**子查询与索引**

1. **子查询**
2. **where 子查询**

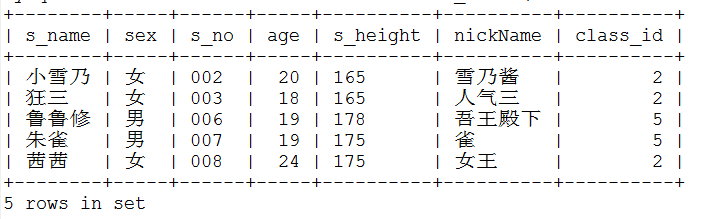
语法：

select \* from 表1 where colName [=|in](select colName from 表2 [where.....]);

例如：

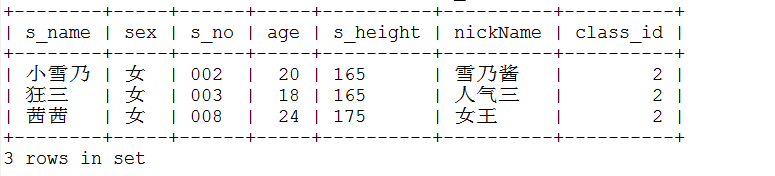
/\*in查询存在课程信息的学员\*/

select \* from students where class\_id in (select id from class);



/\*= 查询存在课程信息的学员\*/

select \* from students where class\_id = (select id from class where className='高级班');



注意：= 查询只适用于子查询结果集只有一条数据，多条数据只能用in，如下：



1. **from子查询**

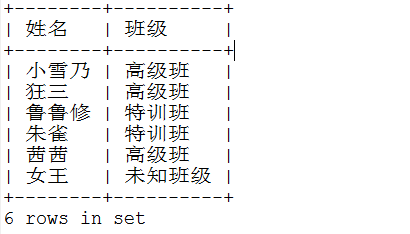
**语法：**

select \* from (select \* from 表名 [where..]) 临时表名 [where...]

例如：

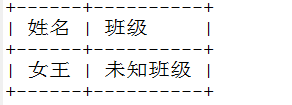
/\*将内层的查询结果集作为临时表，供外层使用\*/

select \* from (select students.s\_name 姓名,ifnull(class.className,'未知班级') 班级 from students left join class on students.class\_id = class.id) tmp;



/\*将班级为未知班级的数据查询出来\*/

select \* from (select students.s\_name 姓名,ifnull(class.className,'未知班级') 班级 from students left join class on students.class\_id = class.id) tmp where tmp.班级='未知班级'



1. **insert 子查询**

例如：

/\*将子查询的结果集(只能是一条数据)作为某个字段的数据值\*/

insert into students(s\_name,age,s\_no,sex,s\_height,nickName,class\_id) values('鲁鲁修',19,'006','男',178,'吾王殿下',(select id from class where className='特训班'));

1. **update 子查询**

例如：

/\*将class\_id等于null值的数据更新成基础班\*/

update students set class\_id=(select id from class where className='基础班') where class\_id is null;

1. **delete 子查询**

例如：

/\*删除课程为基础班的数据\*/

delete from students where class\_id=(select id from class where className='基础班');

/\*删除存在课程为基础班的数据\*/

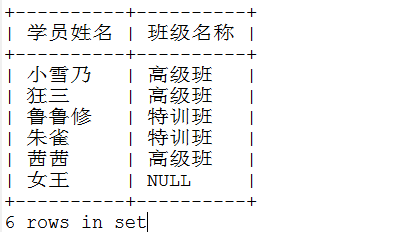
delete from students where class\_id in(select id from class where className='基础班');

1. **关联子查询**

例如：

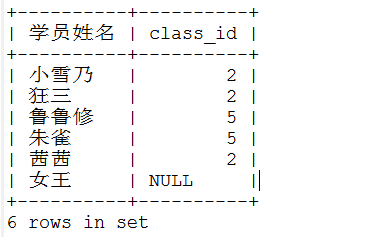
/\*查询学生的姓名与班级名字信息\*/

select s\_name 学员姓名,(select className from class c where c.id=s.class\_id) 班级名称 from students s;



