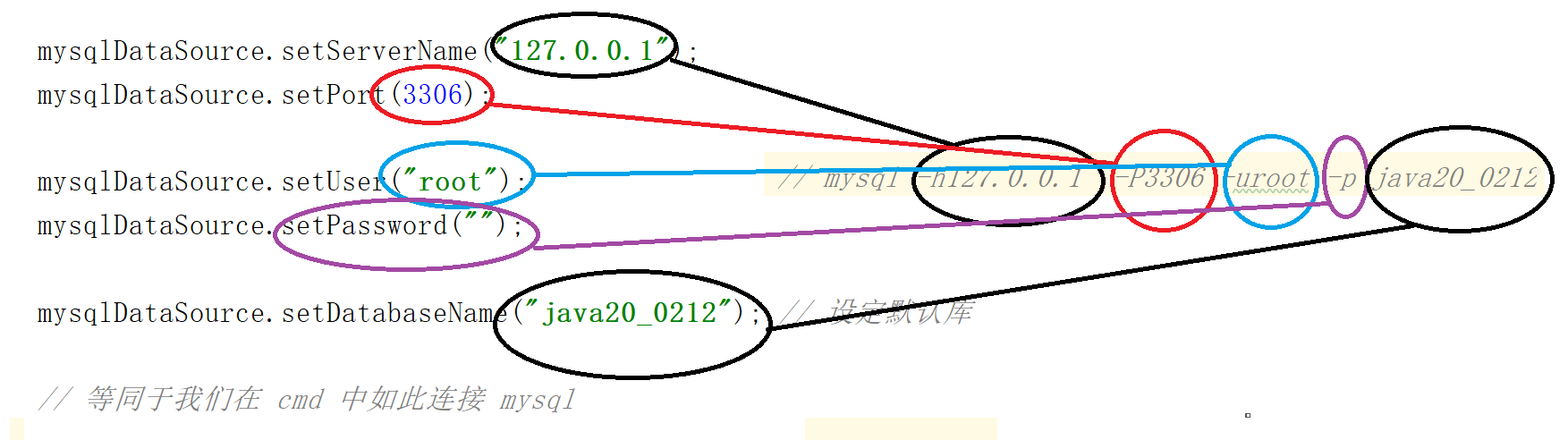
# 复习

CREATE DATABASE java20\_0212 CHARSET utf8mb4;

JDBC 的过程-代码演示

1. 创建 DataSource 对象，并且进行设置，DataSource 对象是全局只有一个就够了（单例模式）



1. 创建 Connection（不是线程安全的）
2. 创建 PrepareStatement
3. 执行 SQL（executeQuery|Update 有什么区别）
4. 销毁

注意点：

1. executeQuery 只能用于 SELECT
2. executeUpdate 不能用于 SELECT
3. execute 用于 DDL 语句

# 复杂查询(Select)

## 基本查询（5分）

SELECT 子句

FROM 子句

WHERE 子句

ORDER BY 子句

LIMIT 子句

## 聚合查询（3.5分）

GROUP BY 子句（可以独立出现）

HAVING 子句（必须跟在 GROUP BY 后边）

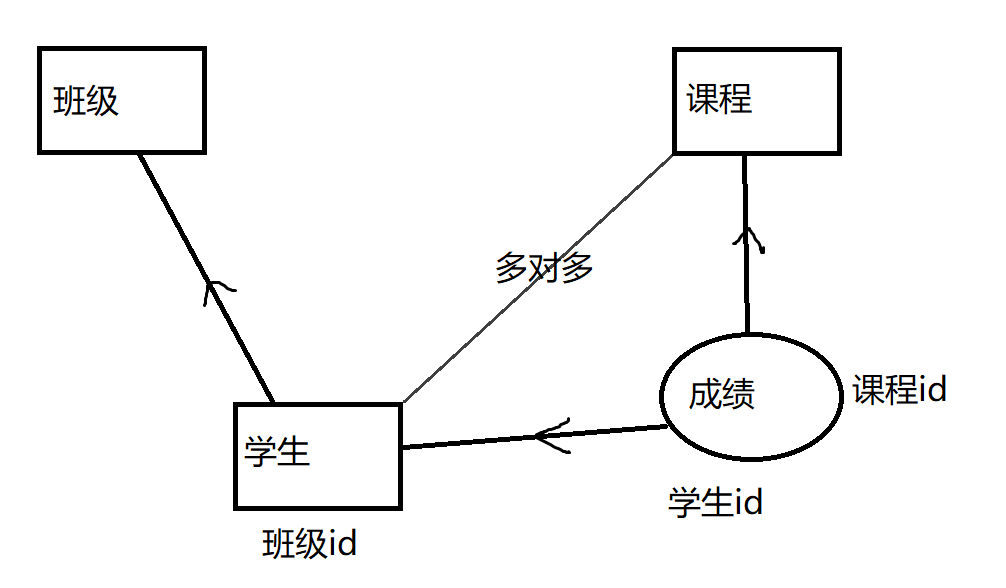
1. 聚合函数（独立于 GROUP BY 出现）
   1. COUNT
   2. SUM/AVG/MAX/MIN
2. GROUP BY
   1. 分组凭证
   2. 支持多个分组
   3. SELECT 子句有限制（MySQL 有例外）：聚合函数 OR 分组凭证
3. HAVING
   1. 和 WHERE 对比：WHERE 对聚合前的数据进行过滤；HAVING 对聚合后的数据进行过滤

## 联表查询（3.5 分）

多张表之间进行联系的查询，不加筛选条件，出现的结果就是一个笛卡儿积

一般都要使用联系字段进行过滤，例如：文章中的作者 id，评论中的作者id，评论中的文章id

练习专用的数据



create table classes(

id int primary key auto\_increment,

name varchar(255),

`desc` varchar(255)

);

create table student (

id int primary key auto\_increment,

sn int,

name varchar(255),

qq\_mail varchar(255),

classes\_id int

);

create table course (

id int primary key auto\_increment,

name varchar(255)

);

create table score (

student\_id int,

course\_id int,

score float

);

insert into classes(name, `desc`) values

('计算机系2019级1班', '学习了计算机原理、C和Java语言、数据结构和算法'),

('中文系2019级3班','学习了中国传统文学'),

('自动化2019级5班','学习了机械自动化');

insert into student(sn, name, qq\_mail, classes\_id) values

('09982','黑旋风李逵','xuanfeng@qq.com',1),

('00835','菩提老祖',null,1),

('00391','白素贞',null,1),

('00031','许仙','xuxian@qq.com',1),

('00054','不想毕业',null,1),

('51234','好好说话','say@qq.com',2),

('83223','tellme',null,2),

('3527','幽灵人',null,8),

('09527','老外学中文','foreigner@qq.com',2);

insert into course(name) values

('Java'),('中国传统文化'),('计算机原理'),('语文'),('高阶数学'),('英文');

