**ORACLE基本语法**

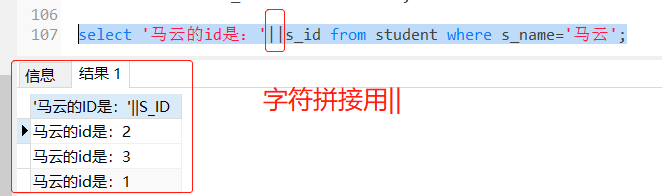
* **Oracle**在创建表的时候**表名或者列名没有用双引号括起**是不区分大小写的，用双引号括起才区分大小写。
* **Foreign key(s\_sid) references students(s\_id) [on delete cascade|| set null]**定义了一个外键约束，被student1表的s\_id约束，on delete 子句说明了当父表的行被删除的时候，子表被约束的行的处理办法。cascade是删除子表所有被父表约束的行，set null 是将被约束的行置空。
* 运算符运算顺序，先运算not，and最后or，但是可以用括号来改变运算顺序。
* And运算结果关系：有null参与运算的话，有false结果是false，无false结果是null；

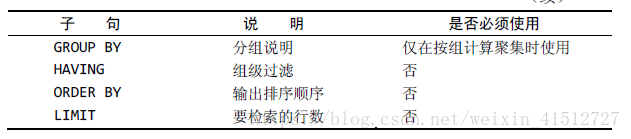
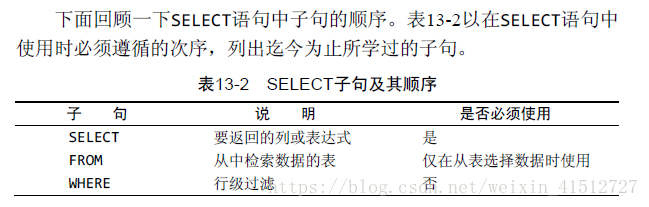
| **值** | **TRUE** | **FALSE** | **NULL** |
| --- | --- | --- | --- |
| TRUE | TRUE | FALSE | NULL |
| FALSE | FALSE | FALSE | FALSE |
| NULL | NULL | FALSE | NULL |

* Or运算结果关系：有null参与运算的话，有true结果是true，无true结果是null；

| **值** | **TRUE** | **FALSE** | **NULL** |
| --- | --- | --- | --- |
| TRUE | TRUE | TRUE | TRUE |
| FALSE | TRUE | FALSE | NULL |
| NULL | TRUE | NULL | NULL |

* Oracle当中字符拼接用 ||完成；



* Where和having的区别:where中不能有分组函数，例如avg(),sum(),max(),min()等。Having后可以有分组函数。
* Where 之后如果有group by子句，where要放在group by之前，having要放在group by之后。

**数据定义语言DDL （Data Definition Language）**

包括：create table \ modify table \ drop table \ create index \ drop index

**1.创建表语言create table**

create table student(

s\_id number(2) primary key,

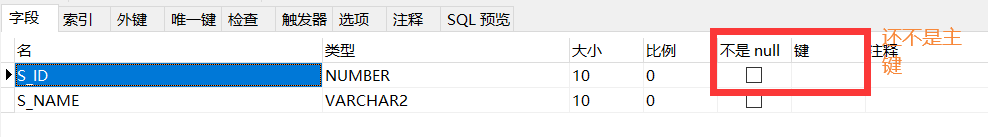
s\_name varchar2(4) ,

s\_sid number(2),

[primary key (s\_id)] 主键约束

[foreign key (s\_sid) references student1(s\_id)] 外键约束

);



create table students as (select s\_id,s\_name from student); 将另一个表查询出来的数据建一个表，不包括索引和主键。

