SQL 必知必会

ljalphabeta

Published
with GitBook

Table of Contents

SQL必知必会

《SQL 必知必会》 精简。

将《SQL必知必会》中的语句摘抄出来，方便自己查阅。

检索数据

SELECT语句

检索单个列

```
SELECT column_name FROM tablename;
```

检索多个列

```
SELECT col_name1, col_name2 FROM tablename;
```

检索所有列

```
SELECT * from tablename;
```

检索不同的值, 使用 DISTINCT关键词

```
SELECT DISTINCT col_name1 FROM tablename;
```

注意：DISTINCT关键字用于所有的列，不仅仅是跟在其后的那一列。

限制返回结果数目, 使用LIMIT关键词

以MySQL为例,

```
SELECT col_name FROM tablename LIMIT 10;

SELECT col_name FROM tablename LIMIT 4 OFFSET 3;

SELECT col_name FROM tablename LIMIT 3,4; --逗号前是OFFSET，后面对应LIMIT
```

使用注释

```
SELECT col_name FROM tablename; --这是注释
```

第3课 排序检索数据

这一课讲授如何使用 SELECT语句的ORDER BY子句，根据需要排序检索出的数据。
ORDER BY默认是升序排序。

排序数据

ORDER BY 子句取一个或多个列的名字，据此对输出进行排序。

```
SELECT col_name FROM tablename ORDER BY col_name;
```

注意：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。如果不是最后一条子句，将会出现错误信息。

提示：可以使用非检索的列排序数据哦。

按多个列排序

要按照多个列排序，简单指定列名，列名见用逗号分开即可。

```
SELECT col_name1, col_name2, col_name3 FROM tablename ORDER BY
col_name1, col_name2;
```

重要的是理解 按多个列排序时，排序的顺序完全按规定进行。比如，仅在多个行具有相同的col_name1时才按照col_name2排序，如果col_name1列中的值全部不同，则不会按照col_name2排序。

指定排序方向

默认数据排序方式是升序。可以借助于DESC关键字进行降序排序。

```
SELECT col_name FROM tablename ORDER BY col_name DESC;

SELECT col_name1, col_name2, col_name3 FROM tablename ORDER BY col_name1
DESC, col_name2;
```

注意：DESC 关键词只应用到直接位于其前面的列名。如果想在多个列上进行降序排序，必须对每一列指定DESC关键词。

第4课 过滤数据

这一课讲授如何使用SELECT语句的 WHERE子句指定搜索条件(过滤条件)。

使用WHERE子句

```
SELECT col_name1, col_name2 FROM tablename WHERE col_name1 = 3.49;
```

这条语句只返回col_name1值为3.49的行。

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

```
SELECT col_name, col_name2 FROM tablename WHERE co_name2 BETWEEN 5 AND 10;

SELECT col_name FROM tablename WHERE col_name IS NULL;

SELECT col_name1 FROM tablename WHERE col_name2 IS NULL; --用另一个列来做过滤条件很常用
```

第5课 高级数据过滤

这一课讲授如何组合WHERE子句建立功能更强、更高级的搜索条件。还将学习如何使用NOT和IN操作符。

组合WHERE子句

上一课时介绍的所有WHERE子句在在过滤数据时使用的都是单一的条件。为了进行更强的过滤控制，SQL允许给出多个WHERE子句。这些子句有两种使用方式，即以**AND**子句或**OR**子句的方式使用。

AND操作符

```
SELECT col_name1, col_name2, col_name3 FROM tablename WHERE
col_name1='ABC' AND col_name2 <= 4;
```

OR操作符

```
SELECT col_name1, col_name2, col_name3 FROM tablename WHERE
col_name1='ABC' OR col_name2 <= 4;
```

求值顺序

WHERE子句可以包含任意数目的AND和OR操作符。允许两者结合以进行复杂、高级的过滤。

注意：**SQL**在处理**OR**操作符前，优先处理**AND**操作符。使用圆括号即可解决此问题。

IN操作符

IN操作符用来指定条件范围，范围中的每个条件都可以进行匹配。IN取一组由逗号分隔、括在圆括号中的合法值。

```
SELECT prod_name, prod_price FROM Products WHERE vend_in IN ('DLL01',
'BSLKJL') ORDER BY prod_name;
```

实际上，IN操作符完成了与OR相同的功能。但IN可以包含其他SELECT语句。

NOT操作符

WHERE子句中的NOT操作符有且只有一个功能，那就是否定其后所跟的任何条件。

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

第6课 用通配符进行过滤

这一课介绍什么是通配符、如何使用通配符以及怎样使用LIKE操作符进行通配搜索，以便对数据进行复杂过滤。

LIKE操作符

为了在搜索子句中使用通配符，必须使用LIKE操作符。

注意：通配符搜索只能用于文本字段（字符串），非文本数据字段不能使用通配符搜索。

百分号(%)通配符

%表示任何字符出现任意次数。例如找出所有以Fish起头的产品:

```
SELECT prod_id, pro_name FROM Products WHERE prod_name LIKE 'Fish%';

SELECT prod_id, prod_name FROM Products WHERE prod_name LIKE '%bean bag%';

SELECT prod_id, prod_name FROM Products WHERE prod_name LIKE 'F%y';
```

%还能匹配0个字符。

注意：%不能匹配NULL。

下划线(_)通配符

下划线的用途与%一样，但它只匹配单个字符，而不是多个字符。

```
SELECT prod*id, prodname FROM Products WHERE prod_name LIKE '* in';
```