insert into score(score, student\_id, course\_id) values

-- 黑旋风李逵

(70.5, 1, 1),(98.5, 1, 3),(33, 1, 5),(98, 1, 6),

-- 菩提老祖

(60, 2, 1),(59.5, 2, 5),

-- 白素贞

(33, 3, 1),(68, 3, 3),(99, 3, 5),

-- 许仙

(67, 4, 1),(23, 4, 3),(56, 4, 5),(72, 4, 6),

-- 不想毕业

(81, 5, 1),(37, 5, 5),

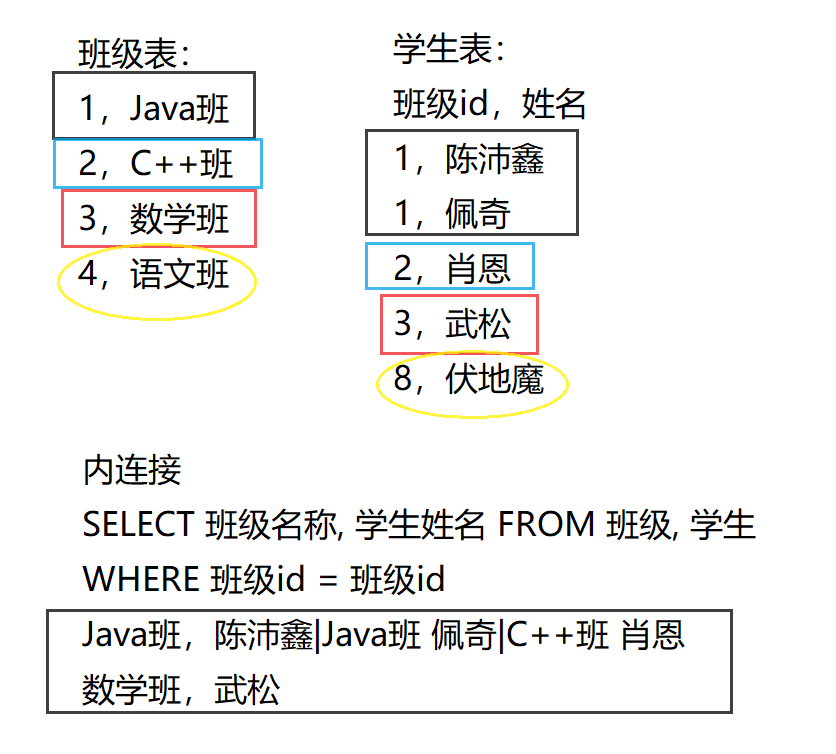
-- 好好说话

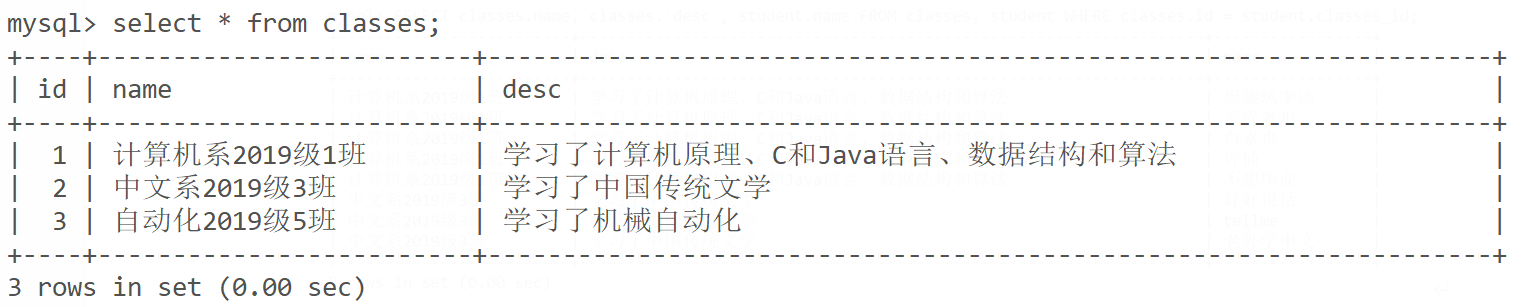
(56, 6, 2),(43, 6, 4),(79, 6, 6),

-- tellme

(80, 7, 2),(92, 7, 6);