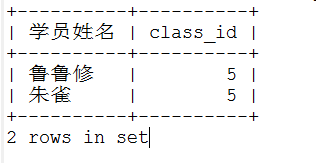
/\*查询了至少有一条满足子查询结果集的数据，就返回所有数据\*/

select s\_name 学员姓名,s.class\_id from students s where exists(select id from class where className='特训班');



/\*查询存只在于特训班的数据\*/

select s\_name 学员姓名,s.class\_id from students s where exists(select id from class where className='特训班' and id=s.class\_id);

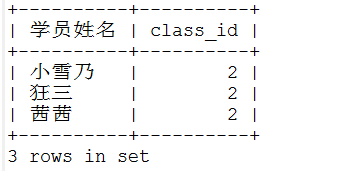


1. **两个表关联查询**

例如：

/\*两个表关联查询，查询出姓名以及class\_id都是2的数据\*/

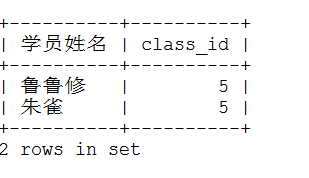
select s\_name 学员姓名,class\_id from students where class\_id=(select class\_id from students where s\_name='小雪乃');



/\*自己表跟自己表关联查询，查询出与鲁鲁修相同班级的所有数据\*/

select s\_name 学员姓名,class\_id from students s where exists(select 1 from students s1 where s\_name='鲁鲁修' and s.class\_id=s1.class\_id) order by

s\_name desc,class\_id;



1. **索引**

1、在数据库系统中是一个排序的数据结构，以协助**快速查询、更新数据库表**中表数据。

2、需要数据库开辟空间存储索引，在**插入和修改数据的时候花费较多时间**（数据变更需要建立或者重建索引）。

3、如果使用**函数或者like**索引就会失效。

**1、单字段索引**

单字段索引是最常见、最简单。

语法：Create Index 索引名 on 表名(列名)

例如：create index index\_s\_name on students(s\_name);

/\*explain分析查询语句\*/

explain select \* from students where s\_name='黑黝黝';

explain select \* from students where sex='男';

**2、唯一索引**

唯一索引用于改善性能和保证数据的完整性，数据不允许重复，效果与单字段索引一样。

语法：create unique index 索引名 on 表名(列名)

例如：

create unique index uindex\_className on test(className);

insert into test values(5,58,'高级班');

delete from test where className='高级班';

explain select \* from test;

**3、组合索引**

语法：create index 索引名 on 表名(列名1,列名2,....);

例如：

create index sex\_age\_height\_index on students(sex,age,s\_height);

create index sex\_age\_index on students(sex,age);

create index sex\_name\_index on students(sex,s\_name);

explain select \* from students where sex='女' and age<24 and s\_height>165;

explain select \* from students where sex='女' and age<24;

explain select \* from students where s\_name='小雪乃' and sex='女';

explain select \* from students;

**4、删除修改索引**

语法：drop index 索引名 on 表名

例如：drop index sex\_age\_height\_index on students;

注意：一般不会去修改索引，只会去重建索引。

**一、何时使用索引：**

1、单表数据太少，索引反而会影响速度，不要加索引

2、where后的条件，order by,group by等这样的过滤时，后面字段最后加上索引，但不是索引越多越好，要适度。

3、联合查询和子查询等多表操作关联字段加索引。

4、不应该对包含大量NULL字段加索引，尽量对字段设置not null。

5、经常要进行增、改等操作的字段，不应该设置索引，索引的新建和重建会消耗时间。

**二、关于索引对应数据类型**

1、越小的数据类型通常更好，越小的数据类型通常在磁盘、内存和cpu缓存中只需要更少的空间，处理起来就会更快。

2、简单的数据类型更好。整形数据比字符处理开销更小，因为字符串更复杂。在mysql中，日期时间应该用日期和时间类型，不要使用字符串类型存储时间。