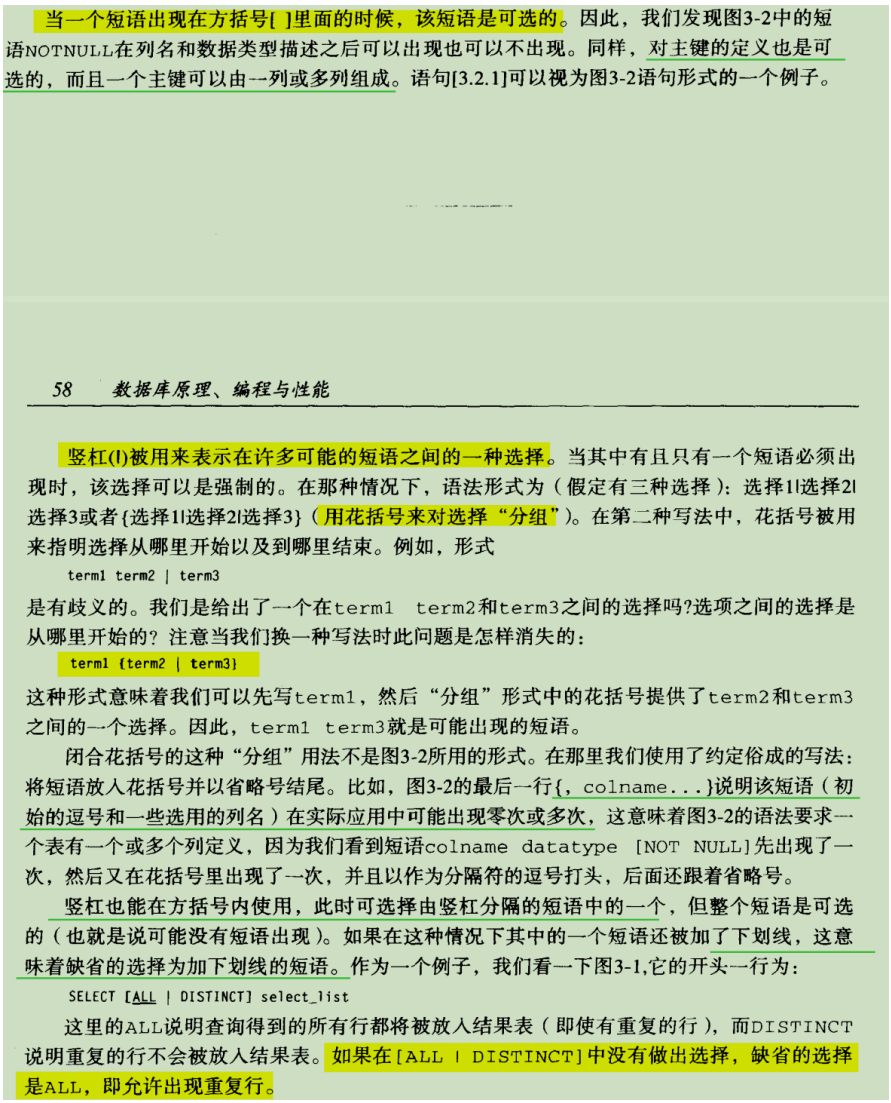
语法描述



当一个短语出现在方括号[ ]里面的时候，该短语是可选的。因此，我们发现图3-2中的短语NOTNULL在列名和数据类型描述之后可以出现也可以不出现。同样，对主键的定义也是可选的，而且一个主键可以由一列或多列组成。语句[3.2.1]可以视为图3-2语句形式的一个例子。

58 数据库原理、编程与性能

竖杠(|)被用来表示在许多可能的短语之间的一种选择。当其中有且只有一个短语必须出现时，该选择可以是强制的。在那种情况下，语法形式为（假定有三种选择）：选择1|选择2|选择3或者{选择1|选择2|选择3}（用花括号来对选择“分组”）。在第二种写法中，花括号被用来指明选择从哪里开始以及到哪里结束。例如，形式

```
term1 term2 | term3
```

是有歧义的。我们是给出了一个在term1 term2和term3之间的选择吗?选项之间的选择是从哪里开始的？注意当我们换一种写法时此问题是怎样消失的：

```
term1 {term2 | term3}
```