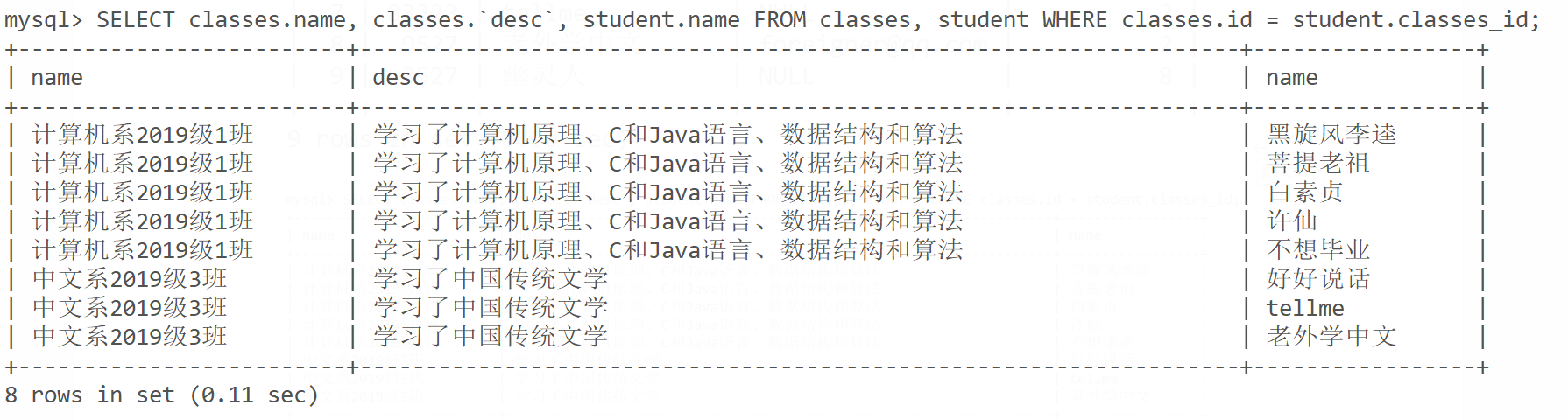
内连接（inner join） vs 外连接（outer join）

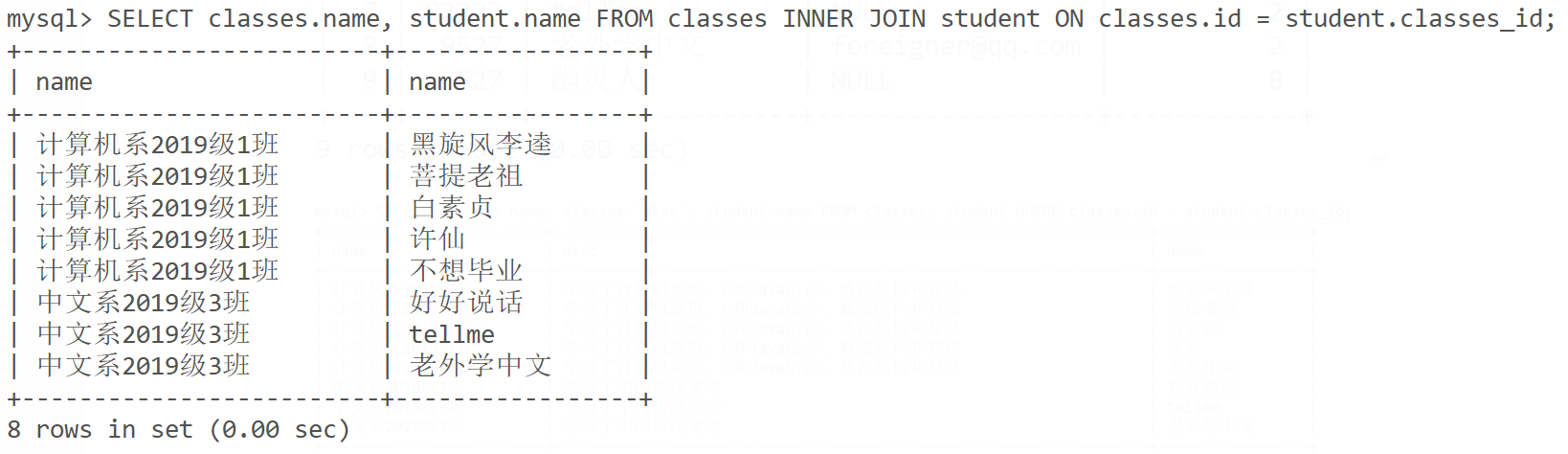


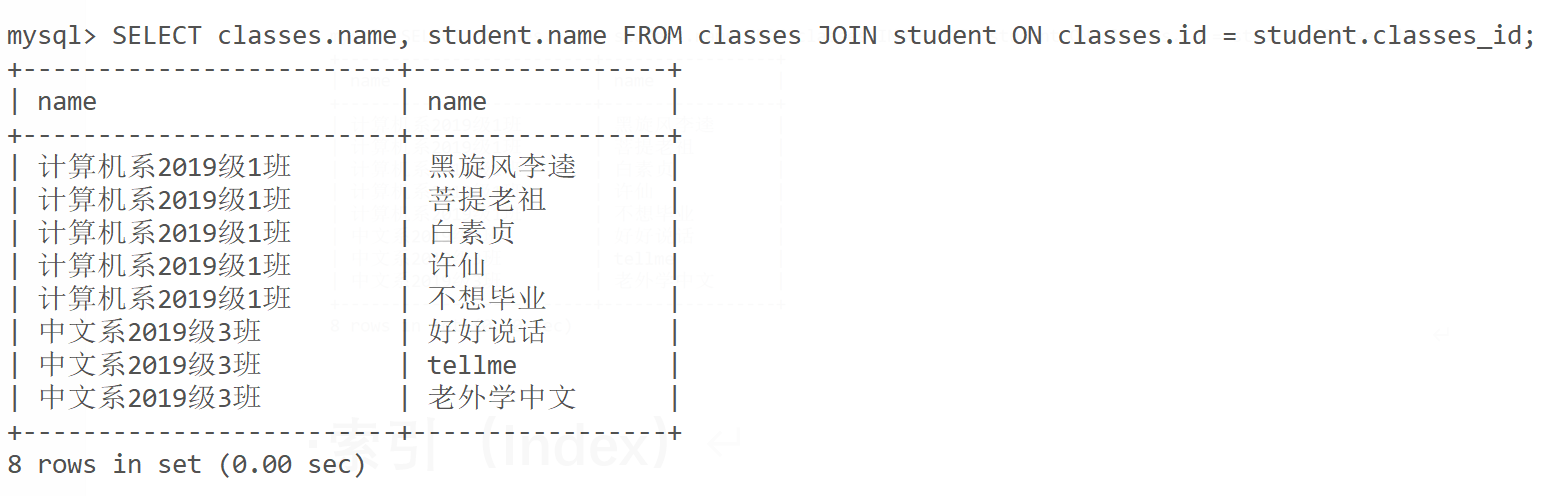




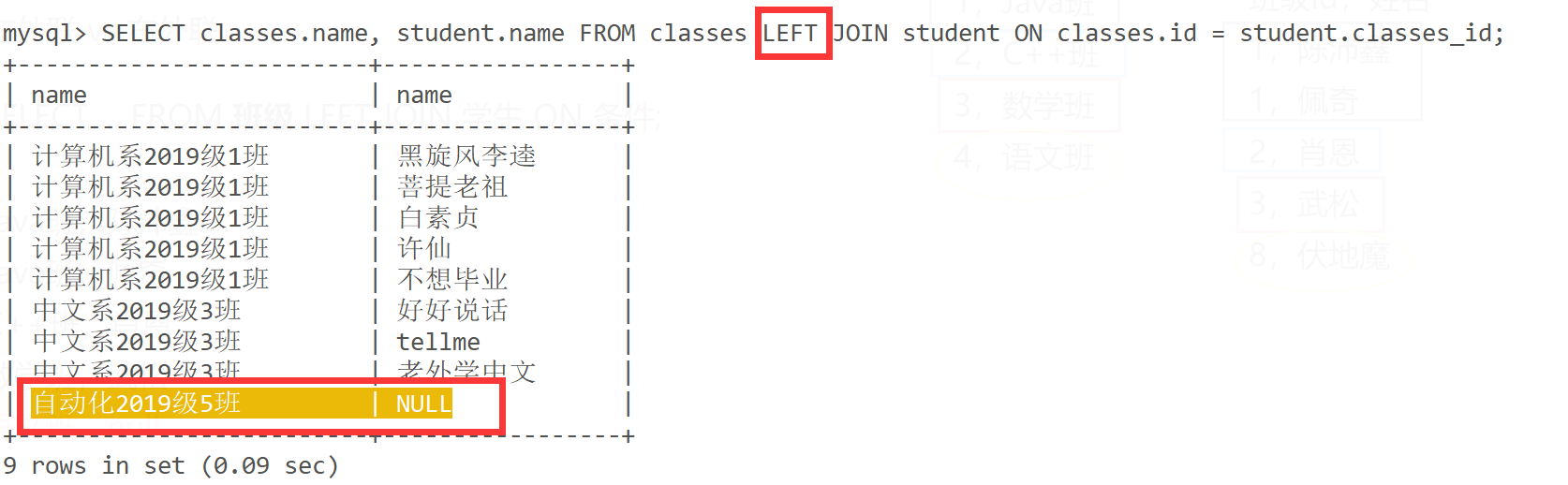
### 内联查询（常用的）







### 左外联



### 右外联

### 全外联（MySQL 不支持 Full Join）

### 练习

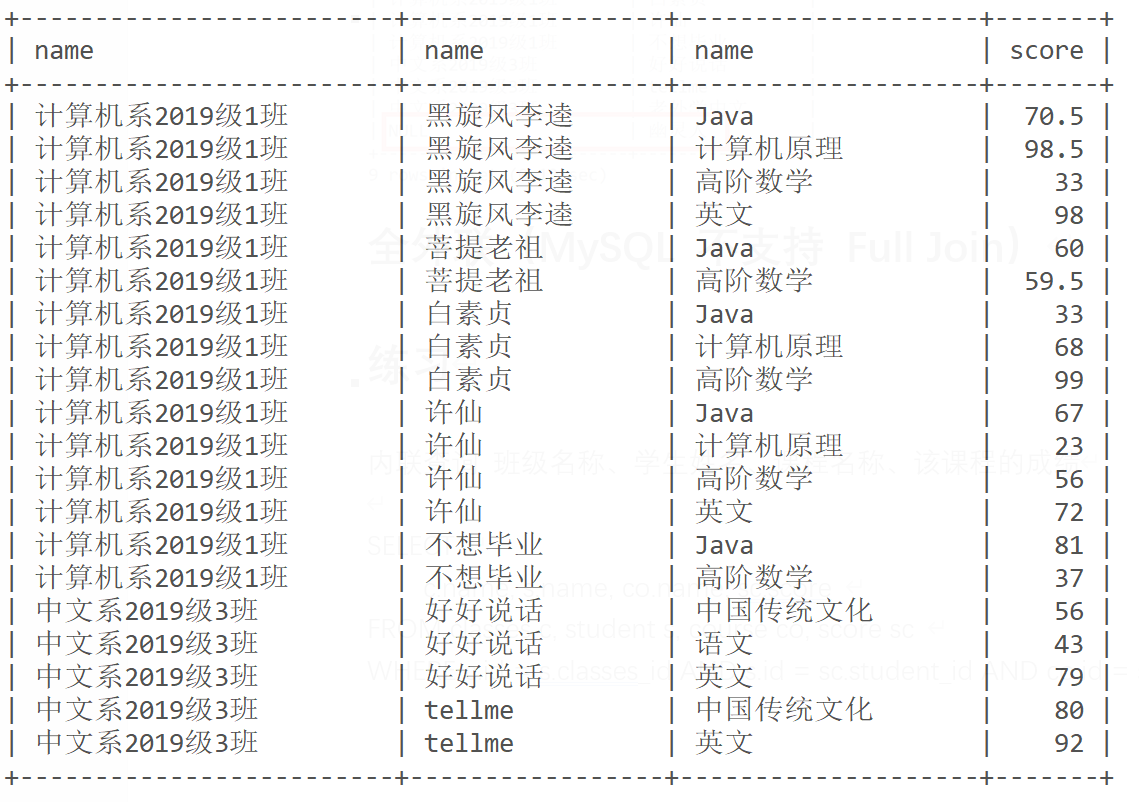
内联查询 班级名称、学生姓名、课程名称、该课程的成绩

SELECT

c.name, s.name, co.name, sc.score

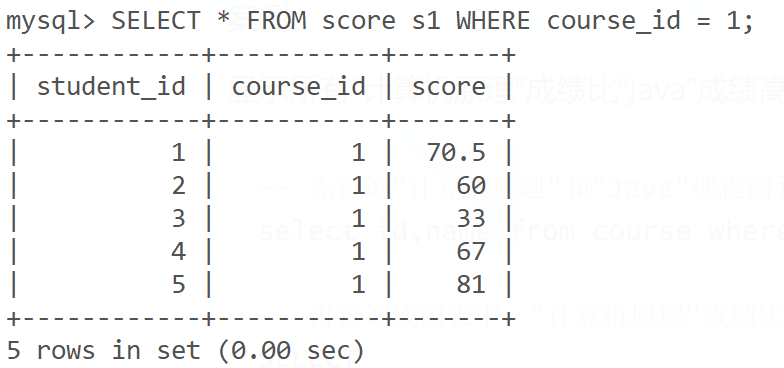
FROM classes c, student s, course co, score sc

