SQL之join使用

一.基本概念

关于sql语句中的连接（join）关键字，是较为常用而又不太容易理解的关键字，下面这个例子给出了一个简单的解释 – 建表user1,user2：

table1 : create table user2(id int, user_name varchar(10), over varchar(10));

insert into user1 values(1, ‘tangseng’, ‘dtgdf’);

insert into user1 values(2, ‘sunwukong’, ‘dzsf’);

insert into user1 values(1, ‘zhubajie’, ‘jtsz’);

insert into user1 values(1, ‘shaseng’, ‘jslh’);

table2 : create table user2(id int, user_name varchar(10), over varchar(10));

insert into user2 values(1, ‘sunwukong’, ‘chengfo’);

insert into user2 values(2, ‘niumowang’, ‘chengyao’);

insert into user2 values(3, ‘jiaomowang’, ‘chengyao’);

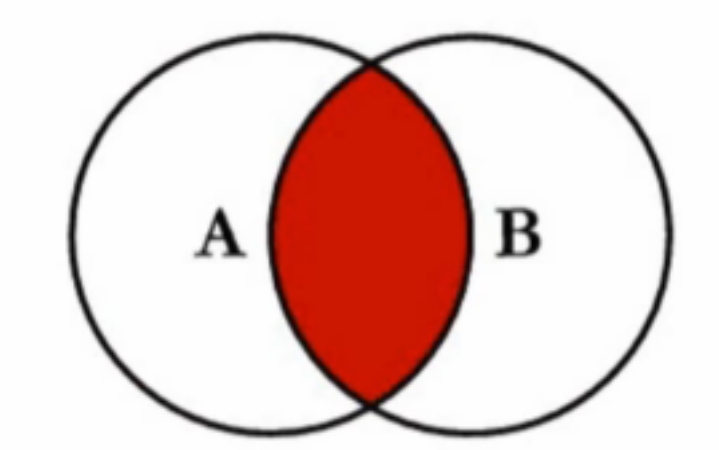
insert into user2 values(4, ‘pengmowang’, ‘chengyao’);

SQL标准中Join的类型

1. 内连接（inner join或join）

⑴.概念：内联接是基于连接谓词将两张表的列结合在一起，产生新的结果表

⑵.内连接维恩图：



(3).sql语句

select a.id, a.user_name, b.over from user1 a inner join user2 b on a.user_name=b.user_name;

结果：

```
mysql> select a.id, a.user_name, b.over from user1 a inner join user2 b on a.use
r_name=b.user_name;
+------+-----------+---------+
| id   | user_name | over    |
+------+-----------+---------+
|    2 | sunwukong | chengfo |
+------+-----------+---------+
```

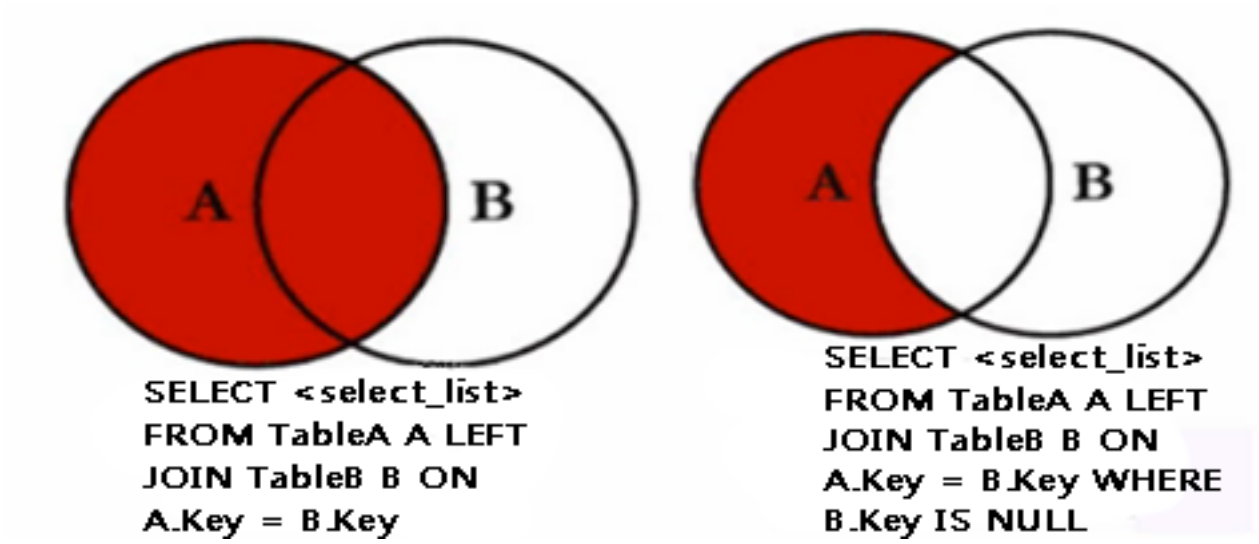
2. 外连接

外连接包括左向外联接、右向外联接或完整外部联接

a.左外连接：left join 或 left outer join

(1)概念：左向外联接的结果集包括 LEFT OUTER 子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值(null)。

