SQL - 内连接与外连接

SQL 内连接 外连接 交叉连接

连接查询在关系型数据库中经常用到，是多表联合查询的基础。
主要包含：**内连接**，**外连接**，**交叉连接**。

内连接

内连接又分为**等值连接**，**不等值连接**，**自然连接**。

> 连接查询中使用的比较运算符有:=, >, <, <>, >=, <=, !>, !<

等值连接

等值连接使用”=”来进行比较运算。
请看下面的例子：
T_student表

student_id	student_name	class_id
1	aaa	15
2	bbb	16
3	ccc	17

T_class表

class_id	class_name
15	五班
16	六班
17	七班
18	八班

如果需要查出一下内容：

> 每个学生所对应的班级名称

此时就可用以下sql语句：

```
SELECT * FROM
    T_student ts,T_class tc
WHERE
    ts.class_id=tc.class_id
```

或者：

```
SELECT * FROM
    T_student ts inner join T_class tc
ON
    ts.class_id = tc.class_id
```

查询结果如下：

student_id	student_name	class_id	class_id	class_name
1	aaa	15	15	五班
2	bbb	16	16	六班
3	ccc	17	17	七班

结论如下：

> 等值连接：
>
> > 若要连接表t1和t2，比较条件为t1.a=t2.a，那么数据库会用t1中a列的所有元素逐个和t2中的a进行比较，如果相等，则输出该行。

至于数据库内部如何实现，我们暂且不去深究。

不等值连接

内连接中，不使用”=”作为比较运算符，就叫不等值连接。
如果需要查询以下内容：

> T_student表和T_class表中，class_id字段不相等的所有组合

此时就可用以下sql语句：

```
SELECT * FROM
    T_student ts,T_class tc
WHERE
    ts.class_id <> tc.class_id
```

或者：

```
SELECT * FROM
    T_student ts inner join T_class tc
ON
    ts.class_id <> tc.class_id
```

查询结果如下：

student_id	student_name	class_id	class_id	class_name
2	bbb	16	15	六班
3	ccc	17	15	七班
1	aaa	15	16	五班
3	ccc	17	16	七班

1	aaa	15	17	五班
2	bbb	16	17	六班
1	aaa	15	18	五班
2	bbb	16	18	六班
3	ccc	17	18	七班

结论如下：

> 不等值连接
>
> > 若要连接表t1和t2，比较条件为t1.a不等于t2.a，那么数据库会用t1中a列的所有元素逐个和t2中的a进行比较，如果不相等，则输出该行。

自然连接

自然连接是一种特殊的等值连接，和等值连接差不多，区别在于：

> 自然连接会去掉重复的列；
> 自然连接要求比较的两个列属性必须相同，等值连接则不需要；

如果需要查出以下内容：

> 每个学生所对应的班级名称

sql语句和等值连接差不多：

```
SELECT ts.*,tc.class_name FROM
    T_student ts inner join T_class tc
ON
    ts.class_id = tc.class_id
```

查询结果如下：

student_id	student_name	class_id	class_name
1	aaa	15	五班

2	bbb	16	六班
3	ccc	17	七班

由此可见：

> 自然连接相当于在等值连接的基础上，加了显示的限定条件，从而实现列去重

外连接

外连接又分为**左连接**，**右连接**，**全连接**。

左连接

左连接以左表为基础，显示左表中的所有记录（显示的记录条数=左表中记录的条数）。再用左表中的指定列来和右表中的指定列比较，满足则输出值，不满足则输出`NULL`
如果需要查出以下内容：

> 每个学生所对应的班级名称

此时就可用以下sql语句：

```
SELECT * FROM
    T_student ts left join T_class tc
ON
    ts.class_id = tc.class_id
```

查询结果如下：

student_id	student_name	class_id	class_id	class_name
1	aaa	15	15	五班
2	bbb	16	16	六班
3	ccc	17	17	七班

但如果T_student表中增加一条字段：

student_id	student_name	class_id

4	ddd	20

此时再用上一条sql语句查询，则结果变为：

student_id	student_name	class_id	class_id	class_name
1	aaa	15	15	五班
2	bbb	16	16	六班
3	ccc	17	17	七班
4	ddd	20	`NULL`	`NULL`

结论如下：

> 左连接
>
> > 以左表为基础，显示的记录条数=左表中记录条数。如果左表中选出的字段符合条件则显示，否则显示 `NULL`

右连接

与左连接恰恰相反。以右表为基础，显示记录的条数=右表中记录条数，然后和左表中的字段比较，符合条件则输出，否则输出 `NULL` 。
使用以下sql语句：

```
SELECT * FROM
    T_student ts right join T_class tc
ON
    ts.class_id = tc.class_id
```

查询结果为：

student_id	student_name	class_id	class_id	class_name
1	aaa	15	15	五班
2	bbb	16	16	六班
3	ccc	17	17	七班

NULL	NULL	NULL	18	八班

结论如下：

> 右连接
>
> > 以右表为基础，显示记录的条数=右表中记录条数。如果左表中选出的字段符合条件则显示，否则显示 NULL

全连接

全连接类似于左连接和右连接的综合，显示记录的条数=指定比较字段在两个表中的不同种类数。对于空余字段则显示 NULL 。
使用以下sql语句：

```
SELECT * FROM
    T_student ts full join T_class tc
ON
    ts.class_id = tc.class_id
```

查询结果为：

student_id	student_name	class_id	class_id	class_name
1	aaa	15	15	五班
2	bbb	16	16	六班
3	ccc	17	17	七班
NULL	NULL	NULL	18	八班

但如果T_student表中增加一条字段：

student_id	student_name	class_id
4	ddd	20

此时再用上一条sql语句查询，则结果变为：

student_id	student_name	class_id	class_id	class_name

1	aaa	15	15	五班
2	bbb	16	16	六班
3	ccc	17	17	七班
4	ddd	20	`NULL`	`NULL`
`NULL`	`NULL`	`NULL`	18	八班

结论如下：

> 全连接
>
> > 指定比较字段在两个表中的不同种类数。对于空余字段则显示 `NULL` 。

交叉连接

交叉连接很简单，就是两个表做 `笛卡尔积` 。
如果不加 `where` 做选择比较，那么显示的记录行数就是两个表行数的乘积。
使用以下sql语句：

```
SELECT * FROM
    T_student cross join T_class
```

或者：

```
SELECT * FROM
    T_student,T_class
```

查询结果为：

student_id	student_name	class_id	class_id	class_name
1	aaa	15	15	五班
2	bbb	16	15	六班
3	ccc	17	15	七班

1	aaa	15	16	五班
2	bbb	16	16	六班
3	ccc	17	16	七班
1	aaa	15	17	五班
2	bbb	16	17	六班
3	ccc	17	17	七班
1	aaa	15	18	五班
2	bbb	16	18	六班
3	ccc	17	18	七班

如果是有`where`进行选择，那就先进行笛卡尔积，然后在笛卡尔积的结果中进行选择（效率很差）。
结论如下：

> 交叉连接
>
> > 没有 `where` 进行选择，结果为两个表的笛卡尔积
> > 有 `where` 进行选择，先做笛卡尔积，在笛卡尔积的结果中进行选择

- Github：@crazyacking，Devin
- 邮箱：crazyacking@gmail.com