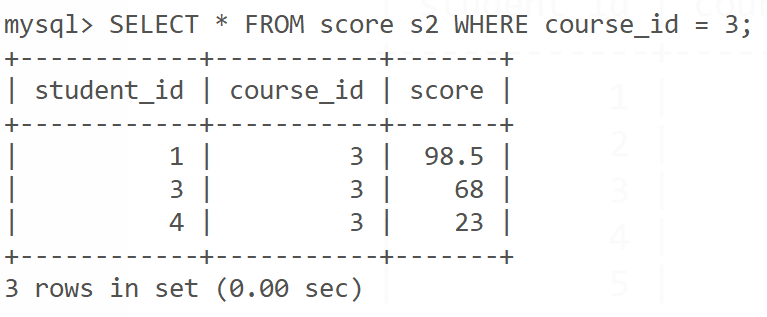
WHERE c.id = s.classes\_id AND s.id = sc.student\_id AND co.id = sc.course\_id;

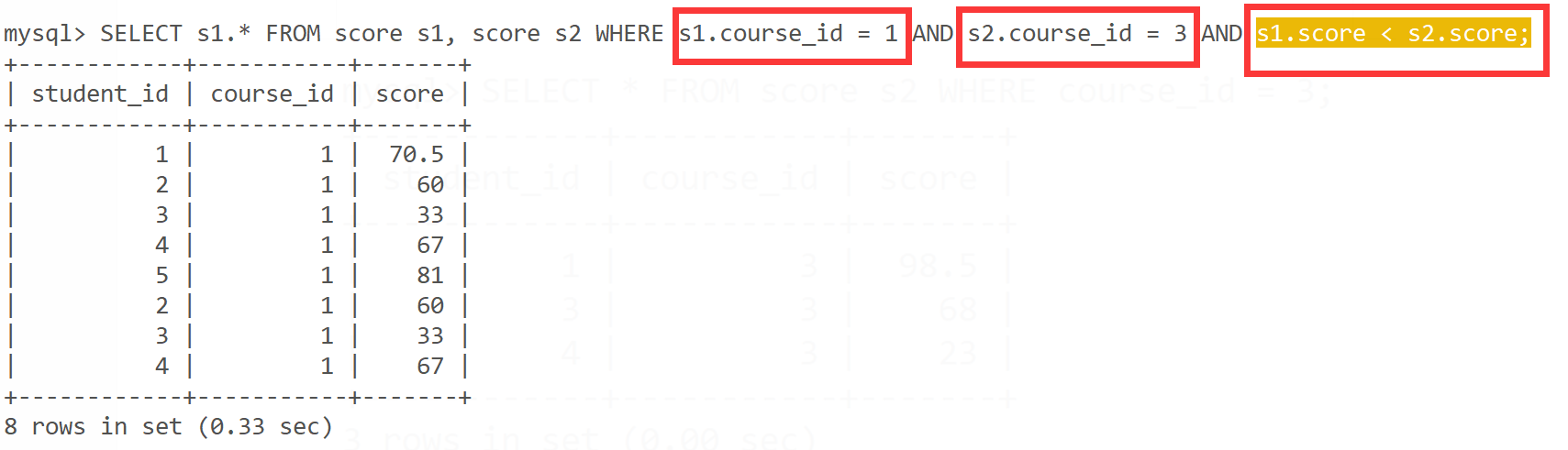


### 自连接

**因为是同一张表联表，所以必须起不同的别名以作区分**







## 子查询（2分）

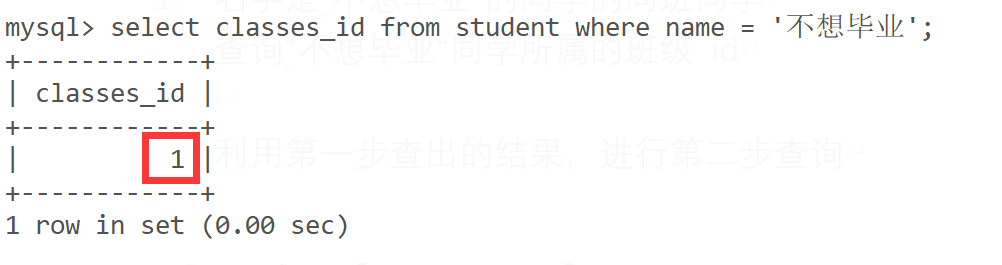
先通过一个 SELECT 查询，得到一组结果集。利用这个结果集做第二次查询

1. 把第一次的结果集看作一张表做新的查询
2. 把第一次的结果集作为过滤条件查询

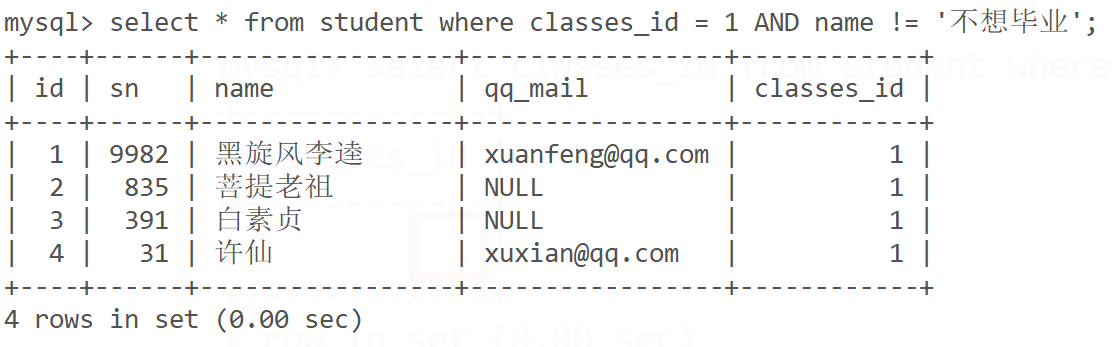
练习：

1. 名字是“不想毕业“的同学的同班同学

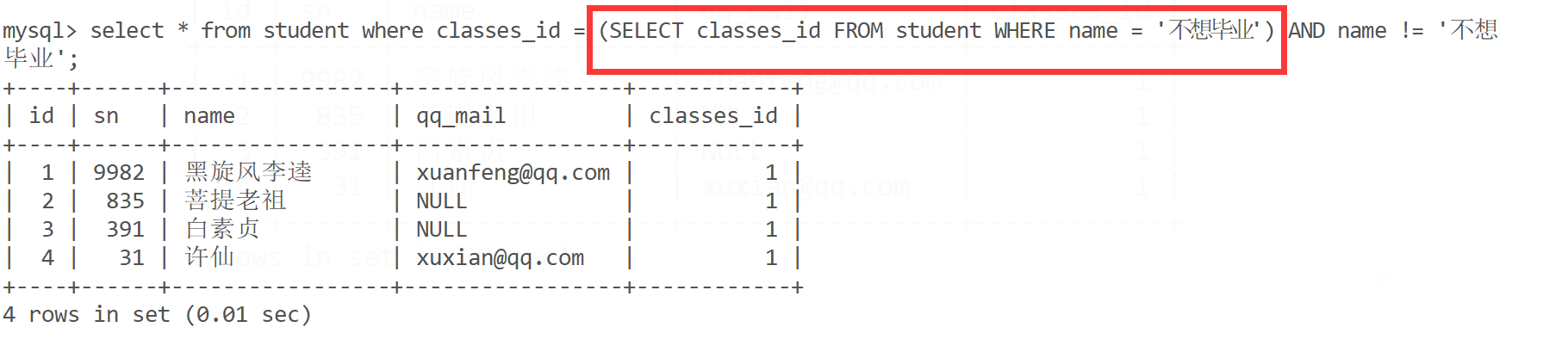
查询“不想毕业“同学所属的班级 id



