[SQL SELECT完整语法](http://blog.csdn.net/younghaiqing/article/details/59482071):参考博客：<http://blog.csdn.net/younghaiqing/article/details/59482071>

**1.SELECT语法**

SELECT[ALL|DISTINCT|DISTINCTROW|TOP]

{\*|talbe.\*|[table.]field1[AS alias1][,[table.]field2[AS alias2][,…]]}

FROM tableexpression[,…][IN externaldatabase]

[WHERE…]

[GROUP BY…]

[HAVING…]

[ORDER BY…]

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

**注意 Where，GroupBy，Having，OrderBy 顺序。**

执行步骤：

1. 先从from字句一个表或多个表创建工作表
2. 将where条件应用于1）的工作表，保留满足条件的行
3. GroupBy 将2）的结果分成多个组
4. Having 将条件应用于3）组合的条件过滤，只保留符合要求的组。
5. Order By对结果进行排序。

**2.FROM子句**

FROM子句是SELECT语句中必不可少的子句，可以使用FROM子句指定查询所需要的数据源名称。语法如下：

FROM table\_source

* 1
* 1

其中，table\_source指定要在SQL语句中使用的表，视图。虽然语句中可用的表源个数的限值可以用内存和查询中其他表达式的复杂性而有所不同，但一个语句中可最多使用256个表源。

注：如果查询中引用了很多表，查询性能会受到影响，编译和优化时间也受到其他因素的影响。

**3.WHERE子句**

在[**数据库**](http://lib.csdn.net/base/mysql)中查询数据时，有时只希望查询所需要的数据，而非数据表中的所有数据，那么就可以使用SELECT语句中的WHERE子句来实现。

WHERE子句通过条件表达式描述关系中元组的选择条件。数据库系统处理该语句时，按行为单位，逐个检查每个行是否满足条件，将不满足条件的行筛选掉。WHERE子句的基本格式如下：

WHERE search\_conditions

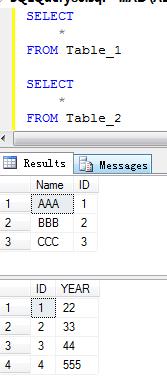
* 1
* 1

其中，search\_conditions为用户所选所需要查询数据行的条件，即查询返回行记录的满足条件。对于用户所需要的所有行，search\_conditions条件为true；而对于其他行，search\_conditions条件为false或未知。

**WHERE子句使用的条件**

| **类别** | **运算符** | **说明** |
| --- | --- | --- |
| 比较运算符 | =,<,>,<=,>=,<> | 比较两个表达式 |
| 逻辑运算符 | AND ,OR, NOT | 组合两个表达式的运算结果或取反 |
| 范围运算符 | BETWEEN,NOT BETWEEN | 搜索值是否在范围内 |
| 列表运算符 | IN,NOT IN | 查询值是否属于列表值之一 |
| 字符匹配符 | LIKE ,NOT LIKE | 字符串是否匹配 |
| 未知值 | IS NULL ,IS NOT NULL | 查询值是否为NULL |

**下面两张表将在后面使用到**

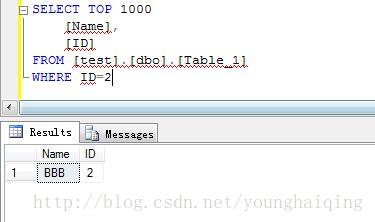


**1.比较运算符**

WHERE expression1 comparison\_operator expression2

* 1
* 1

expression1 ，expression2表示要比较的表达式 ，comparison\_operator 比较运算符。



**2.逻辑运算符**

有时，在查询时指定一个查询条件也很难满足用户需求，需要同时指定多个查询条件，那么久可以使用逻辑运算符将多个查询条件连接起来。

WHERE NOT （expression|expression1 logical\_operator expression2 ）

* 1
* 1

logical\_operator为逻辑运算符

| **运算符** | **功能** |
| --- | --- |
| AND | 只有所有条件满足时才会返回结果结果 |
| OR | 只要其中一个条件满足就会返回查询结果 |
| NOT | 条件不成立时返回查询结果 |