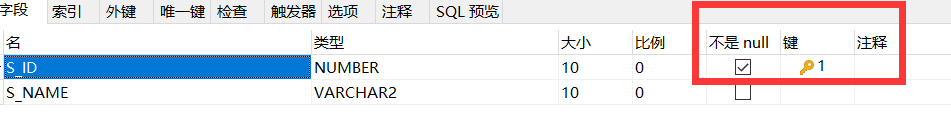
**2.更改表结构 alter table**

**Add** : alter table student add s\_age date;增加属性列

alter table student add primary key(s\_id); 增加主键

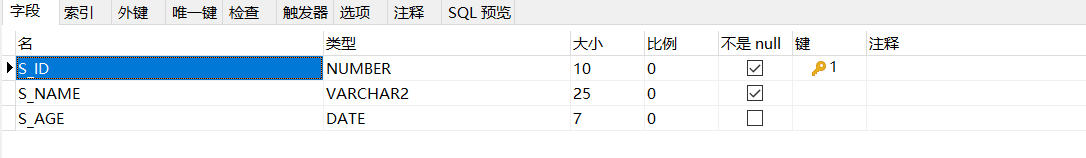
alter table students add foreign key(s\_sid) references student(s\_id) on delete cascade; 增加外键

alter table student add stuts number(1) default 1; 增加属性列并给定默认值





**Modify**: alter table student modify s\_name varchar2(25) not null;修改属性列，包括改类型，非空，约束等。改变属性的时候，要保证所有的数据满足改之后的属性，例如：改变了非空属性，则该字段不能有空数据，否则会改失败。



**Drop**: alter table studet drop column s\_age; 删除属性列

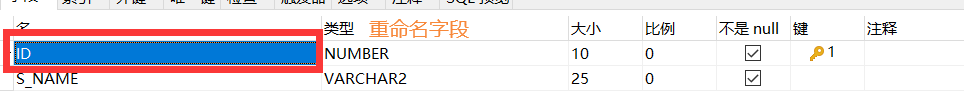
或者可以用：alter table student set unused column s\_age; 逻辑删除s\_age 列

之后再用: alter table student drop unused column; 物理删除s\_age列



**Rename**:alter table student rename column s\_id to id; 重命名属性列

alter table students rename to student; 重命名表



**3.删除表 drop table**

drop table student [cascade constraints ]; cascade 子句可不选，若不选，这张表有被别的表外键引用的时候不能删除，若选了，则会删除这个表和引用的那张表的外键约束。



**4.**

**数据操作语言 （Data Manipulation Language）**

包括 insert \ update \ delete

**1.插入数据 insert into**

insert into students(s\_id,s\_name) values(2,’丽丽’); 插入一行数据

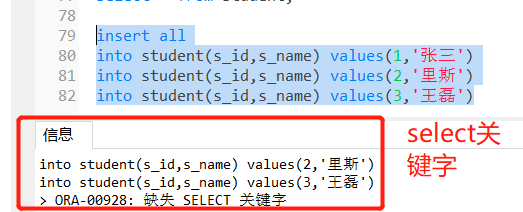
insert all

into student(s\_id,s\_name) values(1,'张三')

into student(s\_id,s\_name) values(2,'里斯')

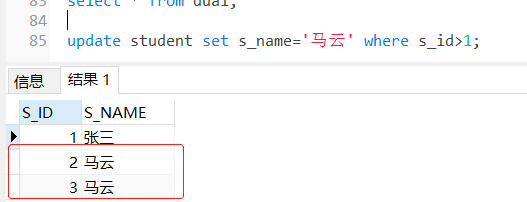
into student(s\_id,s\_name) values(3,'王磊')

select \* from dual; 插入多行数据，最后的select子句不能少，否则会报错。



**2.更新数据 update**

update student set s\_name='马云' where s\_id>1; 更新数据



**3.删除数据 delete**

delete from student where s\_id=1; 删除数据，如果这时候student表被其他表引用了，就会删除失败。如果表定义外键的时候有on delete子句，就会根据子句的 cascade属性或者 set null来处理。

**数据控制语言 （Data Control Language）**

***数据类型***

**Number（p,s）**

P表示精度，是数字中的有效数位，范围为1~38.默认为38.超过会报错。

S表示尺度，取值范围为-84~127，如果取值为正数，表示小数点往右的最大数位，超过这个数位就会被四舍五入。如果为负数，表示这个数要精确到小数点左边的指定数位。默认值为0。