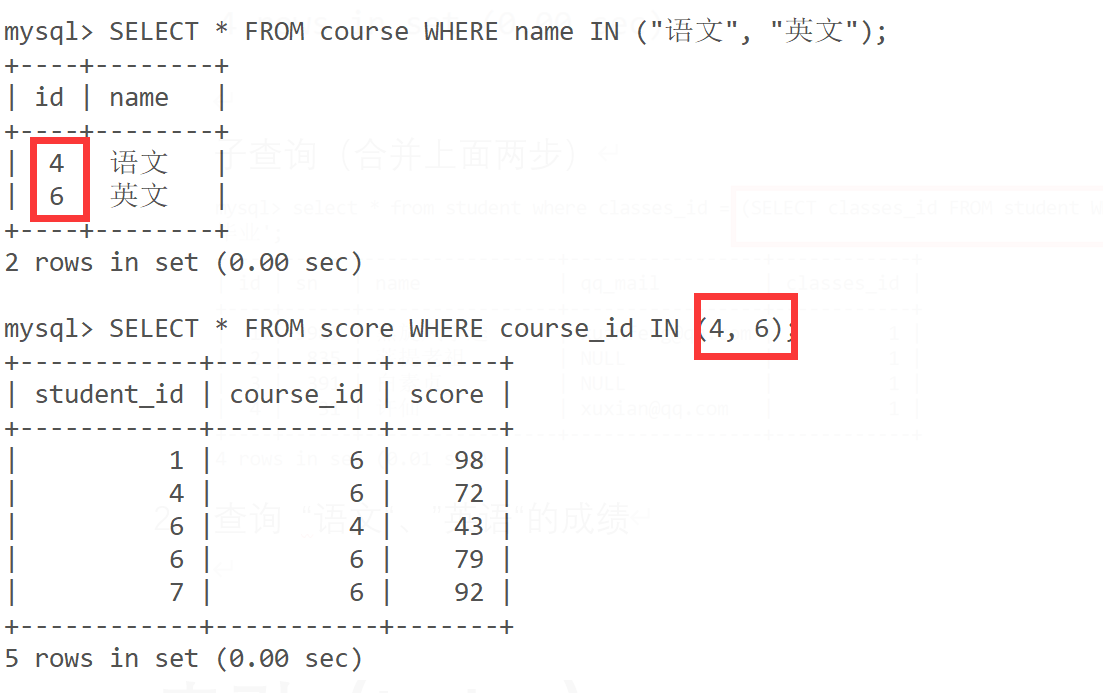
利用第一步查出的结果，进行第二步查询

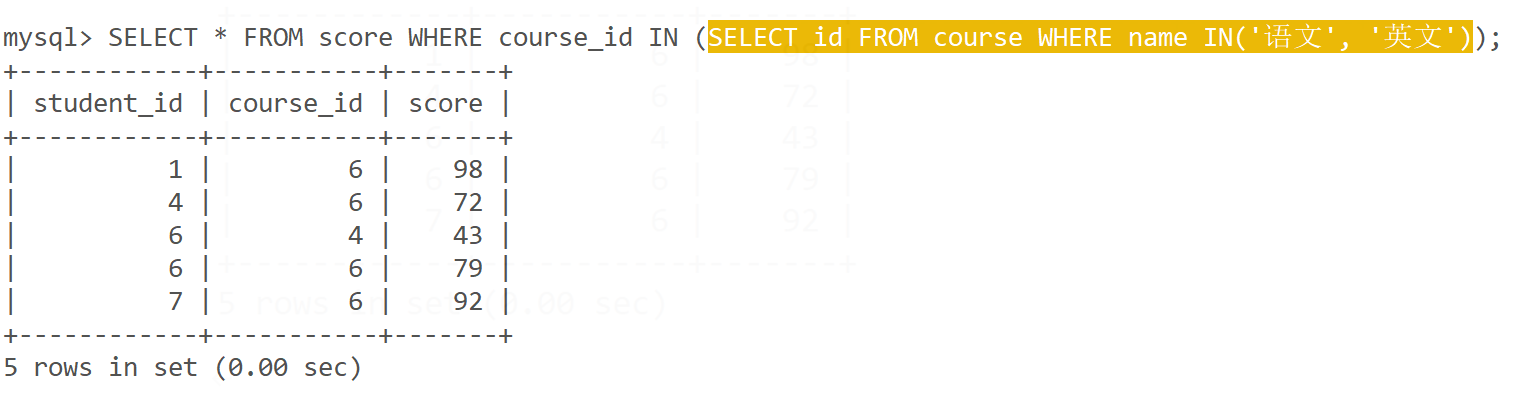


子查询（合并上面两步）



1. 查询 “语文“、”英语“的成绩





select \*

from score sco

where exists (

select sco.score from course cou where (name='语文' or name='英文') and cou.id = sco.course\_id

);

select \*

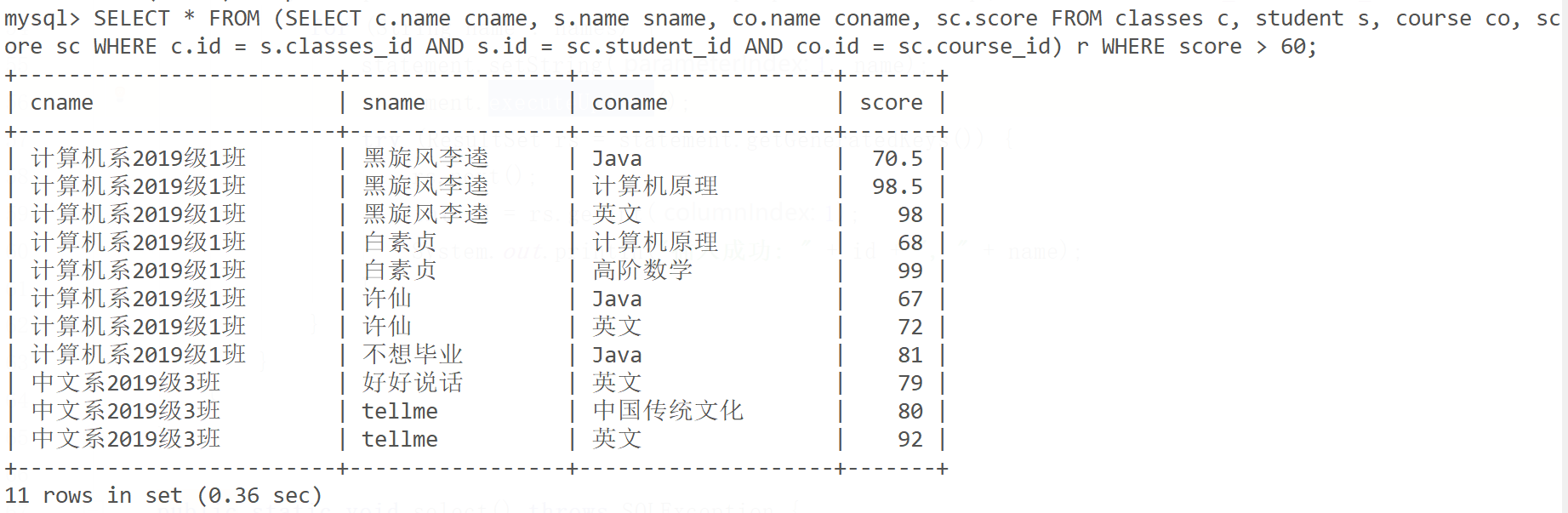
from score sco

where not exists (

select sco.score

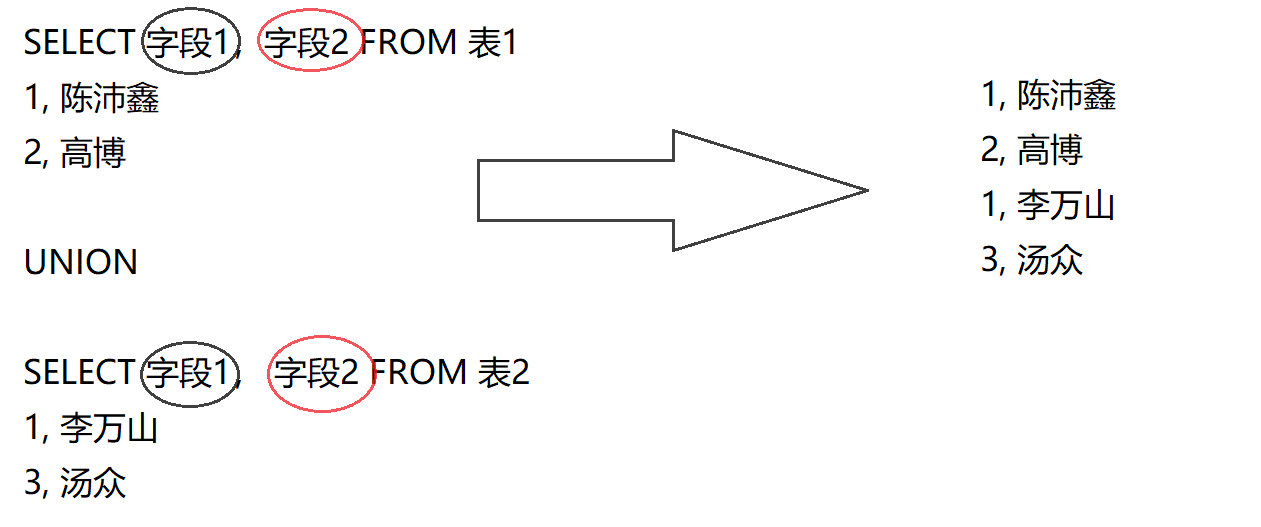
from course cou where (name!='语文' and name!='英文') and cou.id = sco.course\_id