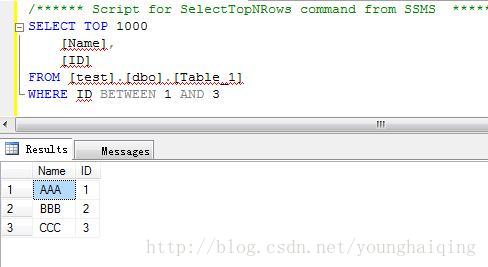


**3.范围运算符**

在WHERE子句中使用BETWEEN关键字查询在一定某个范围内的数据，使用NOT BETWEEN关键字查找不在某一范围内的数据。

WHERE expression [NOT] BETWEEN value AND value2

* 1
* 1



**4.列表运算符**

在WHERE子句中，使用IN关键字可以确定表达式的取值是否属于某一列表值，同样，如果查询表达式不属于某一列值时可以使用NOT IN 关键字。

WHERE expression [NOT] IN value\_list

* 1
* 1

value\_list为列表值，当值不止一个时需要将这些值用括号起来，各列表值之间使用逗号隔开。



**5.字符匹配符**

在WHERE子句中 使用字符匹配符LIKE或NOT LIKE 可以把表达式与字符串进行比较，从而实现对字符串的模糊查询。语法如下：

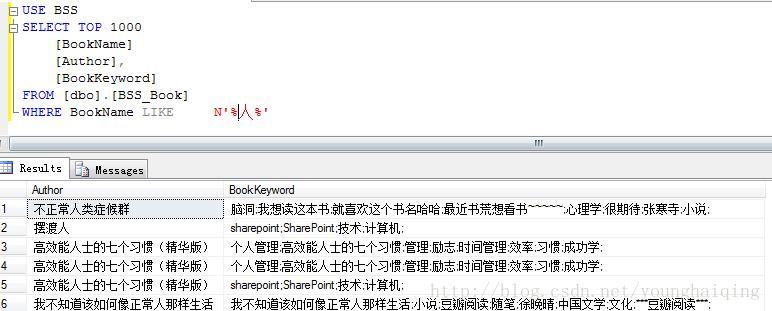
WHERE expression [NOT] LIKE ‘string’

* 1
* 1

其中，[NOT]为可选项，‘string’表示进行比较的字符串。WHERE子句实现对字符串的模糊匹配，进行模糊匹配是在string字符串中使用通配符。

| **通配符** | **说明** | **示例** |
| --- | --- | --- |
| % | 任意多个字符 | H% 表示查询以H开头的任意字符串,如Hello —– %h 表示查询以h结尾的任意字符串，如Growth —— %h% 表示查询在任何位置包含字母的h的所有字–符串，如hui，zhi |
| \_ | 单个字符 | H\_ 表示查询以H开头，后面跟任意一个字符的两位字符串，如Hi，He |
| [] | 指定范围的单个字符 | H[ea]% 表示查询以H开头，第二个字符是e或a的所有字符串，如：Health，Hand ———- [A-G]% 表示查询以A到G之间的任意字符开头的所有字符串，如：Apple，Banana，Guide |
| [^] | 不在指定范围的单个字符 | H[^ea]% 表示查询以H开头，的一个字符不是e或a的所有字符串，如：Hope，Hub ——— [^A\_G]% 表示查询不是以A到G之间的任意字符开头的字符串，如；Job，Zoo |

例如：查找【图书表】中【图书名】还有【人】的所有图书



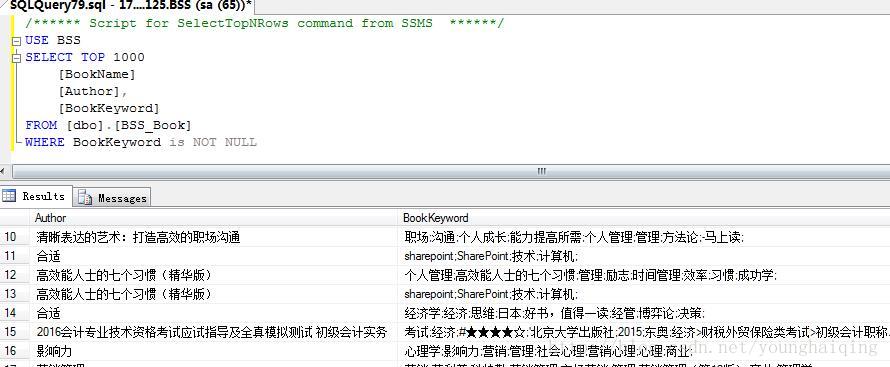
**6.未知值**

当查询数据库中的值为NULL时，可以使用包含IS NULL关键字的WHERE子句进行查询。反之要查询数据库中的值不为NULL时，可以使用IS NOT NULL关键字。