(2)左外连接维恩图：



(3)sql语句：

select a.id, a.user_name, b.over from user1 a left join user2 b on a.user_name=b.user_name;

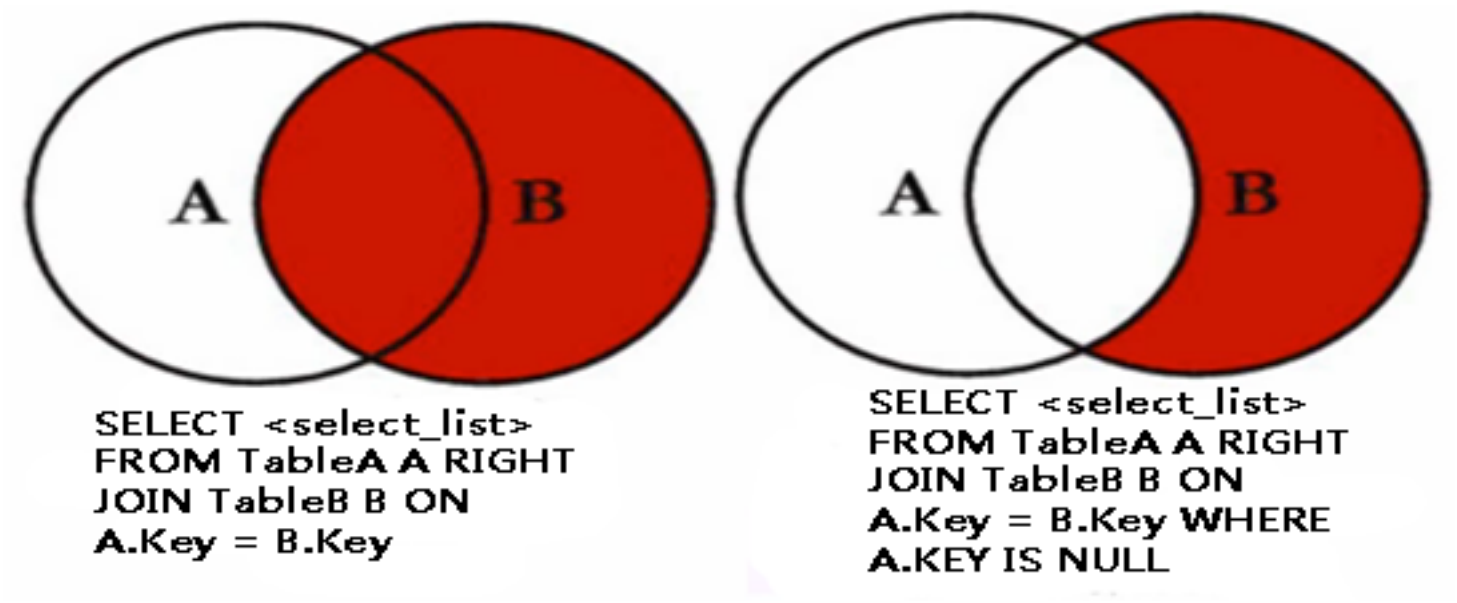
结果：

```
mysql> select a.id, a.user_name, b.over from user1 a left join user2 b on a.user
_name=b.user_name;
+------+-----------+---------+
| id   | user_name | over    |
+------+-----------+---------+
|    1 | tangsen   | NULL    |
|    2 | sunwukong | chengfo |
|    3 | zhubajie  | NULL    |
|    4 | shasen    | NULL    |
+------+-----------+---------+
```

b.右外连接：right join 或 right outer join

⑴右向外联接是左向外联接的反向联接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。

⑵右外连接维恩图：



⑶sql语句

select b.user_name, b.over, a.over from user1 a right join user2 b on a.user_name=b.user_name;