这种形式意味着我们可以先写term1，然后“分组”形式中的花括号提供了term2和term3之间的一个选择。因此，term1 term3就是可能出现的短语。

闭合花括号的这种“分组”用法不是图3-2所用的形式。在那里我们使用了约定俗成的写法：将短语放入花括号并以省略号结尾。比如，图3-2的最后一行{, colname...}说明该短语（初始的逗号和一些选用的列名）在实际应用中可能出现零次或多次，这意味着图3-2的语法要求一个表有一个或多个列定义，因为我们看到短语colname datatype [NOT NULL]先出现了一次，然后又在花括号里出现了一次，并且以作为分隔符的逗号打头，后面还跟着省略号。

竖杠也能在方括号内使用，此时可选择由竖杠分隔的短语中的一个，但整个短语是可选的（也就是说可能没有短语出现）。如果在这种情况下其中的一个短语还被加了下划线，这意味着缺省的选择为加下划线的短语。作为一个例子，我们看一下图3-1,它的开头一行为：

```
SELECT [ALL | DISTINCT] select_list
```

这里的ALL说明查询得到的所有行都将被放入结果表（即使有重复的行），而DISTINCT说明重复的行不会被放入结果表。如果在[ALL | DISTINCT]中没有做出选择，缺省的选择是ALL，即允许出现重复行。

select

```
select a ,b from C ,O where=选择+连接+投影
select * from所有行
select distinct aid from，否则不去重
select all aid from，与distinct对应
select a, b from orders as s.a as s.b where=选择+别名+投影

```

- 添加计算量和注释

```
select distinct x.ordno, x.cid, x.aid, x.pid, .40*(x.qty*p.price)
    -.01*(c.discnt + a.percent)*(x.qty*p.price) as profit          -- column alias
    from orders x, customers c, agents a, products p
    where c.cid = x.cid and a.aid = x.aid and p.pid = x.pid;
```

- 别名2

1. 检索符合下述条件的商品的编号：至少有一个客户通过与该客户位于同一个城市的供应商订购过该商品；

【参考答案】

```
select o.pid
from order o, customer c, agent a
where o.cid=c.cid and o.aid=a.aid and c.city=a.city ;
```

```
select c1.cid, c2.cid
from orders c1,orders c2 where...
```

子查询

> 把结果以括号形式调用

in

- 非相关子查询：内层结果与外面无关

例3.4.1　求出通过住在Duluth或Dallas的代理商订了货的顾客的cid值。我们从找出所有住在Duluth或Dallas的代理商开始。此问题可用下面的查询来解决：

```
select aid from agents
    where city = 'Duluth' or city = 'Dallas';
```

该Select语句可被当作是一个值集（子查询）而放入一个更大的解决原问题的Select语句：

```
select distinct cid from orders
    where aid in (select aid from agents
    where city = 'Duluth' or city = 'Dallas');
```

- in可以替换or

```
where city in ('a','b')
where p.per in (select max(per) from O)
where city ='a' or city='b'
```

- 相关子查询：使用外层数据

例3.4.4　找出订购了产品p05的顾客的名字。当然，我们可用最直接的Select语句来解决这个查询问题：

```
select distinct cname from customers, orders
    where customers.cid = orders.cid and orders.pid = 'p05';
```

但是，这里感兴趣的是子查询从外层Select语句中接收数据的解法：

```
select distinct cname from customers where 'p05' in
    (select pid from orders where cid = customers.cid);
```

快速找最小

- some：至少一，和in相同

```
select cid, cname from customers
    where discnt =some (select discnt from customers
        where city = 'Dallas' or city = 'Boston');
```

- all

例3.4.7　找出佣金百分率最小的代理商的aid值。这可以通过下面的查询来实现：

```
select aid from agents where percent <=all (select percent from agents);
```

- 统计函数in（select max)

6. 检索享有最高销售提成比例的供应商（请分别写出使用统计函数和不使用统计函数的两种不同表示方法）；

【参考答案 1&2】使用统计函数

```
select *
from agent
where per in (
    select max ( per )
    from agent ) ;
```

```
select *
from agent
where per = some (
    select max ( per )
    from agent ) ;
```

【参考答案 3&4】不使用统计函数

```
select *
from agent
where per >= all (
    select per
    from agent ) ;
```

```
select *
from agent a1
where not exists (
    select *
    from agent a2
    where a2.per > a1.per ) ;
```

not exist

> exist正向查询非必须

- exist的连接作用

例3.4.10 检索通过代理商a05订货的所有顾客的名字。此查询的基本思想是看orders表中是否存在一个连接cname和代理商a05的行。

```
select distinct c.cname from customers c
    where exists (select * from orders x
        where c.cid = x.cid and x.aid = 'a05');
```

