第五章： 视图与子查询

- **创建视图**

```
CREATE VIEW 视图名称(<视图列名1>, <视图列名2>, ……)
AS
<SELECT语句>

CREATE VIEW ProductSum (product_type , cnt_product)
AS
    SELECT
        product_type, COUNT(*)
    FROM
        Product
    GROUP BY product_type;

-- 表中存储的是实际数据，而视图中保存的是从表中取出数据所使用的SELECT语句
-- 应该将经常使用的SELECT语句做成视图
-- 定义视图时不能使用ORDER BY子句
-- 通过汇总得到的视图无法进行更新
```

product_type	cnt_product
衣服	2
办公用品	2
厨房用具	4

- **删除视图**

```
DROP VIEW ProductSum;
```

- **子查询**

```sql
-- 在FROM子句中直接书写定义视图的SELECT语句
-- 子查询就是将用来定义视图的SELECT语句直接用于FROM子句当中
-- 子查询作为内层查询会首先执行。

SELECT
    product_type, cnt_product
FROM
    (SELECT
        product_type, COUNT(*) AS cnt_product
    FROM
        Product
    GROUP BY product_type) AS ProductSum;
```

product_type	cnt_product
衣服	2
办公用品	2
厨房用具	4

- **标量子查询**

```sql
-- 标量子查询就是返回单一值的子查询。
-- 选取出销售单价（sale_price）高于全部商品的平均单价的商品
SELECT
    product_id, product_name, sale_price
FROM
    Product
WHERE
    sale_price > (SELECT
            AVG(sale_price)
        FROM
            Product);
```

product_id	product_name	sale_price
0003	运动T恤	4000
0004	菜刀	3000
0005	高压锅	6800

- **在SELECT子句中使用标量子查询**

```sql
SELECT
    product_id,
    product_name,
    sale_price,
    (SELECT
            AVG(sale_price)
        FROM
            Product) AS avg_price
FROM
    Product;
/*
能够使用常数或者列名的地方，无论是 SELECT 子句、GROUP BY 子句、HAVING 子句，还是 ORDER BY 子句，几乎所有的地方都可以使用标量子查询
*/
```

```
product_id| product_name   | sale_price |       avg_price
----------+---------------+------------+----------------------
0001      | T恤衫          |       1000 | 2097.5000000000000000
0002      | 打孔器         |        500 | 2097.5000000000000000
0003      | 运动T恤        |       4000 | 2097.5000000000000000
0004      | 菜刀           |       3000 | 2097.5000000000000000
0005      | 高压锅         |       6800 | 2097.5000000000000000
0006      | 叉子           |        500 | 2097.5000000000000000
0007      | 擦菜板         |        880 | 2097.5000000000000000
0008      | 圆珠笔         |        100 | 2097.5000000000000000
```

- **关联子查询**

```sql
-- 通过关联子查询按照商品种类对平均销售单价进行比较
-- 在细分的组内进行比较时，需要使用关联子查询
SELECT
    product_type, product_name, sale_price
FROM
    Product AS P1
WHERE
    sale_price > (SELECT
            AVG(sale_price)
        FROM
            Product AS P2
        WHERE
            P1.product_type = P2.product_type
        GROUP BY product_type);
```

product_type	product_name	sale_price
办公用品	打孔器	500
衣服	运动T恤	4000
厨房用具	菜刀	3000
厨房用具	高压锅	6800

谓词

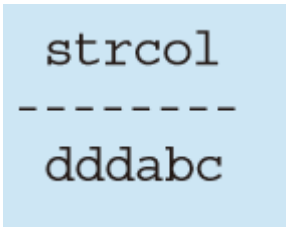
- **LIKE谓词——字符串的部分一致查询**

SampleLike表

strcol（字符串）
abcddd
dddabc
abdddc
abcdd
ddabc
abddc

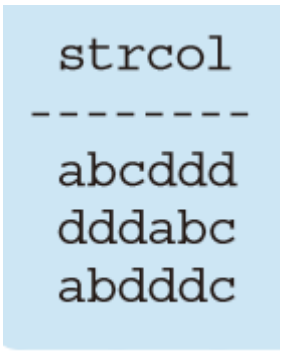
- **使用LIKE进行前方一致查询**

```
SELECT
    *
FROM
    SampleLike
WHERE
    strcol LIKE 'ddd%';
```



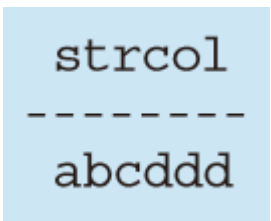
- **使用LIKE进行中间一致查询**

```
SELECT
    *
FROM
    SampleLike
WHERE
    strcol LIKE '%ddd%';
```



- **使用LIKE进行后方一致查询**

```
SELECT
    *
FROM
    SampleLike
WHERE
    strcol LIKE '%ddd';
```



- **使用LIKE和_（下划线）进行后方一致查询**

```
SELECT
    *
FROM
    SampleLike
WHERE
    strcol LIKE 'abc_ _';
```

```
 strcol
--------
 abcdd
```

BETWEEN谓词——范围查询

- **选取销售单价为100～1000日元的商品**

```
SELECT
    product_name, sale_price
FROM
    Product
WHERE
    sale_price BETWEEN 100 AND 1000;
```

```
product_name| sale_price
------------+------------
 T恤衫      |       1000
 打孔器     |        500
 叉子       |        500
 擦菜板     |        880
 圆珠笔     |        100
```