小数点左边有多少位数取决于 p-s,不论s为正还是为负。

Number 1254 1254

Number(5) 1254 1254

Number(3) 1254 报错 有效数位超过3

Number(4,1) 1254.1 报错 有效数位超过4

Number(4,1) 1254.0 报错 小数点左边的数位大于 4-1

Number(4,1) 125.1 125.1

Number(4,1) 125.19 125.2 超过就四舍五入

Number(4,1) 999.99 报错 小数四舍五入之后超过了数的精度

Number(2,3) 0.100 报错 2-3=-1，小数点右边1位不能有数

Number(2,3) 0.0157 0.016 小数点右边超过3位四舍五入

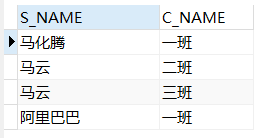
Number(3,-2) 45600 45600 3- -2=5，小数点左边有5位

Number(3,-2) 45690 45700 小数点左边前两位有数就会四舍五入

**表连接**

**1.内连接 inner join**

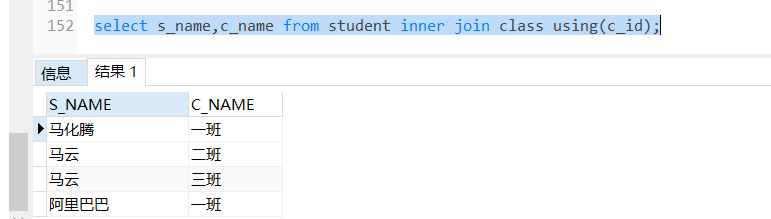
select s\_name,c\_name from student inner join class on student.s\_cid = class.c\_id ;



返回的是两个表中都满足on子句里的条件的行。

on 子句可以换成using子句：

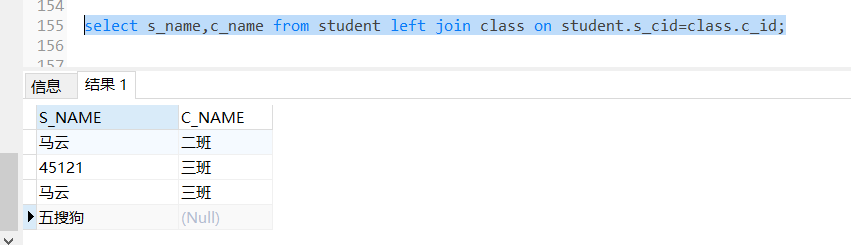
select s\_name,c\_name from student inner join class using(c\_id);



Using子句中出现的列必须要在两张表中都出现，必须要用括号括起。

**2.左连接 left join**

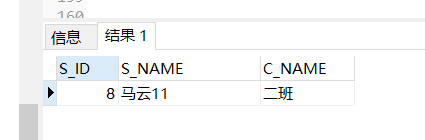
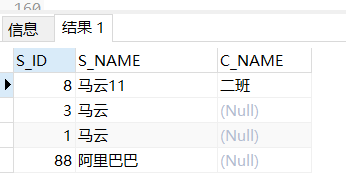
select s\_name,c\_name from student left join class on student.s\_cid = class.c\_id;



左连接返回的数据是：左表的所有数据和右表匹配的数据，右表中不匹配的行会置空然后显示。

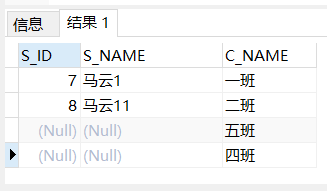
左连接也可以将on子句换成using子句，同样也是要字段在两张表中都出现

左连接中注意where条件，将限制条件写在where中，只会显示满足条件的行，将限制条件写在on中，会返回所有行，不满足条件的行会置null显示。



**3.右连接 right join**

select s\_id,s\_name,c\_name from student right join class on student.s\_cid = class.c\_id;