注意子查询的选择序列为*，而不是某个单独的属性；这是测试子查询的结果是否为空的最简方法。从图2-2的表中检索到的是分别对应于cid值c001和c002的顾客名字：TipTop和Basics。注意上面查询中的相关变量x不是必需的，这里包含它仅仅是为了便于和下面的另一种解法做比较：

```
select distinct c.cname from customers c, orders x
    where c.cid = x.cid and x.aid = 'a05';
```

此查询也解决了提出的问题。注意，如果上面的两个查询都没有包含关键词DISTINCT，那么customers表中拥有相同顾客名字的不同行各自对cid列进行限定会导致名字的重复显示。■

union

例3.5.1 建立一个包含了顾客所在的或者代理商所在的或者两者皆在的城市的名单。这可以通过下面的Select语句来实现：

```
select city from customers
    union select city from agents;
```

可以想象：如果一个城市同时满足了上述两个条件，我们就希望它在结果中显示两次（这样的结果包含了更多的信息）。在那种情况下，我们使用下列语句：

```
select city from customers
    union all select city from agents;
```

注意，从customers和agents两表的笛卡儿积中检索单个city列的查询方法在这里是行不通的。

同以往一样，我们可以对任何表达式加括号，而且当涉及到的子查询数目大于或等于3时，括号可被用来判别UNION和UNION ALL的运算顺序。比如，考虑查询

```
(select city from customers
    union select city from agents)
    union all select city from products;
```

except all

```
Q1 EXCEPT ALL Q2
```

如果Q1的结果包含了三个相同的行且Q2的结果也包含了两个与之相匹配的行，那EXCEPT ALL子查询的结果将包含一个这样的行。也就是说，在EXCEPT ALL的结果中，同一行的出现次数将是该行在Q1中的出现次数减去它在Q2中的出现次数（但最小值应为零）。如果关键词ALL被省略了，那上面提到的重复行将被看做同时在Q1和Q2中出现了一次，所以它不会出现在结果中。注意Core SQL-99只包含了简单的EXCEPT 而没有包含EXCEPT ALL。

名称	参数类型	结果类型	描述
COUNT	任意(可以是*)	数值型	出现次数
SUM	数值型	数值型	参数的和
AVG	数值型	数值型	参数平均值
MAX	字符型或数值型	同参数内容一样	最大值
MIN	字符型或数值型	同参数内容一样	最小值

group by

- 可以加多个

[3.8.1]

```
select pid, sum(qty) as total from orders
    group by pid;
```

Select语句的GROUP BY子句将产生一个行集，其效果好比是执行了下面由循环控制的查询：

```
FOR EACH DISTINCT VALUE v OF pid IN orders;
    select pid, sum(qty) as total from orders where pid = v;
END FOR;
```

Select语句[3.8.1]中的GROUP BY子句的结果为表：

pid	total
p01	4800
p02	400
p03	2400
p04	600
p05	2900
p06	400
p07	1400

having

> 可以理解为先多打印点，再根据having删一些

例3.8.3 当某个代理商所订购的某样产品的总量超过了1000时，打印出所有满足条件的产品和代理商的ID以及这个总量。

```
select pid, aid, sum(qty) as TOTAL from orders
    group by pid, aid
    having sum(qty) > 1000;
```

注意，HAVING子句的操作紧跟在GROUP BY子句的操作之后但先于对选择列表中的表达式的计算。该查询打印出查询[3.8.1]的结果表中最右边那一列超过1000的所有行：

pid	aid	TOTAL
p01	a01	3000
p01	a06	1800
p05	a03	2400

```
subquery ::=
    SELECT [ALL | DISTINCT] { * | expr [[AS] c_alias] {, expr [[AS] c_alias]...}}
        FROM tableref {, tableref...}
        [WHERE search_condition]
        [GROUP BY colname {, colname...}]
        [HAVING search_condition]
    |   subquery UNION [ALL] subquery

Select statement ::=
    Subquery [ORDER BY result_column [ASC | DESC] {, result_column [ASC | DESC]...}]
```

除法

3. 检索符合下述条件的商品的编号：在所有有客户的城市中都被销售过；

【参考答案 1】

```
select p.pid from product p
where not exists (
    select * from customer c1
    where c1.city not in (
        select c2.city from order o, customer c2
        where o.pid=p.pid and o.cid=c2.cid ));
```

