SQL

type	Post
status	Published
date	@2026/01/05
summary	MySQL+copilot
tags	工具 必看精选
category	知识小记
标签	
状态	进行中

1.基本数据结构

表 – 结构化的文件，存储特定类型的数据。如：清单、目录等。

1. 每个表中数据的类型都相同。
2. 每个表都有独属于自己的唯一名字来表示自身。

3．表的一些特性定义了数据再表中的存储方式，包含怎样的数据，数据如何分解等信息。

列

表中的一个字段，所有表都是由一个或多个竖向的列组成的。

每个表列都有相应的数据类型，它限制（或允许）该列中存储的数据。

行

表中的一个记录

表中的数据是按行存储的，所保存的每个记录存储在自己的行内。如果将表想象为网格，网格中垂直的列为表列，水平行为表行。

主键

一列（或一组列），其值能够唯一标识表中每一行。

表中的每一（几）行/列都可以唯一标识自己。而唯一标识表中每行的**这个列称为 主键** 。用它来表示一个特定的行。没有主键，更新或者删除表中特定行极为困难。

列可充当主键的条件：

- 任意两行都不具有相同的主键值；
- 每一行都必须具有一个主键值（主键列不允许 NULL 值）；
- 主键列中的值不允许修改或更新；
- 主键值不能重用（如果某行从表中删除，它的主键不能赋给以后的新行）。

2.数据检索 - SELECT

必须指明选择的对象以及从何处选择。

基本检索

```
SELECT name1, name2, FROM produccts;
              对象1  对象2       对象范围
```

检索所有列

```
SELECT * FROM Products
```

一般而言，尽量不使用 * 通配符。因为检索不需要的列会降低检索和应用程序的性能。

检索不同值 — 使用DISTINCT关键字。

```
SELECT DISTINCT id FROM Products;
```

尽管在id所标识的列中会有多个相同值的行，现在只会输出不同id的行。

限制结果

当我们想只返回某一数量的行时，SQL并不支'0'。

MySQL

```
SELECT prod_name
FROM Products
LIMIT 5;
```

在SQL server 和 ACCESS 中：

```
-- 只返回前五行
SELECT TOP 5 prod_name
FROM Products;
```

DB2

```
SELECT prod_name
FROM Products
```

```
FETCH FIRST 5 ROWS ONLY;
```

Oracle

```
SELECT prod_name
FROM Products
WHERE ROWNUM <=5
```

注释方法

1. -- 这是一条注释

2. # 这是一条注释

3./*这是
一段注释*/

3.排序检索数据 - ORDER BY

排序

子句SQL语句由子句构成，有些子句可选有些必须。一个子句通常由一个关键字加上所提供的数据组成。如上面提到的FROM就是 SELECT 的子句。在指定一条 ORDER BY 子句时，应该保证它是 SELECT 语句中最后一条子句。如果它不是最后的子句，将会出现错误消息。

```
SELECT prod_name
FROM Products
ORDER BY prod_name; -- 默认以pro_name 的字母顺序对数据排列
```

按多个列排序

简单指定列名，在ORDER BY 后的两个列名用逗号分开即可。

```
SELECT prod_id, prod_price, prod_name
FROM Products
ORDER BY prod_price, prod_name;

-- 输出
prod_id prod_price  prod_name
------- ---------- --------------------
BNBG02  3.4900      Bird bean bag toy
BNBG01  3.4900      Fish bean bag toy
BNBG03  3.4900      Rabbit bean bag toy
RGAN01  4.9900      Raggedy Ann
BR03    11.9900 18  inch teddy bear
```

按列位置排序

ORDER BY 还支持按相对列位置进行排序。

若我们希望先按照SELECT清单中的第二个列排序然后按照第三个列排序：

```
SELECT prod_id, prod_price, prod_name
FROM Products
ORDER BY 2, 3;
```

