联结表

什么是联结?

联结 (join) 是利用SQL中的select语句的一个重要操作。关系数据库中的表中包含各类**关系**, 各个表之间通过**外键**相互关联。

每个表都有一个**主键**, 该主键可以被其他表用来作外键。

> 外键:外键是某个表中的某些字段, 包含其他表中的主键, 定义了表之间的关系。

这样, 可以看出关系数据库的优点:

- 可伸缩性好:能适应不断增加的工作量。

为什么使用联结?

将数据分散成多个表有方便之处, 如可伸缩性好。但是也有些许不好之处。

数据存储在多个表中, 检索的语句要稍微麻烦一点。使用联结机制, 在一条select语句中关联表。联结在运行时关联表中的正确的行。

> 维护引用完整性: 使用关系表时, 如果products表中插入非法的供应商ID (即没有在vendors表中出现) 的供应商产品, 则这些产品是不可访问的, 因为它们没有关联到某个供应商。

> 为了防止这种情况发生, 可指示MySQL只允许在products表中供应商ID字段出现合法值 (即vendors中含有的ID) 。这就是维护引用完整性。这通过在表中定义中指定主键和外键来实现。

创建联结

```
select vend_name,prod_name,prod_price from vendors,products
where vendors.vend_id=products.vend_id
order by vend_name,prod_name;    -- 优先以vend_name排序,vend_name相同以prod_name排序
```

select 语句查询到不是一个表中的字段, from子句跟着两个表, where子句指示MySQL匹配两个表中的vend_id。该SQL语句先建立联结, 然后返回这两个表中对应的字段。(可以理解为将两个表合为一个虚拟的表, 这个表按照vend_id一一对应起来)

在select语句联结几个表时, 相应的关系是在运行时构造的。数据库表的定义中不存在能指示MySQL如何对表进行联结的东西。在两个联结时, 第一个表中的每一行都会与第二个表中的每一行配对。where子句是过滤条件, 它只包含给定条件的行。如果没有where的过滤条件, 第一个表中的每一个行都会与第二个表中的每一个行进行配对, 而不管它们逻辑上是否可以在一起。如,

```
select vend_name,prod_name,prod_price from vendors,products
order by vend_name,prod_name;
```

没有联结条件（where的过滤条件）返回的结果为**笛卡儿积**的结果。检索的数目将是第一个表中的行数乘以第二个表中的行数。

介绍各类join前，先介绍一下笛卡尔积结果。现在有两个表：

id	name
1	jack
2	tony
3	pony

另一个表：

id	address
1	NY
2	DT
3	LD

使用`select * from table1,table2`产生笛卡尔积结果：

id	name	id	address
1	jack	1	NY
2	tony	1	NY
3	pony	1	NY
1	jack	2	DT
...	...	...	...

以下介绍的各类join，inner join,outer join, left outer join, right outer join等都是从笛卡尔积中选取满足join条件的记录（行）。

内部联结

内部join又称为等值join。用`inner join`指定表的join关系，on指明联结条件。如果不加联结条件，检索结果为笛卡尔积结果。加上on的join条件，但是会有重复行。如下：

```
select *from table1,table2 where table1.id=table2.id;
```

检索结果如下：

id	name	id	address

id	name	id	address
1	jack	1	NY
2	tony	2	DT
3	pony	3	LD

之前的两个表中的相等测试`where table1.cust_id=table2.cust_id`，这种称为内部联结。其实，对于这种联结可以使用稍微不同的语法来明确指定join的类型。如

```
select vend_name,prod_name,prod_price
from vendors INNER JOIN products
ON vendors.vend_id=products.vend_id;
```

此时，from子句中指示两个的join关系，以inner join的方式。同时，使用on子句而不是where子句。

自然连接（natural join）

自然连接就是两个符合直觉思维的一种连接。比如

id	name
1	jack
2	tony
3	pony

和

id	addr	email
1	LA	123@qq.com
2	NY	...

使用`select * from table1 natural join table2;`,检索结果如下：

id	name	addr	email
1	jack	LA	123@qq.com
2	tony	NY	...