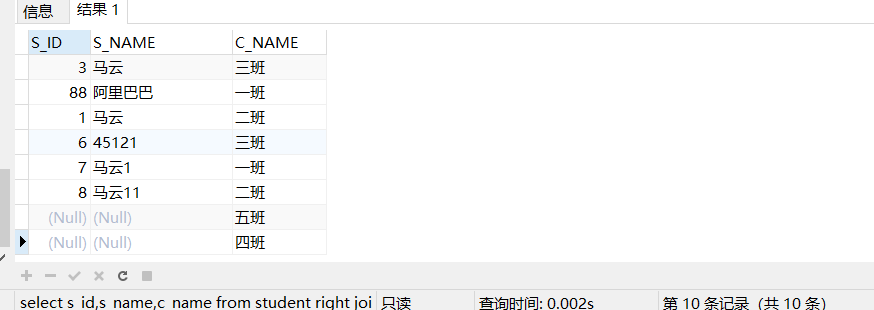
右连接返回的是右表的所有行，左表不匹配的行会置空然后展示。

同理右连接也可以有using子句和需要注意where子句里的条件。

**4.全连接 full join**

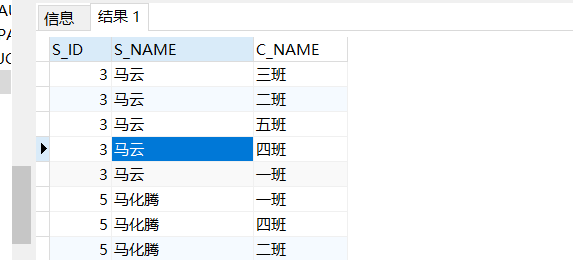
select student.s\_name,student.s\_id,class.c\_name from student full join class on student.s\_cid=class.c\_id;

全连接其实就是左连接和右连接的合集



**5.交叉连接 cross join（笛卡尔连接）**

**select s\_id,s\_name,c\_name from student cross join class order by s\_id;**



**操作符**

**1.all操作符**

操作符之前必须有运算符：=，!=,>，>=，<，<=

select s\_id,s\_name,age from student where age>all(0,2,5);查找大于5的值

select c from c where c1>all(c1,c2,c3);

all表示满足所有查询结果，等价于：

Select c from c where c1>c1 and c1>c2 and c1>c3;

多个条件之间用and连接

All 操作符和各个运算符的关系：col 和all(list) 的关系

= :col = all(list) col 与list里的每一个值完全匹配 为true

!= : col 与list里的所有值都不匹配时为true

> : col 大于list的最大值时为true

>=: col 大于等于list 的最大值时为true

< : col 小于 list的最小值时为true

<= : col 小于等于 list的最小值时为true

**2.any和some操作符**

操作符之前必须有运算符：=，!=,>，>=，<，<=

Any 和 some的行为相同，所以他们完全可以互相替换。

select s\_id,s\_name,age from student where age>any(0,2,5);查找大于0的值

select c from c where c1>any(c1,c2,c3);

any/some表示满足至少一个查询结果，等价于：

Select c from c where c1>c1 or c1>c2 or c1>c3;

多个条件之间用or连接

Any/some 操作符和各个运算符的关系：col 和all(list) 的关系

= :col = all(list) col 与list里的至少一个值匹配 为true

!= : col 与list里的至少一个值不匹配时为true

> : col 大于list的最小值时为true

>=: col 大于等于list 的最小值时为true

< : col 小于 list的最大值时为true

<= : col 小于等于 list的最大值时为true

**3.exists和not exists 操作符**

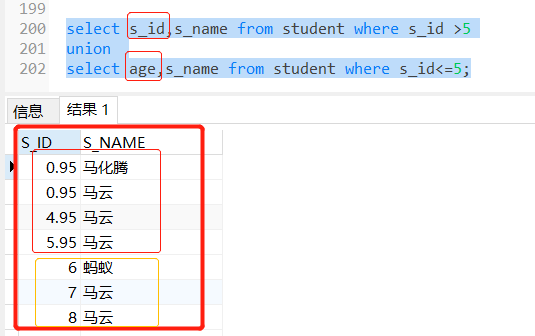
**4.union 操作符**

select s\_id,s\_name from student where s\_id >5

union

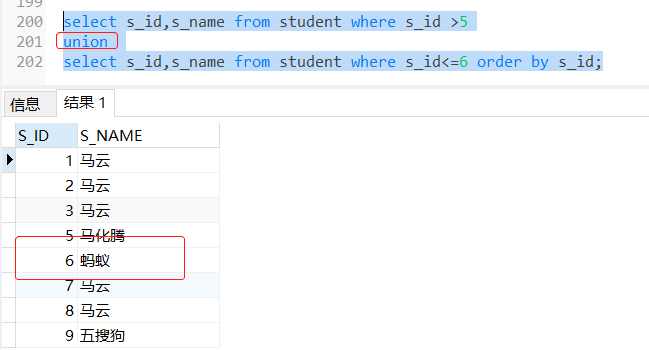
select s\_id,s\_name from student where s\_id<=5;