指定排序方向

降序： DESC

升序： ASC

```
SELECT prod_id, prod_price, prod_name
FROM Products
ORDER BY prod_price DESC, prod_name;
-- 按pro_price降序排序，并在其值相同时按照prod_name的字母顺序排序
```

4. 过滤 - WHERE

💡 通常只会根据特定操作或报告的需要提取表数据的子集。只检索所需数据需要指
定搜索条件（search criteria），搜索条件也称为过滤条件（filter condition）。
数据根据 WHERE 子句中指定的搜索条件进行过滤。

```
-- 从 products 表中检索两个列，但不返回所有行，只返回prod_price 值为 3.49 的行
SELECT prod_name, pro_price
FROM Products
WHERE pro_price = 3.49;
```

🚫 在同时使用 ORDER BY 和 WHERE 子句时，应该让 ORDER BY 位于 WHERE 之后，否则将会产生错误

WHERE 子句操作符

表 4-1 WHERE 子句操作符

操作符	说明
=	等于
<>	不等于
!=	不等于
<	小于
<=	小于等于
!<	不小于
>	大于
>=	大于等于
!>	不大于
BETWEEN	在指定的两个值之间
IS NULL	为 NULL 值

💡 !=和<>通常可以互换。但是，并非所有 DBMS 都支持这两种不等于操作符。

```sql
-- 检索所有价格小于等于 10 美元的产品
SELECT prod_name, prod_price
FROM Products
WHERE prod_price <= 10;
```

不匹配检查 - != 和 <>

```sql
-- 列出所有不是供应商 DLL01 制造的产品
SELECT vend_id, prod_name
FROM Products
WHERE vend_id <> 'DLL01';
```

💡 单引号用来限定字符串。如果将值与字符串类型的列进行比较，就需要限定引号。
用来与数值列进行比较的值不用引号。

范围值检查 - BETWEEN

它需要两个值，即范围的开始值和结束值。

```sql
-- 检索价格在 5 美元和 10 美元之间的所有产品。
SELECT prod_name, prod_price
FROM Products
WHERE prod_price BETWEEN 5 AND 10;
```

空值检查 - IS NULL

💡 NULL
无值（no value），它与字段包含 0、空字符串或仅仅包含空格不同。

```sql
--返回所有没有价格（空 prod_price 字段，不是价格为 0）的产品
SELECT prod_name
FROM Products
WHERE prod_price IS NULL;
```

组合 WHERE 子句

AND 操作符 — 给 WHERE 子句附加条件

```sql
-- 检索由供应商 DLL01 制造且价格小于等于 4 美元的所有产品的名称和价格
SELECT prod_id, prod_price, prod_name
FROM Products
WHERE ven_id = 'DLL01' AND prod_price <= 4;
```

OR 操作符 - 检索匹配任一条件的行

💡 第一个条件得到满足的情况下，就不再计算第二个条件

```sql
-- 检索由任一个指定供应商制造的所有产品的产品名和价格
SELECT prod_name, prod_price
FROM Products
WHERE vend_id = 'DLL01' OR vend_id = ‘BRS01’;
```

求值顺序

❗ SQL在处理 OR 操作符前，优先处理 AND 操作符

```sql
--错误写法
-- 返回的行中有 4 行价格小于 10 美元，但是我们需要列出价格为 10美元及以上。
SELECT prod_name, prod_price
```

```sql
-- 用括号提高 OR 的优先级
SELECT prod_name, prod_price
FROM Products
WHERE (vend_id = 'DLL01' OR ven
```

```
FROM Products
WHERE vend_id = 'DLL01' OR vend
_id = 'BRS01'
AND prod_price >= 10;
--输出
prod_name prod_price
------------------ ----------
Fish bean bag toy 3.4900
Bird bean bag toy 3.4900
Rabbit bean bag toy 3.4900
18 inch teddy bear 11.9900
Raggedy Ann 4.9900
```

```
d_id = 'BRS01')
AND prod_price >= 10;
```