注：虽然，可以参考在关系代数中‘除法’的推导公式的思路，将上述SQL语句表示成
如参考答案2的表示形式，但仍然含有三层嵌套查询，并没有简化表示。

【参考答案 2】

```
( select pid from product ) except
( select p.pid from product p
  where exists (
    select * from customer c1
    where c1.city not in ( select c2.city from order o, customer c2
                           where o.pid=p.pid and o.cid=c2.cid ));
```

解法一括号内：这个顾客所在城市，不在完成过对应交易的城市范围里

例3.5.2 求出通过住在New York的所有代理商订了货的顾客的cid值。通过上述讨论，该请求的答案为：

```
select cid from customers where cond2;
```

或者写成完整的形式：

```
select c.cid from customers c where        -- select c.cid if...
    not exists (select * from agents a     -- ...there is no agent a.aid
    where a.city = 'New York' and          -- ...living in New York
    not exists (select * from orders x     -- ...where no order row
    where x.cid = c.cid
        and x.aid = a.aid));               -- ...connects c.cid and a.aid
```

听风～ 2021/10/28 23:17:30

1、并未和newyork所有代理商都交易过的顾客（无反例顾客的所有代理商）
2、所有在newyork的代理商且与最终答案的顾客未构成过订单的

- 构造反例：关键对象不在顾客，而在供应商，反例是那个供应商
- 不存在这种反例，外套not exist

- 固定句式：select c1.city not in (select c2.city.....

4. 检索符合下述条件的供应商的编号：为居住在Duluth和Kyoto的所有客户订购过同一种商品；

【参考答案 1】

```
select distinct o1.aid from order o1
where not exist (
    select * from customer c
    where (c.city='Duluth' or c.city='Kyoto') and not exists (
        select * from order o2
        where o2.cid=c.cid and o2.aid=o1.aid and o2.pid=o1.pid)) ;
```

注：
① 因题目中没有提明确要求，最外层**select**子句中的**distinct**保留字可加可不加（后续题目中不再赘述）；
② 可以将内层的**'not exists'**修改为使用**'not in'**（见参考答案**2**）。

【参考答案 2】

```
select distinct o1.aid from order o1
where not exist (
    select * from customer c
    where (c.city='Duluth' or c.city='Kyoto') and c.cid not in (
        select o2.cid from order o2
        where o2.aid=o1.aid and o2.pid=o1.pid)) ;
```

- 哪种商品？缺信息，需要在最顶上给出

仅一次

7. 检索只购买过一次商品的顾客的编号(请写出三种不同类型的查询语句)；

【参考答案 1】在**where**子句中使用**not exists** + 嵌套子查询。

```
select distinct x.cid
from order x
where not exist ( select *
                  from order y
                  where y.cid=x.cid and y.ordno<>x.ordno ) ;
```

【参考答案 2】在**where**子句中使用量词 + 统计查询。

```
select c.cid
from customer c
where 1 = all ( select count(*) from order o where o.cid=c.cid ) ;
```

【参考答案 3】使用 **group by** + **having** 的分组选择统计查询。

```
select cid
from order
group by cid
having count(*) = 1 ;
```

统计并分组排序

8. 对每一位供应商在每一年的销售情况进行统计，结果返回供应商的编号和名称、年份、该供应商在这一年的累计销售金额、订单条数及顾客人数，统计结果先按照年份从小到大排序、同一年份的再按照累计销售金额从大到小排序输出。

【参考答案】在where子句中使用not exists + 嵌套子查询。

```
select a.aid, a.aname, year(o.orddate) as ordyear,
       sum(o.dols) as dols_of_year,
       count(*) num_of_ord,
       count(distinct o.cid) num_of_cus
from order o, agent a
where o.aid = a.aid
group by ordyear, a.aid, a.aname
order by ordyear ASC, dols_of_year DESC ;
```

注：

① 在使用到**group by**子句的分组统计查询中，其**select**子句中只能包含两类结果属性：对所有分组属性的投影，以及统计结果；
② **group by**子句中的所有分组属性，都必须投影到结果关系中去；
③ 在同一条语句中，可以同时进行多种不同的统计。

- 别名直接在select里面取
- sum别名需要as，count 不需要
- count可以（*）还可以加distinct
- where在group前，用于连接表
- group by是每一行的检索信息，真的好用
- ASC升序 DESC降序

> ascend，descend

```
包含字符串：where cname like '%lalalall%'
```