);



必须起好别名

### 合并查询（1分）



UNION vs UNION ALL

UNION 会进行合并重复项

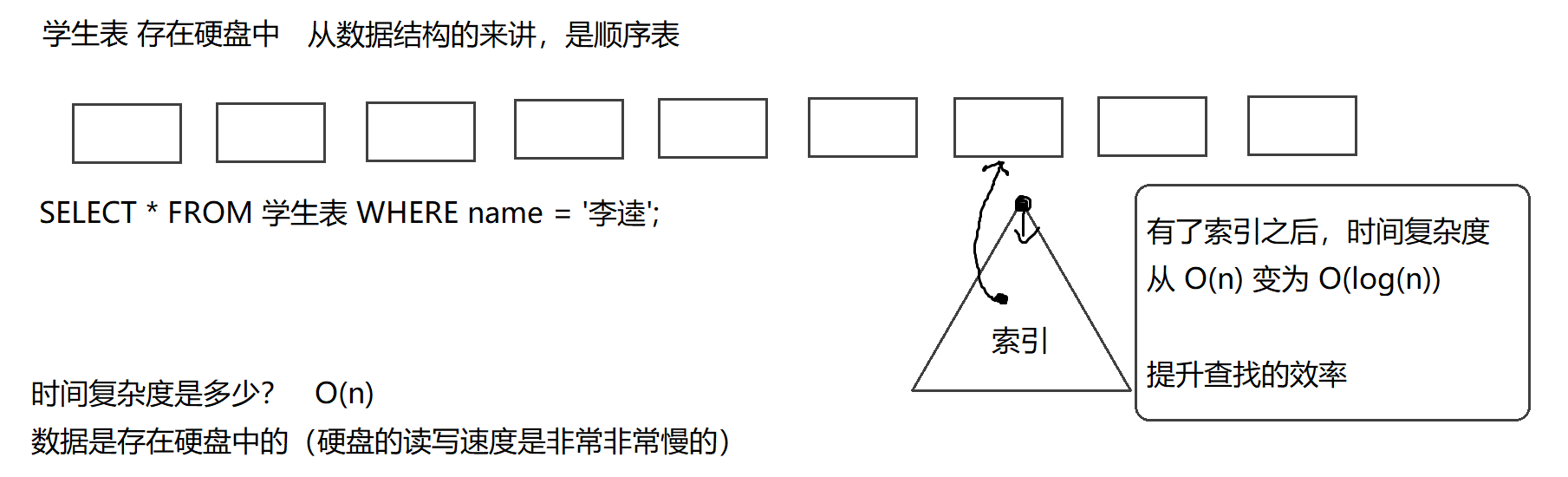
UNION ALL 不会合并重复项

## 总结(查询重点)

1. 基本查询
2. 聚合查询 GROUP BY | HAVING
3. 联表查询 内联 vs 左联 vs 右联
4. 子查询 作为 条件 vs 新的源
5. 联合查询 UNION vs UNION ALL

# 索引（Index）接触

加速查询的一种数据结构



### 优点 vs 缺点

提升查询效率-优点

降低了插入/修改/删除效率-缺点

增加了硬盘的使用空间-缺点

### 使用场景

索引不是无限制加的，根据需要来加

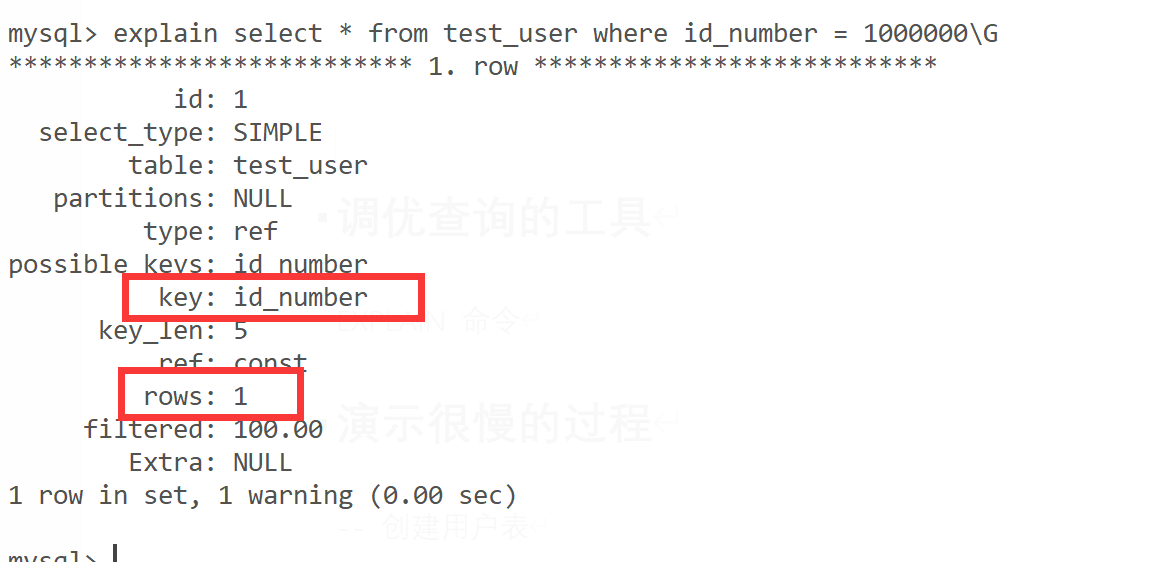
1. 数据量够大
2. 索引应该加在经常查找的字段上

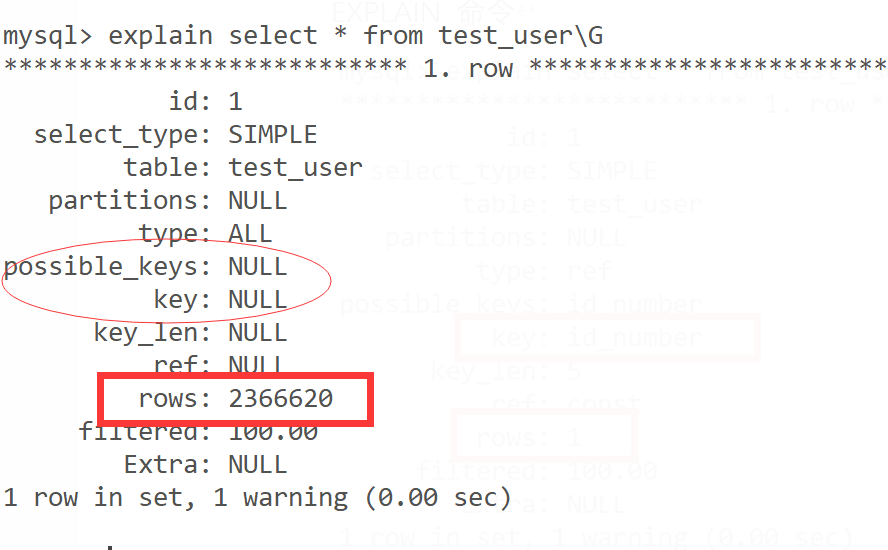
### 索引的分类

1. 主键 Primary Key
2. 唯一键 Unique
3. 普通索引（一般提到索引，没有特别说明，都是这个） Key|Index

### 调优查询的工具

EXPLAIN 命令





### 演示很慢的过程

-- 创建用户表

DROP TABLE IF EXISTS test\_user;

CREATE TABLE test\_user (

id\_number INT,

name VARCHAR(20) comment '姓名',

age INT comment '年龄',

create\_time timestamp comment '创建日期'

);