IN 操作符号 - 指定条件范围

范围中的每个条件都可以进行匹配。
其实IN 和 OR 具有相同的功能,但是IN的最大优点在于可以包含其他SELECT语句，能动态地建立WHERE子句。

```
-- 检索由供应商 DLL01 和 BRS01 制
造的所有产品
SELECT prod_name, prod_price
FROM Products
WHERE vend_id IN ( 'DLL01', 'BRS0
1' )
ORDER BY prod_name;
```

```
-- same
SELECT prod_name, prod_price
FROM Products
WHERE vend_id = 'DLL01' OR vend
_id = 'BRS01'
ORDER BY prod_name;
```

NOT 操作符 - 否定其后所跟的任何条件。

```
SELECT prod_name
FROM Products
WHERE NOT ven_id = 'DLL01'
ORDER BY prod_name;
```

5 用通配符进行过滤

1、通配符（wildcard）
用来匹配值的一部分的特殊字符。为在搜索子句中使用通配符，必须使用 LIKE 操作符。但是通配符只能用于文本字段（字符串），非文本数据类型字段不能使用通配符搜索。
2、搜索模式（search pattern）
由字面值、通配符或两者构成的搜索条件。

谓词 - 当操作符作为谓词时就不再充当操作符

通配符 %(in DBMS) * (in MA)

%表示任何字符出现任意次数。

```
-- 匹配任何位置上包含文本 bean bag 的值
SELECT prod_id, prod_name
FROM Products
WHERE prod_name LIKE '%bean bag%';
```

```
-- 找出所有以词 Fish 起头的产品
SELECT prod_id, prod_name
FROM Products
WHERE prod_name LIKE 'FISH%';
```

% 不能与NULL匹配

通配符 下划线（_） 单个_只匹配单个字符

1. DB2 不支持 _
2. Access 中为 ?

```
-- 搜索模式给出了后面跟有文本的两个通配符
SELECT prod_id, prod_name
FROM Products
WHERE prod_name LIKE '__ inch teddy bear';
```

通配符 （[]） - 必须匹配指定位置（通配符的位置）的一个字符

```
-- 找出所有名字以 J 或 M 起头的联系人
-- [JM] 匹配方括号中任意一个字符，它也只能匹配单个字符
-- % 通配符匹配第一个字符之后的任意数目的字符，返回所需结果
SELECT cust_contact
FROM Customs
WHERE cust_contact LIKE '[JM]%'
ORDER BY cust_contact;
```

❗ 如果使用的是 Microsoft Access，需要用!而不是^来否定一个集合。

6.计算字段

💡 运行时在SELECT 语句内创建的。

拼接字段

1. 存储在vend_name列中的名字；
2. 包含一个空格和一个左圆括号的字符串；
3. 存储在vend_country列中的国家；
4. 包括一个右圆括号的字符串。

```
-- || 与 + 所表示的含义相同
SELECT vend_name + '(' + vend_country + ')'
-- SELECT vend_name || ' (' || vend_country || ')'
FROM Vendors
ORDER BY vend_name;
```

Mysql 或者 MariaDB 中实现相同方法：

```
-- contact（ob1, ob2）
SELECT Concat(vend_name, ' (', vend_country, ')')
FROM Vendors
ORDER BY vend_name;
```

RTRIM()：去掉字符串右边的空格
LTRIM()（去掉字符串左边的空格）
TRIM()（去掉字符串左右两边的空格）

```
SELECT RTRIM(vend_name) + ' (' + RTRIM(vend_country) + ')'
FROM Vendors
ORDER BY vend_name;
```

