查询的过滤条件

- 如何限定结果集是最为关键的因素
- 也是使用SQL各种技巧的判断因素

过滤条件的含义

- Where子句和having子句
  - Join过滤条件
  - Select过滤条件

```
select .....
   from t1
     inner join t2
       on t1.join1 = t2.join2
   where ...
```

如果存在t1.c2>100这个条件放哪里？

过滤条件的含义

- 假设有一个参数表 p（pname，ptype，pvalue）无论ptype定义了什么参数属性，pvalue都是用字符串表示（请记住这是一个错误的用法）

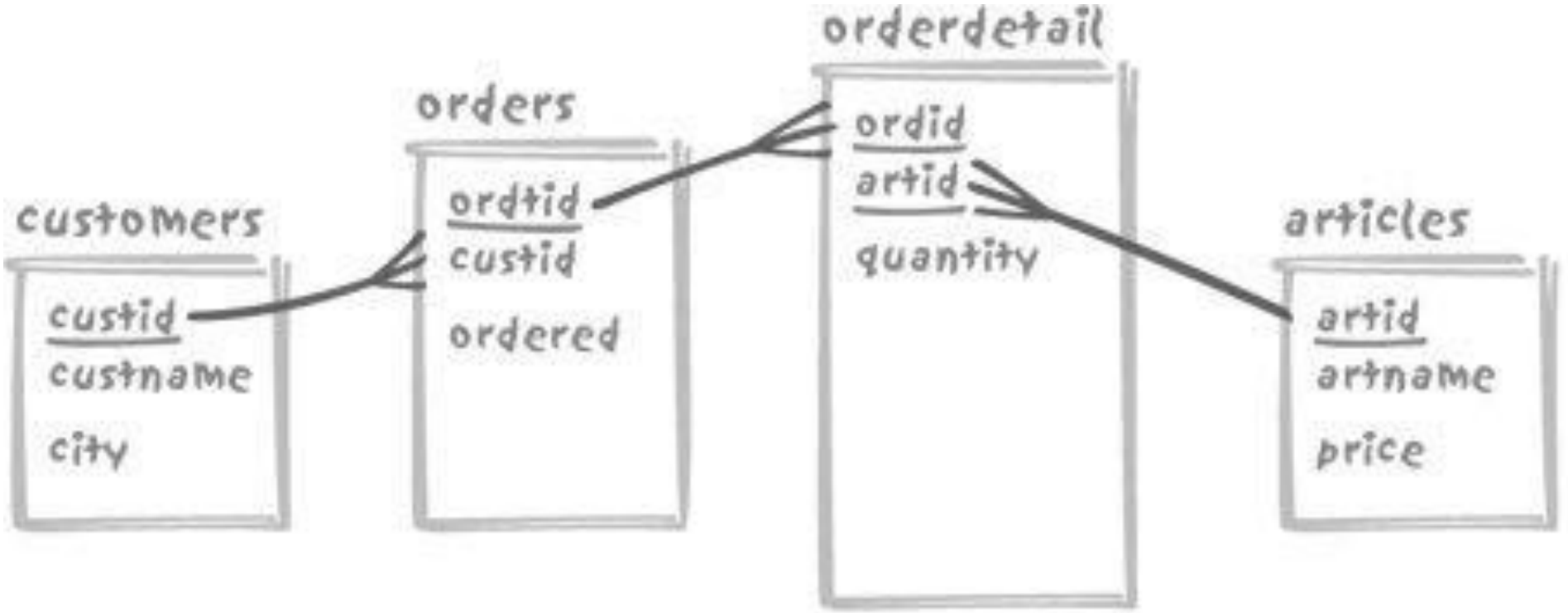
```
select * from p
   where pname like '%size'
    and ptype = 'NUMBER'
    and int(pvalue) > 1000
```

过滤条件的好坏

- 最终需要的数据是什么，来自哪些表
- 哪些输入值会传递到DBMS引擎
- 能过滤掉不想要的数据的条件有哪些

- 高效过滤条件是查询的主要驱动力

来，去买BMW.....


orderdetail
ordid
artid
quantity
orders
ordtid
custid
ordered
customers
custid
custname
city
articles
artid
artname
price

找出最近6个月住在nanjing，购买了BMW的所有客户

```
select distinct  c.custname
   from customers c
      join orders o
       on o.custid = c.custid
      join orderdetail od
       on od.ordid = o.ordid
      join articles a
       on a.artid = od.artid
   where c.city = ‘Nanjing'
    and a.artname = ‘BMW'
    and o.ordered >= somefunc /*函数，返回六个月前的具体日期*/
```

```
join orders o
      on o.custid = c.custid
      and o.ordered >= somefunc
```

