# 《SQL必知必会》阅读笔记

@author: sdubrz
@date: 2020.04.03
@e-mail: lwyz521604#163.com
重点记录的是我现在已经遗忘的一些东西，所以这个笔记可能对初学者不太友好。

# 第1课 了解SQL

#### 主键

表中的任何列都可以作为主键，只要它满足以下条件：

- 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

#### 什么是SQL

SQL是Structured Query Language的缩写。

# 第2课 检索数据

#### SQL语句的大小写

SQL语句不区分大小写，因此SELECT与select是相同的，同样，写成Select也没有关系。许多SQL开发人员喜欢对SQL关键字使用大写，而对列名和表名使用小写。

#### 结束SQL语句

多条SQL语句必须以分号分隔，单条的话可加可不加，但是可能有的DBMS会要求必须加。

#### DISTINCT

使用DISTINCT关键字，可以指示数据库只返回不同的值。

```
SELECT DISTINCT vend_id
FROM Products;
```

需要注意的是，不能部分使用DISTINCT。DISTINCT关键字作用于所有的列，不仅仅是跟在其后的那一列。例如，如果有如下SQL语句，除非指定的两列完全相同，否则，多有的行都会被检索出来。

```
SELECT DISTINCT vend_id, prod_price
FROM Products;
```

#### 限制结果

如果，想要只返回第一行或者只返回一定数量的行，不同的数据库中这一SQL实现可能不同。

在SQL Server和Access中使用SELECT时，可以使用TOP关键字来限制最多返回多少行，如下所示

```
SELECT TOP 5 prod_name
FROM Products;
```

而在DB2中，则需要用如下实现方式

```
SELECT prod_name
FROM Products
FETCH FIRST 5 ROWS ONLY;
```

如果使用Oracle，需要基于ROWNUM（行计数器）来计算行，像这样

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

如果使用MySQL、MariaDB、PostgreSQL或者SQLite，需要使用LIMIT子句，像这样

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

上面的LIMIT 5指示MySQL等DBMS返回不超过5行的数据。如果想得到后面的5行数据，需要指定从哪儿开始以及检索的行数，像这样

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

其中`LIMIT 5 OFFSET 5`指示MySQL等DBMS返回从第5行起的5行数据。第一个数字是指从哪儿开始，第二个数字是检索的行数。

**第0行**

行号是从0开始的，第一个被检索的行是第0行，而不是第1行。因此，`LIMIT 1 OFFSET 1`会检索第2行，而不是第1行。

**MySQL中的简化版**

MySQL和MariaDB支持简化版的`LIMIT 4 OFFSET 3`语句，即`LIMIT 3,4`。使用这个语法，逗号之前的值对应OFFSET，逗号之后的值对应LIMIT。

# 第3课 排序检索数据

如果不排序，数据一般将以它在底层表中出现的顺序显示，这有可能是数据最初添加到表中的顺序。但是如果数据随后进行过更新或删除，那么这个顺序将会受到DBMS重用回收存储空间的方式的影响。关系数据库设计理论认为，如果不明确规定排序顺序，则不应该假定检索出的数据的顺序有任何意义。

**ORDER BY**

要对结果进行排序可以使用`ORDER BY`子句，如下所示

```
SELECT prod_name
FROM Products
ORDER BY prod_name;
```

需要说明的是ORDER BY子句中的列不一定非得是SELECT中的列。

如果需要按照多个列排序，只需要在ORDER BY子句中将各个列名称用逗号分隔开就可以了。

除了能用列名指出排序顺序外，ORDER BY还支持按相对列位置进行排序。如下是按第2个和第3个属性进行排序。

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

**指定排序方向**

可以通过`DESC`关键字设置降序排序。

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

DESC关键字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关键字。下面的语句就是只作用于价格。

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

# 第4课 过滤数据

**WHERE子句的位置**

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**

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

# 第5课 高级数据过滤

**AND操作符**

**OR操作符**

与大多数编程语言一样，SQL中AND的优先级会比OR高。如果有需要优先计算OR的话，可以加括号。

**IN操作符**

```
SELECT prod_name, prod_price
FROM Products
WHERE vend_id IN ('DLL01', 'BRS01')
ORDER BY prod_name
```

为什么要用IN操作符？其优点如下

- 在很多合法选项时，IN操作符的语法更清楚，更直观
- 在与其他AND和OR操作符组合使用IN时，求值顺序更容易管理
- IN操作符一般比一组OR操作符执行得更快
- IN的最大优点是可以包含其他SELECT语句，能够更动态地简历WHERE子句。

**NOT操作符**

```
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

**LIKE操作符**

**%通配符**

%表示任意字符出现任意次数。

如果是Access，需要使用星号而不是百分号。

下面的例子找出以F开头，以y结尾的所有产品

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

**下划线通配符**

下划线匹配单个字符，而不是多个字符。需要注意的是DB2不支持下划线通配符，如果是Access需要使用问号而不是下划线。

与百分号能匹配0个字符不同，下划线总是刚好匹配一个字符，不能多也不能少。

**方括号通配符**

# 第7课 创建计算字段

**拼接（concatenate）**

将值联结到一起（将一个值附加到另一个值）构成单个值。有的用加号，有的用双竖杠，还有的需要用特殊函数。

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

`TRIM()`去除字符串左右两边的空格

`LTRIM()`去除字符串左边的空格

`RTRIM()`去除字符串右边的空格

**使用别名**

可以使用AS关键字来赋予别名

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

# 第8课 使用函数处理数据

每一个DBMS都有特定的函数。事实上，只有少数几个函数被所有主要的DBMS等同地支持。

## 文本处理函数

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 |
| LENGTH()(也使用DATALENGTH()或LEN()) | 返回字符串的长度 |
| LOWER()(Access使用LCASE()) | 转化为小写 |
| LTRIM() | 去除字符串左边的空格 |
| RIGHT() | 返回字符串右边的字符 |
| RTRIM() | 去除字符串右边的空格 |
| SOUNDEX() | 返回字符串的SOUNDEX值 |
| UPPER()(Access使用UCASE()) | 变为大写 |

其中，上面的`SOUNDEX()`函数可以用于查询发音相似的字符串。

## 数值处理函数

| 函数 |
| --- |
| ABS() |
| COS() |
| EXP() |
| PI() |
| SIN() |
| SQRT() |
| TAN() |

# 第9课 汇总数据

**聚集函数（aggregate function）**

对某些行运行的函数，计算并返回一个值。

常用的聚集函数

| 函数 | 说明 |
| --- | --- |
| AVG() | 平均值 |
| COUNT() | |
| MAX() | |
| MIN() | |
| SUM() | |

例如，下面的语句可以计算所有产品的平均价格

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

**AVG()函数**

AVG()函数忽略值为NULL的行。

**COUNT()函数**有两种使用方式：

- 使用`COUNT(*)`对表中行的数目进行计数，不管表列中包含的是不是`NULL`
- 使用`COUNT(column)`对特定列中具有值的行进行计数，忽略`NULL`值。

# 第10课 分组数据

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。

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