结果：

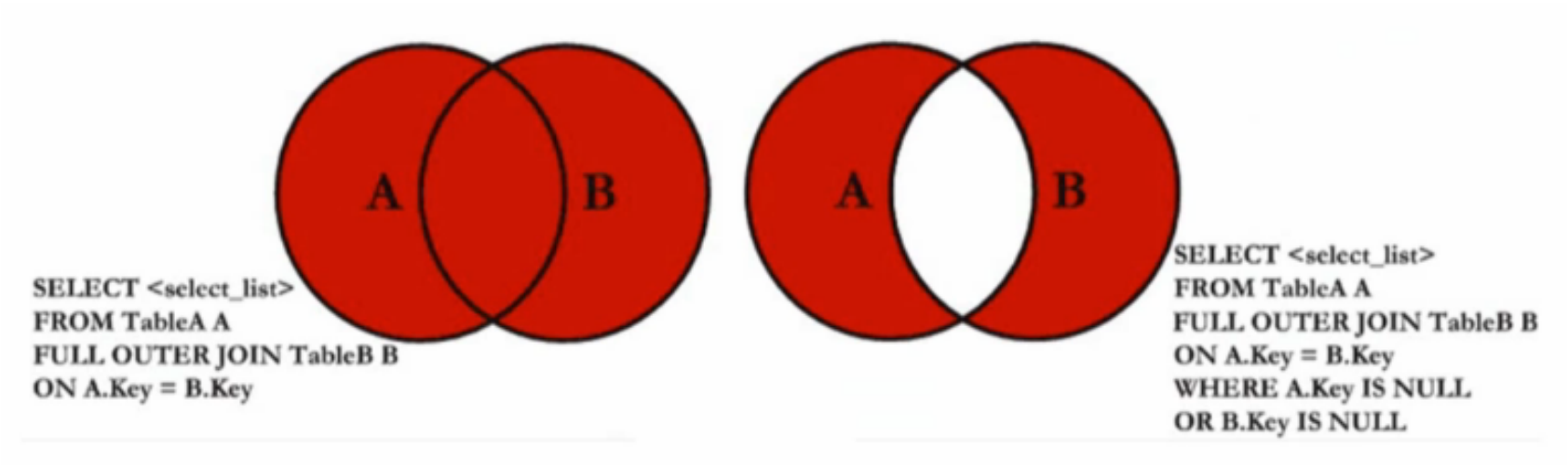
```
mysql> select b.user_name, b.over, a.over from user1 a right join user2 b on a.u
ser_name=b.user_name;
+------------+----------+------+
| user_name  | over     | over |
+------------+----------+------+
| sunwukong  | chengfo  | dzsf |
| niumowang  | chengyao | NULL |
| jiaomowang | chengyao | NULL |
| pengmowang | chengyao | NULL |
| shituowang | chengyao | NULL |
+------------+----------+------+
```

c.全外连接：full join 或 full outer join

⑴完整外部联接返回左表和右表中的所有行。当某行在另一个表中没有匹配

行时，则另一个表的选择列表列包含空值。如果表之间有匹配行，则整个结果集行包含基表的数据值。

⑵右外连接维恩图：



⑶sql语句

select a.id, a.user_name, b.over from user1 a full join user2 b on a.user_name=b.user_name

在mysql中查询全连接会报1064的错误，mysql不支持全连接查询，代替语句：

select a.user_name,a.over,b.over from user1 a left join user2 b on a.user_name = b.user_name union all select b.user_name,b.over ,a.over from user1 a right join user2 b on a.user_name = b.user_name;

结果：

```
mysql> select a.user_name,a.over,b.over from user1 a left join user2 b on a.user
_name = b.user_name union all select b.user_name,b.over ,a.over from user1 a rig
ht join user2 b on a.user_name = b.user_name;
+------------+----------+---------+
| user_name  | over     | over    |
+------------+----------+---------+
| tangsen    | gdef     | NULL    |
| sunwukong  | dzsf     | chengfo |
| zhubajie   | jtsz     | NULL    |
| shasen     | jslh     | NULL    |
| sunwukong  | chengfo  | dzsf    |
| niumowang  | chengyao | NULL    |
| jiaomowang | chengyao | NULL    |
| pengmowang | chengyao | NULL    |
| shituowang | chengyao | NULL    |
+------------+----------+---------+
```

3. 笛卡尔连接（交叉连接）

1.概念：没有 WHERE 子句的交叉联接将产生联接所涉及的表的笛卡尔积。第一个表的行数乘以第二个表的行数等于笛卡尔积结果集的大小。（user1和user2交叉连接产生4*4=16条记录）

2.交叉连接：cross join (不带条件on)

3.sql语句：

```
select a.user_name,b.user_name, a.over, b.over from user1 a cross join user2 b;
```