别名

💡 SELECT 语句可以很好地拼接地址字段，但是拼接好的字段没有名字，只是一个值。
别名（alias）是一个字段或值的替换名。用 AS 关键字赋予。
别名的名字既可以是一个单词，也可以是一个字符串。如果是后者，字符串应该括在引号中。

```
SELECT RIRIM（vend_namej) + '('
+ RTRIM)vend_country) + ')'
   AS vend_title
FROM vendors
ODER BY vend_name;
```

```
vend_title
----------------------------
Bear Emporium (USA)
Bears R Us (USA)
Doll House Inc. (USA)
Fun and Games (England)
Furball Inc. (USA)
Jouets et ours (France)
```

算数运算

```
-- 检索订单号 20008 中的所有物品
select prod_id, quantity, item_price
from OderItems
where order_num = 20008;
```

```
-- 汇总总物品的价格（单价*订购数量）
SELECT prod_id,
   quantity,
   item_price,
   quantity*item_price AS expanded_price
FROM OrderItems
WHERE order_num = 20008;
```

7 . 函数

不同的DBMS所支持的函数往往不尽相同。所以在使用函数后尽量做好注释^^

使用函数

1. 文本处理
2. 算术操作
3. 处理日期和时间值并提取特定数据
4. 返回DBMS正在使用的特殊信息

文本处理函数

1. upper()：将文本转换为大写

```
SELECT vend_name, UPPER(vend_name) AS vend_name_upcase
 FROM Vendors
 ORDER BY vend_name
```

2. SOUNDEX()：是一个将任何文本串转换为描述其语音表示的字母数字模式的算法。

```
-- 使用 SOUNDEX()函数进行搜索，它匹配所有发音类似于Michael Green的联系名：
SELECT cust_name, cust_contact
FROM Customers
WHERE SOUNDEX(cust_contact) = SOUNDEX('Michael Green');
```

表7-1 常用的文本处理函数

函数	说明
LEFT()（或使用子字符串函数）	返回字符串左边的字符
RIGHT()（或使用子字符串函数）	返回字符串右边的字符
LENGTH()（也使用 DATALENGTH() 或 LEN()）	返回字符串的长度
LOWER()（Access 使用 LCASE()）	将字符串转换为小写
UPPER()（Access 使用 UCASE()）	将字符串转换为大写
LTRIM()	去掉字符串左边的空格

函数	说明
RTRIM()	去掉字符串右边的空格
SOUNDEX()	返回字符串的 SOUNDEX 值
UPPER()（Access使用UCASE()）	将字符串转换为大写

时间和日期处理函数

```
-- 检索 2012 年的所有订单
1. SQL server
SELECT order_num
FROM Orders
WHERE DATEPART(yy, order_date) = 2012;

2. Access
SELECT order_num
FROM Orders
WHERE DATEPART('yyyy', order_date) = 2012;
```

数值处理函数

表7-2 常用数值处理函数

函数	说明
ABS()	返回一个数的绝对值
COS()	返回一个角度的余弦
EXP()	返回一个数的指数值
PI()	返回圆周率
SIN()	返回一个角度的正弦
SQRT()	返回一个数的平方根
TAN()	返回一个角度的正切

8.汇总

聚集函数

💡 对某些行运行的函数，计算并返回一个值。

表8-1 SQL聚集函数

函数	说明	例子	用法
AVG()	返回某列的平均值	AVG(prod_price) AS avg_price	返回 Products 表中所有产品的平均价格
COUNT()	返回某列的行数	COUNT(*) AS num_cust	返回 Customers 表中顾客的总数
MAX()	返回某列的最大值	MAX(prod_price) AS max_price	返回 Products 表中最贵物品的价格
MIN()	返回某列的最小值	MIN(prod_price) AS min_price	返回 Products 表中最便宜物品的价格
SUM()	返回某列值之和	SUM(quantity) AS items_ordered	返回订单中所有物品数量之和

聚集不同值

1. 对所有行执行运算，指定ALL参数或者不指定参数

2. 只包含不同的值，指定DISTINCT参数

```
SELECT AVG(DISTINCT prod_price)
AS avg_price
FROM Products
WHERE vend_id = 'DLL01';
```