古老的自然连接方式

```
select distinct c.custname
   from customers c,
       orders o,
       orderdetail od,
       articles a
   where c.city = ‘Nanjing'
    and c.custid = o.custid
    and o.ordid = od.ordid
    and od.artid = a.artid
    and a.artname = ‘BMW'
    and o.ordered >= somefunc
```

进一步

- 避免在最高层distinct应该是一条基本规则
  - 发现重复数据容易，发现不准确的连接难
  - 发现结果不正确就更难了

摆脱distinct的方法

```
select c.custname
   from customers c
   where c.city = 'Nanjing'
    and exists (select null
                from orders o,
                     orderdetail od,
                     articles a
               where a.artname = 'BMW'
                  and a.artid = od.artid
                  and od.ordid = o.ordid
                  and o.custid = c.custid
                  and o.ordered >= somefunc )
```

客户在Nanjing市，
而且满足Exists存在性测试
即在最近六个月买了BMW

Exists嵌套子查询和外层
select关系非常密切

非关联子查询

```
select custname
from customers
where city = ‘Nanjing'
    and custid in (select o.custid
                   from orders o,
                        orderdetail od,
                        articles a
                   where a.artname = ‘BMW'
                     and a.artid = od.artid
                     and od.ordid = o.ordid
                     and o.ordered >= somefunc)
```

关联子查询中，orders表中custid字段要有索引，而对非关联子查询则不需要，因为要用到的索引是customers的主键索引

内层查询不再依赖外层查询，只需要执行一次

还可以进一步嵌套

```
select custname
  from customers
  where city = 'NanJing'
   and custid in
          (select o.custid
           from orders o
           where o.ordered >= somefunc
            and exists (select null
                  from orderdetail od,
                     articles a
                  where a.artname = 'BMW'
                   and a.artid = od.artid
                   and od.ordid = o.ordid))
```

```
select custname
  from customers
  where city = 'NanJing'
   and custid in
          (select custid
           from orders
           where ordered >= somefunc
            and ordid in (select od.ordid
                  from orderdetail od,
                     articles a
                  where a.artname = 'BMW'
                   and a.artid = od.artid)
```

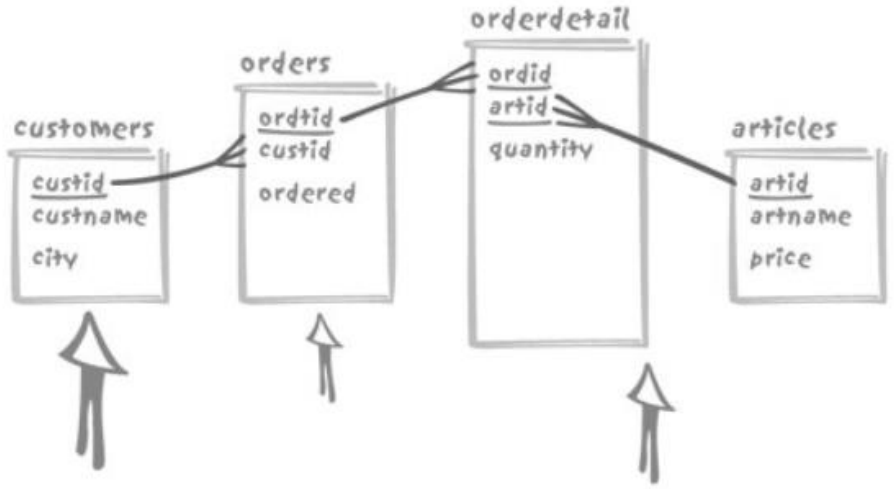
还没看够不同的SQL写法嘛?