WHERE column IS [NOT] NULL

* 1
* 1

例如：在【图书表】中查询还关键字不为NULL的数据



**4.ORDER BY 子句**

有些时候，在使用SELECT语句进行数据查询后，想先看到众多数据中最新的信息或某列的最大值，就可以使用ORDER BY子句对生成的结果集进行排序。ORDER BY子句在SELECT语句中的语法格式：

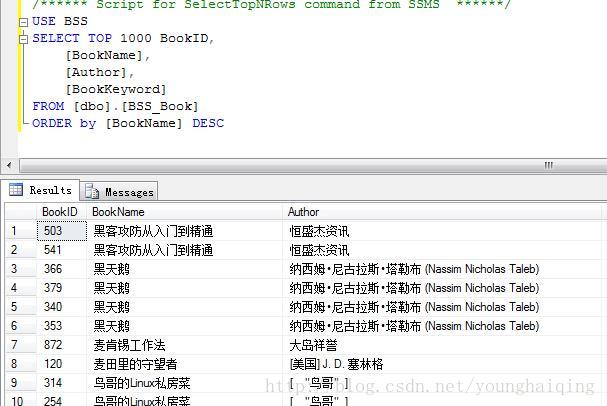
ORDER BY order\_experssion[ASC | DESC]

* 1
* 1

其中，order\_experssion表示用于排序的列或列名及表达式。当有多个排序列时，每个排序了列用逗号隔开，而且列后都可以跟一个排序要求。

* ASC—–升序排序（默认值）
* DESC—降序排序

例如：将【图书表】中的信息按【BookName】倒序排序



例如：将【图书表】中的信息按【BookName】倒序排序和【BookID】倒序排序



**5.GROUP BY 子句**

使用GROUP BY 子句可以将查询结果按照某一列数据值进行分类，换句话说，就是对查询结果的信息进行归纳，以汇总相关数据。

GROUP BY group\_by\_expression[ WITH ROLLUP|CUBE ]

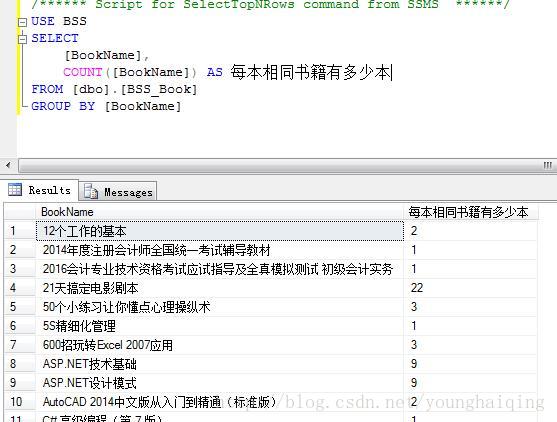
* 1
* 1

其中 ，group\_by\_expression表示分组所依据的列，ROLLUP表示只返回第一个分组条件指定的列的统计行，若改变列的顺序就会使返回的结果行数据发生变化。CUBE是ROLLUP的扩展，表示除了返回由GROUP BY子句指定的列外，还返回按组统计的行。GROUP BY 子句通常与统计函数联合使用。如下表：

| **函数名** | **功能** |
| --- | --- |
| COUNT | 求组中项数 |
| SUM | 求和 |
| AVG | 求平均值 |
| MAX | 求最大值 |
| MIN | 求最小值 |
| ABS | 求绝对值 |
| ASCII | 求ASCII码 |
| RAND | 产生随机数 |

在使用GROUP BY子句时，将GROUP BY子句中的列称为分割列或分组列，而且必须保证SELECT语句中列是可计算的值并且GROUP BY列表中。

例如：查找【图书表】中，相同书名的书籍存在多少本



**6.HAVING**

HAVING子句的用法类似WHERE子句，它指定了组或集合的搜索条件。HAVING子句通常与GROUP BY子句一起使用。HAVING子句的语法格式为：

HAVING search\_conditions

* 1
* 1

其中search\_conditions为查询所需的条件，即返回查询结果的满足条件。在使用GROUP BY 子句时，HAVING子句将限定整个GROUP BY子句创建的组。其具体规则如下：

1. 如果指定了GROUP BY 子句，则HAVING 子句的查询条件应用于GROUP BY子句创建的组
2. 如果指定了WHERE子句而没有指定GROUP BY子句，则HAVING子句的查询条件将应用于WHERE子句的输出结果集
3. 如果既没有指定WHERE子句又没有指定GROUP BY子句，则HAVING子句的查询条件将用于FROM子句的输出结果集