💡 在使用了 DISTINCT 后，此例子中的 avg_price 比较高，因为有多个物品具有相同的较低价格。排除它们提升了平均价格。

❗ DISTINCT 不能用于 COUNT(*)

组合聚集函数

```
SELECT COUNT(*) AS num_it
ems,
```

```
num_items price_min price_max price_a
vg
```

```
MIN(prod_price) AS price_mi
n,
MAX(prod_price) AS price_ma
x,
AVG(prod_price) AS price_avg
FROM Products;
```

```
---------- -------------- --------------
---------
9 3.4900 11.9900 6.823333
```

9 . 分组数据

创建分组 - SELECT 语句的 GROUP BY 子句

```
-- 指定了两个列：vend_id 包含产品供应商的 ID，num_prods 为计算字段（用
COUNT(*)函数建立）。
-- GROUP BY 子句指示DBMS 按 vend_id 排序并分组数据。这就会对每个 vend
_id 而不是整个表计算 num_prods 一次。
SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id;
```

❗

1. GROUP BY 子句可以包含任意数目的列，因而可以对分组进行嵌套，更细致地进行数据分组。
2. 如果在 GROUP BY 子句中嵌套了分组，数据将在最后指定的分组上进行汇总。
3. GROUP BY 子句中列出的每一列都必须是检索列或有效的表达式（但不能是聚集函数）。如果在 SELECT 中使用表达式，则必须在 GROUP BY子句中指定相同的表达式。不能使用别名。
4. 大多数 SQL 实现不允许 GROUP BY 列带有长度可变的数据类型（如文本或备注型字段）。
5. 除聚集计算语句外，SELECT 语句中的每一列都必须在 GROUP BY 子句
   中给出。
6. 如果分组列中包含具有 NULL 值的行，则 NULL 将作为一个分组返回。
   如果列中有多行 NULL 值，它们将分为一组。
7. GROUP BY 子句必须出现在 WHERE 子句之后，ORDER BY 子句之前

过滤分组 - HAVING 子句

💡 HAVING 非常类似于 WHERE。事实上，目前为止所学过的所有类型的 WHERE 子句都可以用 HAVING 来替代。唯一的差别是，WHERE过滤行，而 HAVING 过滤分组。
WHERE 在数据分组前进行过滤，HAVING 在数据分组后进行过滤。
使用 HAVING 时应该结合 GROUP BY 子句，而 WHERE 子句用于标准的行级过滤。

```
SELECT cust_id, COUNT(*) AS ord
ers
FROM Orders
GROUP BY cust_id
HAVING COUNT(*) >= 2;
```

```
cust_id orders
---------- -----------
1000000001 2
```

分组和排序

表9-1 ORDER BY 与 GROUP BY

特性	ORDER BY	GROUP BY
作用	对产生的输出排序	对行分组，但输出可能不是分组的顺序
可使用的列	任意列都可以使用（甚至非选择的列也可以使用）	只可能使用选择列或表达式列，而且必须使用每个选择列/表达式
是否必须	不一定需要	如果与聚集函数一起使用列（或表达式），则必须使用

不要忘记 ORDER BY
一般在使用 GROUP BY 子句时，应该也给出 ORDER BY 子句。这是保证数据正确排序的唯一方法。千万不要仅依赖 GROUP BY 排序数据。

```
-- 按订购物品的数目排序输出
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT(*) >= 3
ORDER BY items, order_num;
```

SELECT 子句顺序

表9-2 SELECT子句及其顺序

子句	说明	是否必须使用
SELECT	要返回的列或表达式	是
FROM	从中检索数据的表	仅在从表选择数据时使用
WHERE	行级过滤	否
GROUP BY	分组说明	仅在按组计算聚集时使用
HAVING	组级过滤	否
ORDER BY	输出排序顺序	否