_总是刚好匹配一个字符，不能多也不能少。

方括号([])通配符

方括号通配符用来指定一个字符集，他必须匹配指定位置的一个字符。

例如匹配J或M开头的任意cust_contact

```
SELECT Customers FROM Products WHERE cust_contact LIKE '[JM]%' ORDER BY cust_contact;
```

方括号通配符可以使用^来否定。

```
SELECT Customers FROM Products WHERE cust_contact LIKE '[JM]%' ORDER BY cust_contact;
```

第7课 创建计算字段

这一课介绍什么是计算字段，如何创建计算字段，以及如何从应用程序中使用别名引用它们。

计算字段

拼接字段

```
SELECT Concat(vend_name, ' (', vend_country, ')') FROM Vendors ORDER BY vend_name;
```

使用别名

别名是一个字段或值的替换名。别名用AS关键字赋予。

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

执行算术计算

计算字段的另一常见用途是对检索出的数据进行算术计算。

```
SELECT prod_id, quantity, item_price, quantity*item_price AS expended_price FROM OrderItems WHERE order_num=20008;
```

第8课 使用函数处理数据

这一课介绍什么是函数，数据库系统支持何种函数，以及如何使用这些函数，还将讲解为什么SQL函数的使用可能会带来问题。

使用函数

文本处理函数

函数	说 明
LEFT()	返回字符串左边的字符
LENGTH()	返回字符串的长度
LOWER()	将字符串转为小写
UPPER()	将字符串转为大写
LTRIM()	去掉字符串左边的空格
RIGHT()	返回字符串右边的字符
RTRIM()	去掉字符串右边的空格
SOUNDEX()	返回字符串的SOUNDEX值

日期和时间处理函数

数值处理函数

数值处理函数仅仅处理数值数据。

第9课 汇总数据

这一课介绍什么是SQL的聚集函数，如何使用他们汇总表的数据。

聚集函数

我们常常需要汇总数据而不用把他们实际检索出来。

函数	说 明
AVG()	返回某列的平均值
COUNT()	返回某列的行数
MAX()	返回某列的最大值
MIN()	返回某列的最小值
SUM()	返回某列值之和

AVG()函数

AVG()函数通过对表中函数计数并计算其列值之和，求得该列的平均值。

```
SELECT AVG(prod_price) AS avg_price FROM Products;

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

注意：AVG()只能用来确定特定数值列的平均值，而且列名必须作为函数参数给出。为了获得多个列的平均值，必须使用多个AVG()函数。 AVG()函数忽略值为NULL的行。

COUNT()函数

COUNT()函数进行计数。可利用COUNT()确定表中行的数目或符合特定条件的行的数目。

COUNT()函数有两种使用方式：

- COUNT(*)对表中行的数目进行计数，不管表列中包含的是空值NULL还是非空值
- COUNT(column_name)对特定列中具有值的行进行计数，忽略NULL值。

```
SELECT COUNT(*) AS num_cust FROM Customers;

SELECT COUNT(cust_email) AS num_cust FROM Customers;
```

MAX()函数

MAX()函数要求指定列名。

```
SELECT MAX(prod_price) AS max_price FROM Products;
```

MAX()函数忽略值为NULL的行。

MIN()函数

与MAX()相反。

SUM()函数

SUM()用来返回指定列值的和（总计）。

```
SELECT SUM(item_price*quantity) AS total_price FROM OrderItems WHERE order_num = 20005;

SELECT SUM(quantity) FROM OrderItems;
```

聚集不同值, 使用DISTINCT关键词

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

组合聚集函数

```
SELECT COUNT(*) AS num_items, MIN(prod_price) AS min_price, MAX(prod_price) AS max_price, AVG(prod_price) AS price_avg FROM Products;
```

第10课 分组数据

这一课介绍如何分组数据，以便汇总表内容的子集。这涉及两个新SELECT子句：GROUP BY子句和HAVING子句。

数据分组

使用分组可以将数据分为多个逻辑组，对每个组进行聚集计算。

创建分组

分组是使用SELECT语句的GROUP BY子句建立的。

```
SELECT vend_id, COUNT(*) AS num_prods FROM Products GROUP BY vend_id;
```

GRUOP BY子句指示数据库系统按vend_id排序并分组数据。 GROUP BY子句指示数据库系统分组数据，然后对每个组而不是整个结果集进行聚集。意思是每个分组都单独进行COUNT()计算。

注意：GROUP BY子句必须出现在WHERE子句之后，ORDER BY子句之前。

过滤分组，使用**HAVING**

HAVING子句类似WHERE。事实上，目前为止所学过的所有类别的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。

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

GROUP BY子句在HAVING之前。

HAVING和**WHERE**的差别：WHERE在数据分组前进行过滤，HAVING在数据分组后进行过滤。

```
SELECT vend_id, COUNT() AS num_prods FROM Products WHERE prod_price >=4 GROUP BY vend_id HAVING COUNT() >=2;
```

分组和排序

ORDER BY	GROUP BY
对产生的输出排序	对行分组，单输出可能不是分组的排序
任意列都可以使用(甚至非选择的列也可以)	只可能使用选择列或表达式列，而且必须使用每个选择列的表达式
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

> SELECT order_num, COUNT() *AS items FROM OrderItems GROUP BY order_num HAVING COUNT()* >=3;
>
> 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

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

第11课 使用子查询

这一课介绍什么是子查询，如何使用它们。