IS NULL、IS NOT NULL——判断是否为NULL

- **选取进货单价（purchase_price）不为NULL的商品**

```sql
SELECT
    product_name, purchase_price
FROM
    Product
WHERE
    purchase_price IS NOT NULL;
```

product_name	purchase_price
T恤衫	500
打孔器	320
运动T恤	2800
菜刀	2800
高压锅	5000
擦菜板	790

IN谓词——OR的简便用法

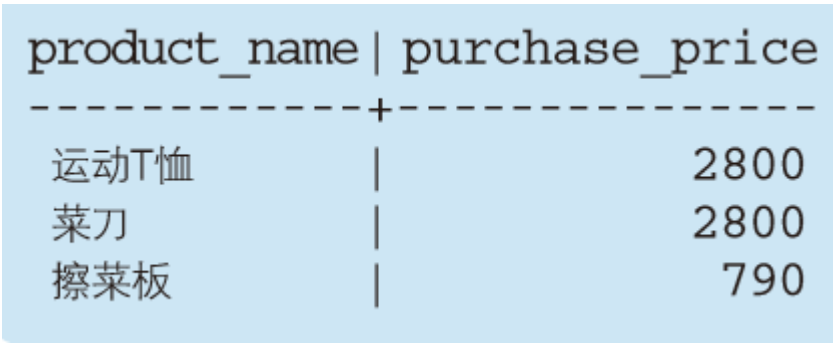
- **通过IN来指定多个进货单价进行查询**

```sql
SELECT
    product_name, purchase_price
FROM
    Product
WHERE
    purchase_price IN (320 , 500, 5000);
-- 在使用 IN 和NOT IN 时是无法选取出 NULL 数据的
```

product_name	purchase_price
T恤衫	500
打孔器	320
高压锅	5000

- **使用NOT IN进行查询时指定多个排除的进货单价进行查询**

```sql
SELECT
    product_name, purchase_price
FROM
    Product
WHERE
    purchase_price NOT IN (320 , 500, 5000);
```



product_name	purchase_price
运动T恤	2800
菜刀	2800
擦菜板	790

- **使用子查询作为IN谓词的参数**

ProductShop表

shop_id（商店）	shop_name（商店名称）	product_id（商品编号）	quantity（数量）
000A	东京	0001	30
000A	东京	0002	50
000A	东京	0003	15
000B	名古屋	0002	30
000B	名古屋	0003	120
000B	名古屋	0004	20
000B	名古屋	0006	10
000B	名古屋	0007	40
000C	大阪	0003	20
000C	大阪	0004	50
000C	大阪	0006	90
000C	大阪	0007	70
000D	福冈	0001	100

- **使用子查询作为IN的参数**

```sql
-- 取得“在大阪店销售的商品的销售单价”
SELECT
    product_name, sale_price
FROM
    Product
WHERE
    product_id IN (SELECT
            product_id
        FROM
            ShopProduct
        WHERE
            shop_id = '000C');
```

product_name	sale_price
叉子	500
运动T恤	4000
菜刀	3000
擦菜板	880

CASE表达式

- **搜索CASE表达式**

```sql
CASE    WHEN <求值表达式> THEN <表达式>
        WHEN <求值表达式> THEN <表达式>
        WHEN <求值表达式> THEN <表达式>
        ELSE <表达式>
END

-- 通过CASE表达式将A～C的字符串加入到商品种类当中
SELECT
    product_name,
    CASE
        WHEN product_type = '衣服' THEN 'A : ' || product_type
        WHEN product_type = '办公用品' THEN 'B: ' || product_type
        WHEN product_type = '厨房用具' THEN 'C : ' || product_type
        ELSE NULL
    END AS abc_product_type
FROM
    Product;
```

product_name	abc_product_type
T恤衫	A：衣服
打孔器	B：办公用品
运动T恤	A：衣服
菜刀	C：厨房用具
高压锅	C：厨房用具
叉子	C：厨房用具
擦菜板	C：厨房用具
圆珠笔	B：办公用品

- **使用CASE表达式进行行列转换**

```
-- 对按照商品种类计算出的销售单价合计值进行行列转换
SELECT
    SUM(CASE
        WHEN product_type = '衣服' THEN sale_price
        ELSE 0
    END) AS sum_price_clothes,
    SUM(CASE
        WHEN product_type = '厨房用具' THEN sale_price
        ELSE 0
    END) AS sum_price_kitchen,
    SUM(CASE
        WHEN product_type = '办公用品' THEN sale_price
        ELSE 0
    END) AS sum_price_office
FROM
    Product;
```

sum_price_clothes	sum_price_kitchen	sum_price_office
5000	11180	600

- **简单CASE表达式**

```
CASE <表达式>    WHEN <表达式> THEN <表达式>
                 WHEN <表达式> THEN <表达式>
                 WHEN <表达式> THEN <表达式>
                 ELSE <表达式>
```

```
END

SELECT
    product_name,
    CASE product_type
        WHEN '衣服' THEN 'A : ' || product_type
        WHEN '办公用品' THEN 'B: ' || product_type
        WHEN '厨房用具' THEN 'C : ' || product_type
        ELSE NULL
    END AS abc_product_type
FROM
    Product;
```