-- 构建一个8000000条记录的数据

-- 构建的海量表数据需要有差异性，所以使用存储过程来创建， 拷贝下面代码就可以了，暂时不用理解

-- 产生名字

drop function if exists rand\_name;

delimiter $$

create function rand\_name(n INT, l INT)

returns varchar(255)

begin

declare return\_str varchar(255) default '';

declare i int default 0;

while i < n do

if i=0 then

set return\_str = rand\_string(l);

else

set return\_str =concat(return\_str,concat(' ', rand\_string(l)));

end if;

set i = i + 1;

end while;

return return\_str;

end $$

delimiter ;

-- 产生随机字符串

drop function if exists rand\_string;

delimiter $$

create function rand\_string(n INT)

returns varchar(255)

begin

declare lower\_str varchar(100) default

'abcdefghijklmnopqrstuvwxyz';

declare upper\_str varchar(100) default

'ABCDEFJHIJKLMNOPQRSTUVWXYZ';

declare return\_str varchar(255) default '';

declare i int default 0;

declare tmp int default 5+rand\_num(n);

while i < tmp do

if i=0 then

set return\_str =concat(return\_str,substring(upper\_str,floor(1+rand()\*26),1));

else

set return\_str =concat(return\_str,substring(lower\_str,floor(1+rand()\*26),1));

end if;

set i = i + 1;

end while;

return return\_str;

end $$

delimiter ;

-- 产生随机数字

drop function if exists rand\_num;

delimiter $$

create function rand\_num(n int)

returns int(5)

begin

declare i int default 0;

set i = floor(rand()\*n);

return i;

end $$

delimiter ;

-- 向用户表批量添加数据

drop procedure if exists insert\_user;

delimiter $$

create procedure insert\_user(in start int(10),in max\_num int(10))

begin

declare i int default 0;

set autocommit = 0;

repeat

set i = i + 1;

insert into test\_user values ((start+i) ,rand\_name(2, 5),rand\_num(120),CURRENT\_TIMESTAMP);

until i = max\_num

end repeat;

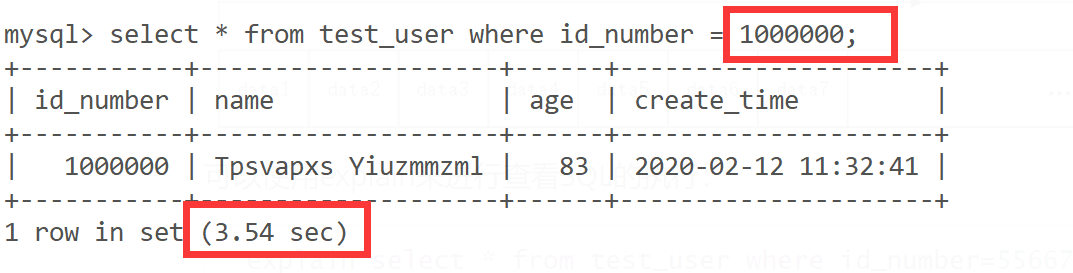
commit;

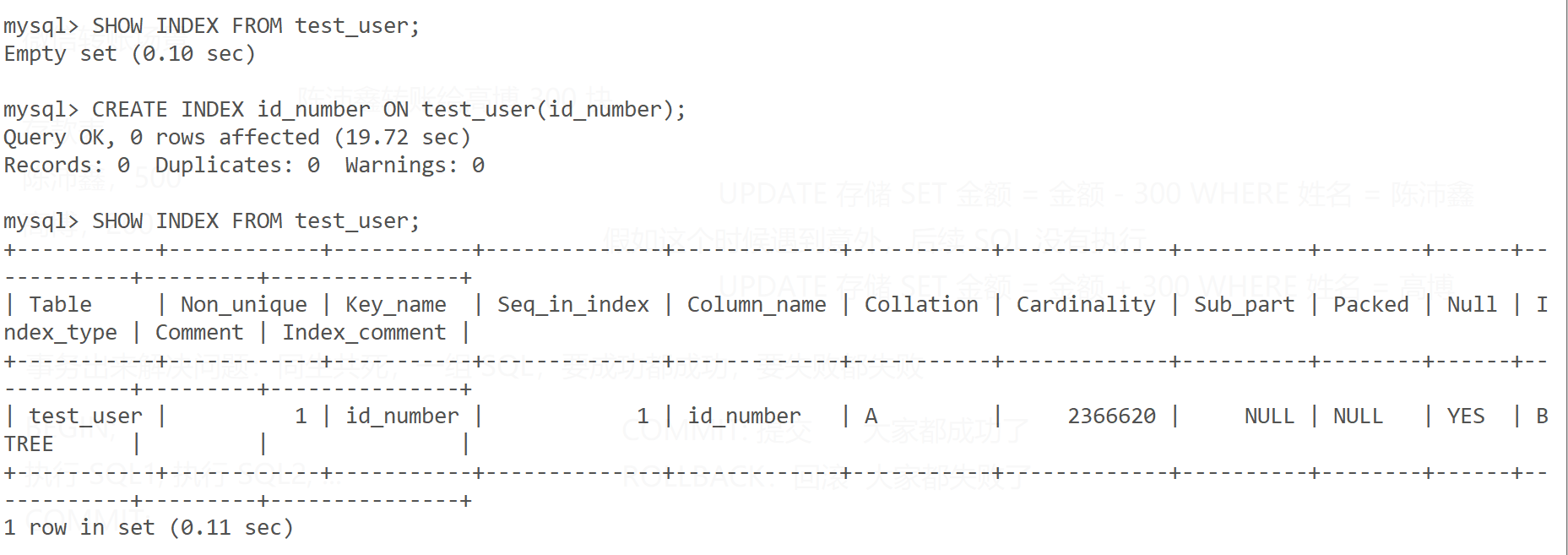
end $$

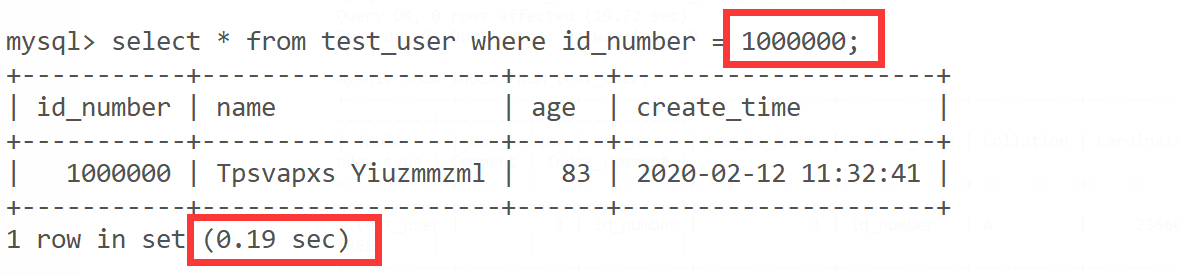
delimiter ;

-- 执行存储过程，添加8000000条用户记录

call insert\_user(1, 8000000);