可以看出，table1中的第三行并没有没join。两个表如果没有相同记录的字段，则检索结果为空。例如，table2中的id分别为4，5时，自然join的结果为空。同时可以看出了自然join和inner join其实非常类似，都是基于等值测试。区别就是inner join的等值测试需要显示指出，并且natual join自动重叠相同的列。

自然join基于等值，如果R和S具有相同的属性组Y，则自然join可以记为，

```
R*S={trnts|tr∈R ^ ts∈S ^ tr[Y]=ts[Y]}
```

join多个表

语法如下，

```
select prod_name,vend_name,prod_price,quantity
from orderitems,products,vendors    -- join三个表
where products.vend_id=vendors.vend_id and orderitems.prod_id=products.prod_id and
order_num=20005;
```

> join对性能影响较大。

join表可以代替嵌套select子句。如，

```
select cust_name,cust_contact from customers
where cust_id in (select cust_id from orders
                    where order_nums in (
                        select order_num from orderitems
                        where prod_id ='TNT2'
                    )
                );
-- 替换为
select cust_name,cust_contact
from customers,orders,orderitems
where customers.cust_id=orders.cust_id
and orderitems.order_num=orders.order_num
and prod_id='TNT2';
```

join表的用法，join三个表，然后通过where的三个过滤条件，返回检索结果。

高级join

使用表别名

sql可以给字段和计算字段起别名（导出字段）。同样可以给表起别名。

```
select cust_name,cust_contact
from customers as c,orders as o,orderitems as oi
where c.cust_id=o.cust_id
and oi.order_num=o.order_num
and prod_id='TNT2';
```

自联结 self-join

自联结也是一种inner join。

```
select prod_id,prod_name
from products
where wend_id=(select vend_id from products where prod_id='DTNTR');

-- 使用join的版本

select p1.prod_id,p1.prod_name
from products as p1, products as p2
where p1.vend_id=p2.vend_id
and p2.prod_id='DTNTR'; -- 不可以为p1.prod_id='DTNTR';
```

可以借着self-join给出几个注意：

- self-join的两个表是完全相同的，因此上面的select语句一定要使用完全限定用法，显示指明返回哪个表的哪个字段。不是self-join联结的多个表也有这种歧义可能，将多个表join成一个大表，最好使用完全限定用法。这也就是下面介绍的自然join。

自然join

进行join的时候，会返回所有数据，这些字段中会有重复的。**自然联结**可以排除多次出现的情况，使得每个字段只返回一次。

> natural join 关键字在MySQL中不再适用。自然join的工作完全交给了user。

```
select c.*,o.order_num,o.order_date,oi.prod_id,oi.quantity,oi.item_price
from customer as c,orders as o, orderitems as oi
where c.cust_id=o.cust_id
and oi.order_num=o.order_num
and prod_id='FB';
```

通配符只对第一个表使用，其他字段显示指出（因为两个表有重复字段）。事实上，几乎用到的都是自然join。因此，MySQL中希望将natural join的工作交给用户完成。

外部join

外部join将inner join的没有关联的记录也会显示出来。如有

id	name
1	jack
2	tony
3	pony

和

id	addr
3	NY
5	DT

inner join(给定join条件`table1.id=table2.id`的前提)只会返回：

id	name	id	addr
3	pony	3	NY

而outer join可以保留未匹配关联的记录 (行) , 使用`select * from table1 left outer join table2 on table1.id=table2.id`, 返回：

id	name	id	addr
1	jack	NULL	NULL
2	tony	NULL	NULL
3	pony	3	NY

为了理解left outer join和right outer join的区别, 看一下使用`select * from table1 right outer join table2 on table1.id=table2.id`, 返回：

id	name	id	addr
3	pony	3	NY
NULL	NULL	5	DT

可以看出left,right outer join的区别在于以左侧表为基准还是右侧。例如`table1 right outer join on table1.id=table2.id`, 将会以table2为基准, 如果table1中没有记录和table2中的记录 (行) 匹配, 则以NULL值返回。

> left,right outer join 不可以忽略on联结条件。

full outer join

full outer join或full join, 是左右join的结合版本。

> MySQL未实现full outer join, 但是很容易通过union来实现

```
select *from table1 left outer join table2 on table1.id=table2.id
union
select *from table2 right outer join table1 on table1.id=table2.id;
```

检索结果如下：

id	name	id	addr
1	jack	NULL	NULL
2	tony	NULL	NULL
3	pony	3	NY
NULL	NULL	5	DT

使用带聚集函数的join

aggregate function用来汇总数据，当然可以汇总多个表的数据。即联合join一起使用即可。

```
select customers.cust_name,customers.cust_id,count(orders.order_num) as num_ord
from customers inner join orders
on customers.cust_id =orders.cust_id
group by customers.cust_id;
```

该语句首先使用inner join将两个表关联，接着group by进行分组，随后count对每个组进行计数。