二.使用技巧

1. 使用join更新表

我们使用下面语句将user1表中同时存在user1表和user2表中记录的over字段更新为‘qtda’。

```
update user1 set over=’qtds’where user1.user_name in (select b.user_name from user1 a inner join user2 b on a.user_name = b.user_name);
```

这条语句在sql server, oracle中都可以正确执行，在mysql却报错，mysql不支持更新子查询的表，那么我们使用下面语句可以在做到。

```
update user1 a join (select b.user_name from user1 a join user2 b on a.user_name = b.user_name) b on a.user_name = b.user_name set a.over = ‘qtds’
```

2. 使用join优化子查询

子查询效率比较低效，使用下面语句进行查询

```
select a.user_name, a.over,(select over from user2 b where a.user_name=b.user_name) as over2 from user1 a;
```

使用join优化子查询，可以实现同样的效果

```
select a.user_name, a.over, b.over as over2 from user1 a left join user2 b on a.user_name = b.user_name;
```

3. 使用join优化聚合子查询

引入一张新表：user_kills

```
create table user_kills(user_id int, timestr varchar(20), kills int(10));
insert into user_kills values(2, ‘2015-5-12’, 20);
insert into user_kills values(2, ‘2015-5-15’, 18);
insert into user_kills values(3, ‘2015-5-11’, 16);
insert into user_kills values(3, ‘2015-5-14’, 13);
insert into user_kills values(3, ‘2015-5-16’, 17);
insert into user_kills values(4, ‘2015-5-12’, 16);
insert into user_kills values(4, ‘2015-5-10’, 13);
```

查询user1中每人对应user_kills表中kills最大的日期，使用聚合子查询语句：

```
select a.user_name,b.timestr, b.kills from user1 a join user_kills b on a
.id = b.user_id where b.kills = (select MAX(c.kills) from user_kills c where
c.user_id = b.user_id);
```

使用join优化聚合子查询（避免子查询）

```
select a.user_name, b.timestr, b.kills from user1 a join user_kills b on
a.id = b.user_id join user_kills c on c.user_id = b.user_id group by a.user_name,
b.timestr, b.kills having b.kills = max(c.kills);
```

结果：

```
mysql> select a.user_name, b.timestr, b.kills from user1 a join user_kills b on
a.id = b.user_id join user_kills c on c.user_id = b.user_id group by a.user_name
, b.timestr, b.kills having b.kills = max(c.kills);
+-----------+-----------+-------+
| user_name | timestr   | kills |
+-----------+-----------+-------+
| shasen    | 2015-5-12 |    16 |
| sunwukong | 2015-5-10 |    26 |
| zhubajie  | 2015-5-15 |    17 |
+-----------+-----------+-------+
```

4. 实现分组选择数据

要求查询出user1中每个人kills对多的前两天。

首先，我们可以通过下面语句查询出某个人kills最多的两天；

```
select a.user_name, b.timestr, b.kills from user1 a join user_kills b on
a.id = b.user_id where a.user_name =’sunwukong’ order by b.kills desc limit 2;
```

那么如何通过一个语句查询出所有人kills最多的两天的呢？看下面的语句：

```
WITH tmp AS (select a.user_name, b.timestr, b.kills, ROW_NUMBER()
over(partition by a.user_name order by b.kills) cnt from user1 a join user_kills b
on a.id = b.user_id) select * from tmp where cnt <= 2;
```

上面的语句在sql server和oracle都是支持的，但是mysql不支持分组排序函数ROW_NUMBER(),下面提供一种替代方法：

```
select d.user_name,c.timestr, kills from (select user_id, timestr, kills, (select
count(*) from user_kills b where b.user_id = a.user_id and a.kills <= b.kills) as
cnt from user_kills a group by user_id, timestr, kills) c join user1 d on c.user_id =
d.id where cnt <= 2;
```

结果：

```
mysql> select d.user_name,c.timestr, kills from (select user_id, timestr, kills,
(select count(*) from user_kills b where b.user_id = a.user_id and a.kills <= b.
kills) as cnt from user_kills a group by user_id, timestr, kills) c join user1 d
 on c.user_id = d.id where cnt <= 2;
+-----------+-----------+-------+
| user_name | timestr   | kills |
+-----------+-----------+-------+
| sunwukong | 2015-5-10 |    26 |
| sunwukong | 2015-5-12 |    20 |
| zhubajie  | 2015-5-11 |    16 |
| zhubajie  | 2015-5-15 |    17 |
| shasen    | 2015-5-10 |    13 |
| shasen    | 2015-5-12 |    16 |
+-----------+-----------+-------+
```