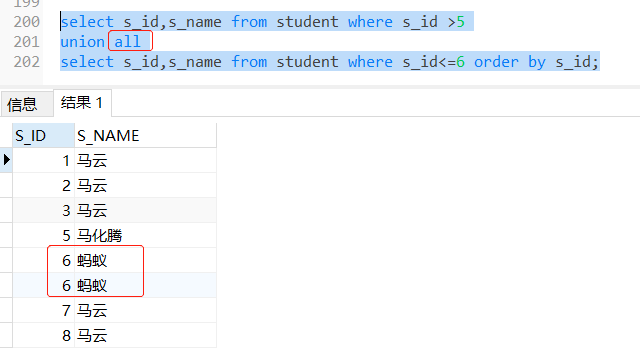
union将两个或多个select语句联合在一起，所有select语句必须返回相同的列数，对应列的类型也要相同。如果l



Union 和union all的区别：

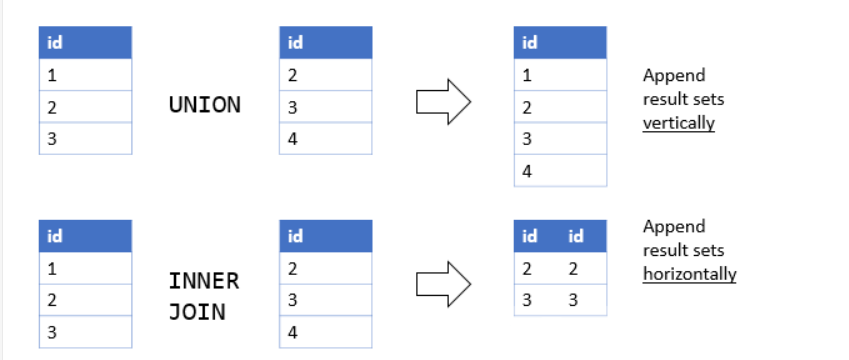
Union会将不同查询返回的相同行过滤，而union all会全部显示。





Union和join的区别，

Union将返回的结果纵向组合，而join将结果横向组合。



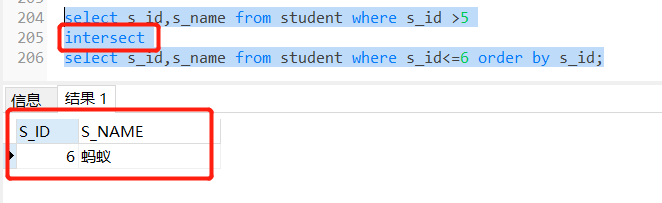
**5.intersect 操作符**

和union操作符类似，intersect操作也要求返回相同的列数，对应列也要相同类型，不同于union的是，intersect返回的是两个或以上查询相同的行。

select s\_id,s\_name from student where s\_id >5

intersect

select s\_id,s\_name from student where s\_id<=6;



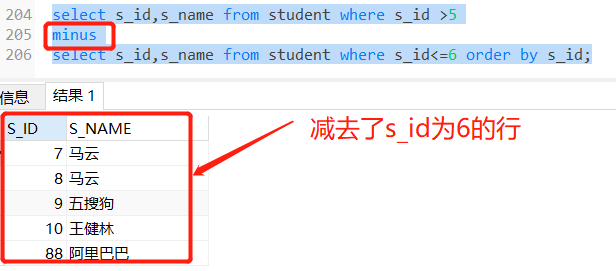
**6.minus 操作符**

和union、intersect操作符类似，minus也要求列数相同和类型相同，不同的是minus返回的是第一个查询减去第二个查询的记录，

select s\_id,s\_name from student where s\_id >5

minus

select s\_id,s\_name from student where s\_id<=6;



**函数**

分为单行函数和多行函数；

单行函数：作用于一行，返回一个值。

多行函数：作用于多行，返回一个值。

**单行函数**

**1.字符函数：**