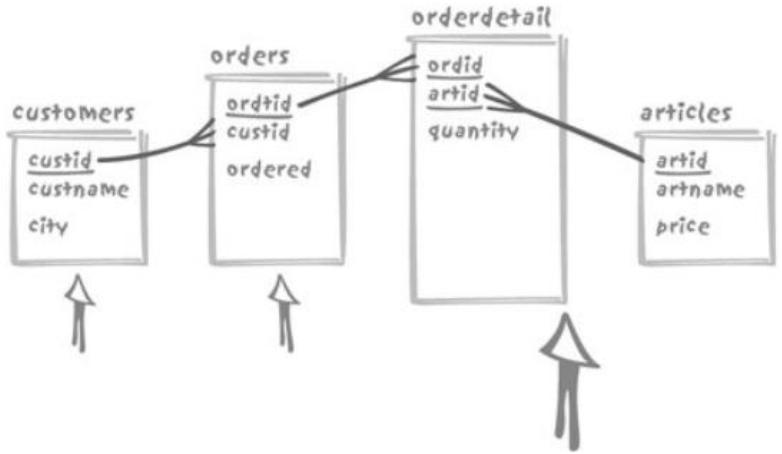
- 对于很多数据库来说，非关联子查询还可以写成from子句的内嵌视图

```
select custname
   from customers
   where city = 'Nanjing'
    and custid in
            (select o.custid
             from orders o,
                 (select distinct od.ordid
                  from orderdetail od,
                      articles a
                  where a.artname = 'BMW'
                   and a.artid = od.artid) x
            where o.ordered >= somefunc
             and x.ordid = o.ordid)
```

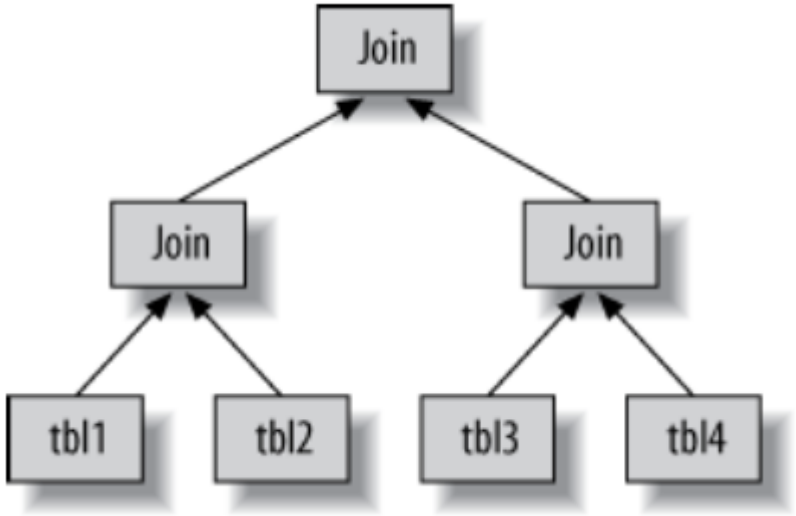
跟我去买BMW例子的总结

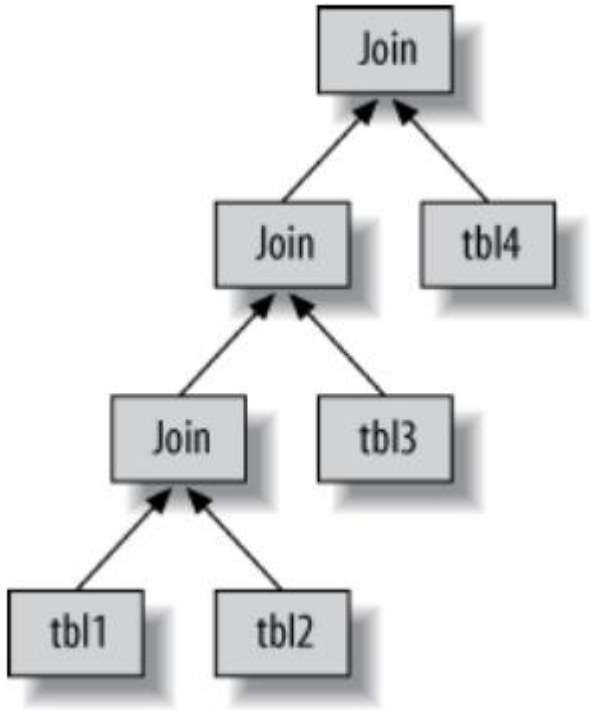
- 找到分辨率最强的条件
  - 解决方案不止一种，查询和数据隐含的假设密切相关
  - 预先考虑优化器的工作，以确定它能找到所需要的数据





多表关联的不同优化器的策略





思考题

- 你可以比较两个查询，在MySQL的Sakila示例数据库中分析不同查询的差异

1

```
SELECT DISTINCT film.film_id
FROM sakila.film
        INNER JOIN sakila.film_actor USING(film_id);
```

2

```
SELECT film_id
FROM sakila.film
WHERE EXISTS(
        SELECT * FROM sakila.film_actor
                WHERE film.film_id = film_actor.film_id);
```

使用EXPLAIN命令查询执行计划和执行时间，分析一下性能差异的原因

End

下一讲，我们讲一下怎么改写SQL降低表连接