# 事务（Transaction）接触



## 演示

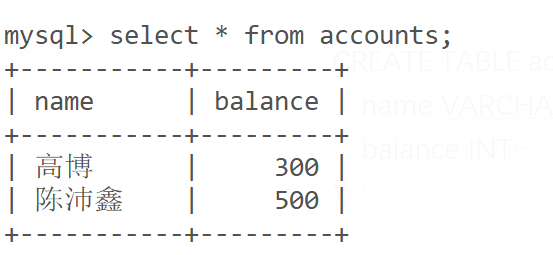
CREATE TABLE accounts (

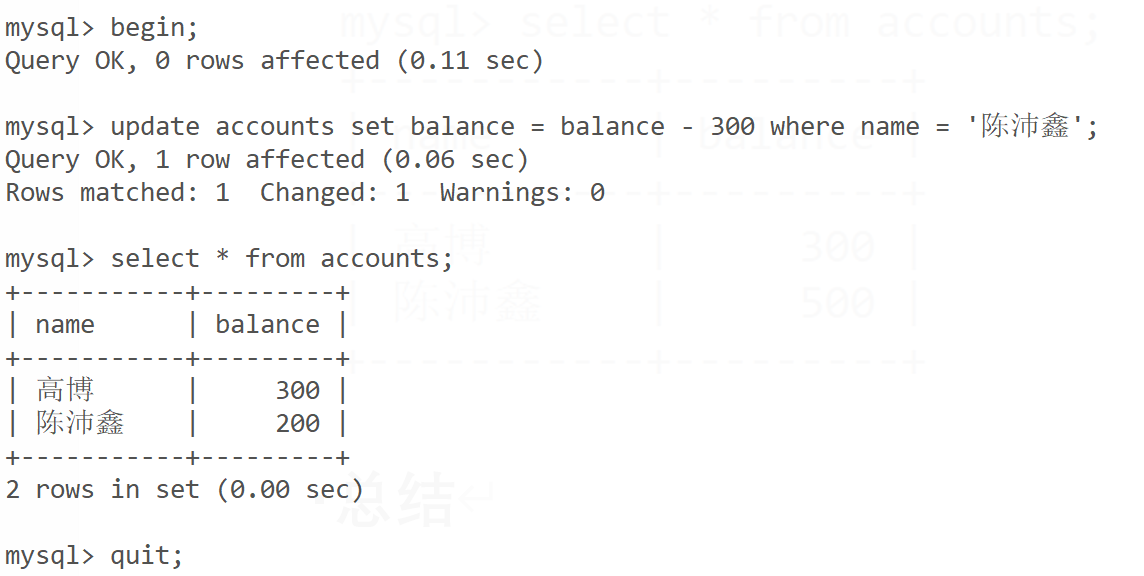
name VARCHAR(255),

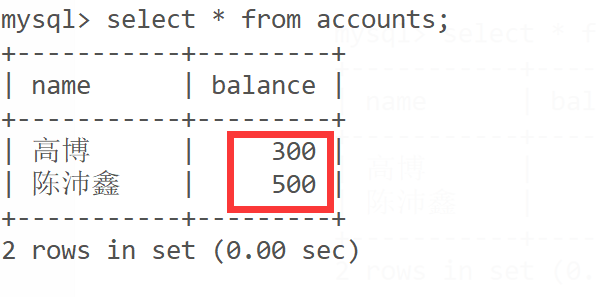
balance INT

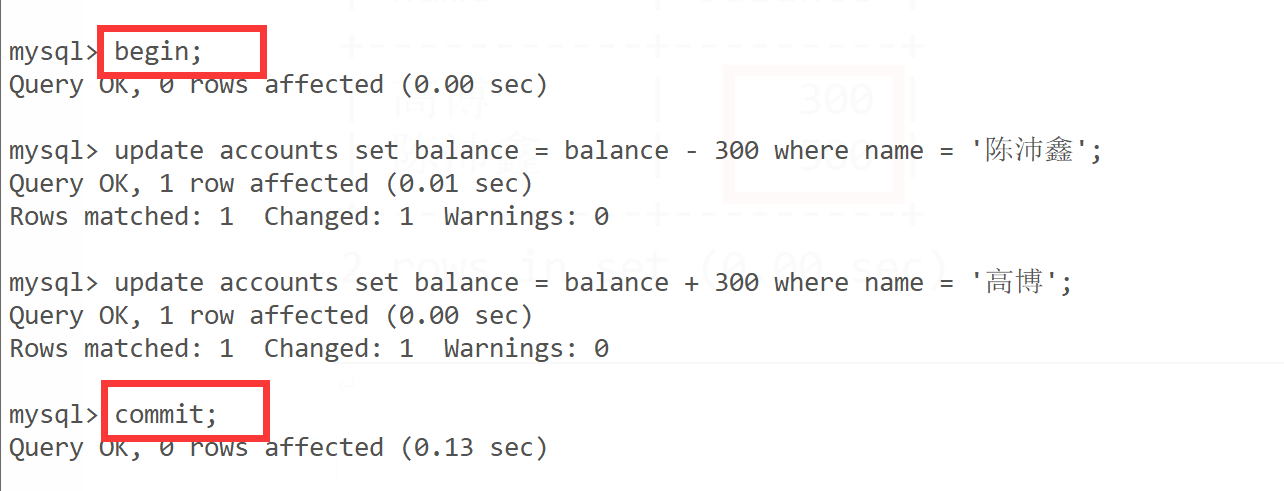
);

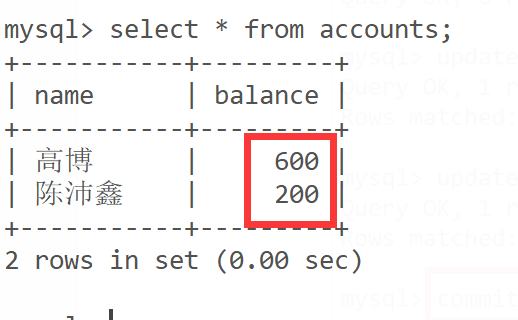
INSERT INTO accounts (name, balance) VALUES (‘高博’, 300), (‘陈沛鑫’, 500);





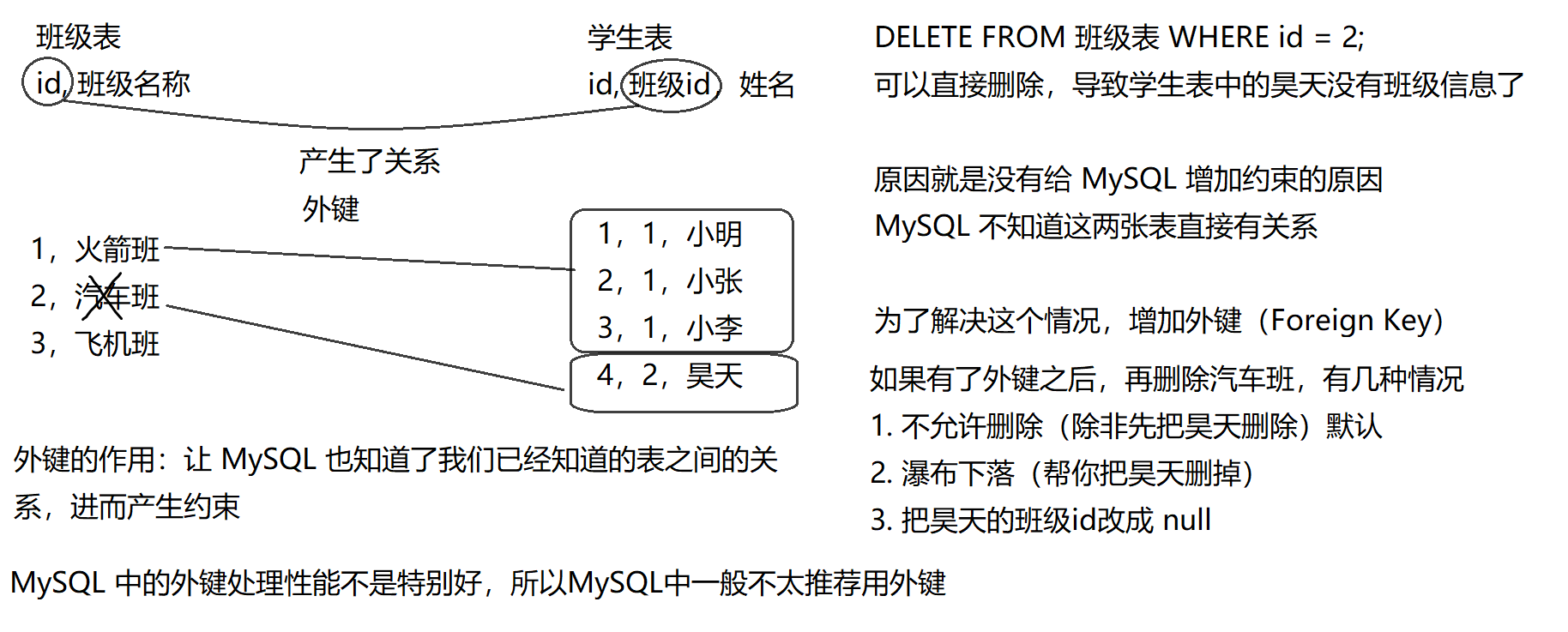






# 总结

## 外键



目标：以用为主

MySQL 的基本概念 + 基本操作（1%）：

1. C/S 模型
2. 背后实际就是文件存储
3. 可以和 Java 的类和对象对比理解
4. 安装/卸载/登录

MySQL 常见语法

DDL：库/表

MySQL 增删改（15%）

INSERT/DELETE/UPDATE

MySQL 查询（60%）

1. 基本查询
2. 复杂查询
   1. 聚合
   2. 联表
   3. 子查询
   4. 联合

JDBC 的使用（20%）

套路

索引和事务（5%）未来比重很大

练习：JDBC 结合去练习语法