在此查询语句中。HAVING自己与WHERE子句一样，可以使用各种运算符。

**二：SELECT操作多表数据**

在实际查询应用中，用户所需要的数据并不都是在一个表或视图中，而是多个表中，这时就要使用多表查询。多表查询把多个表中数据组合，再从中获取所需要的数据信息。多表查询首先要在这些表中建立连接， 表之间的连接就是查询的结果集，而实现连接的好处是在向数据库添加新类型数据时没有限制的，具有很大的灵活性。通常总是通过连接创建一个新表，以包含不同表中的数据。如果新表有合适的域，就可以把它连接到现有的表中。

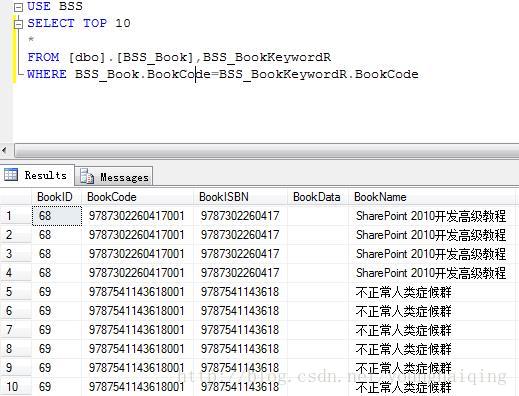
**1.JOIN连接**

在进行多表操作时，最简单的连接方式就是在SELECT语句中引用多个表的字段，在其FROM子句中用逗号将不同的基本表隔开。如果使用WHERE子句创建一个同等连接则能使查询结果集更加丰富。同等连接是指第一个或多个列值与第二个基表中对应一个或多个列值相等的连接。通常情况下使用键码建立连接，即一个基表中的主键码与第二个基表中的外键码保持一致，以保持整个数据库的参照完整性。

用户在进行基本连接操作时，可以遵循以下基本原则。

* SELECT子句列表中，每个目标列都要加上基表名称
* FROM子句应包括所有使用的基表
* WHERE子句应定义一个同等连接

例子：查询【图书表】和【关键字表】，通过BookCode相关联



使用JOIN连接查询和基本连接查询一样都是通过连接 多个表进行操作。其连接条件主要是通过以下方法定义。

* 指定每个表中用于连接的目标列。即在一个基本表中指定外键，在另一个基本表中指定与其关联的键
* 指定在比较各目标列的值时要使用的比较运算符

连接可以在SELECT语句的FROM子句或WHERE子句中创建。连接条件WHERE子句和HAVING子句组合，用于控制在FROM子句引用的基表中所选定的行。JOIN连接查询的语法格式为：

SELECT select\_list

FROM table1 join\_type table [on join\_conditions]

[WHERE search\_conditions]

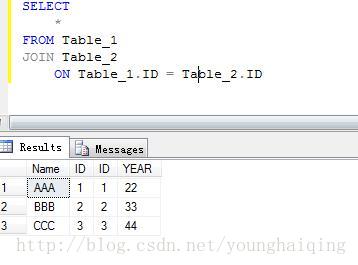
[ORDER BY order\_expression]

* 1
* 2
* 3
* 4
* 1
* 2
* 3
* 4

上诉语法中，table1hetable2为基表，join\_type指定连接类型，join\_conditions指定连接条件。其中类型分为内连接，外连接，交叉连接和自连接。

**2.内连接**

内连接是一种比较常用的数据连接查询方式，它使用比较运算符进行多个基表间数据的比较操作，并列出这些基表中与连接条件相匹配的所有数据行。一般用INNER JOIN或JOIN关键字来指定内连接，它是连接查询默认的连接方式。

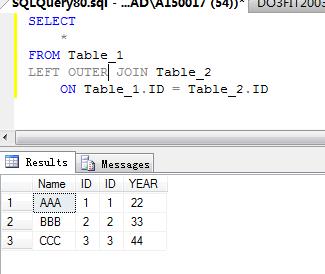


内连接返回的条件是：当且仅当至少有一个同属于两个表的行符合连接条件。内连接从第一个表中消除与另一个表中任何不匹配的行。（2张表都存在的行数据）

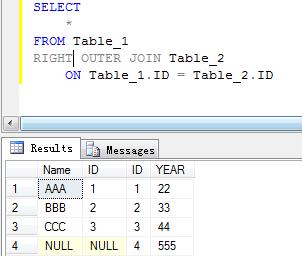
**3.外链接**

外连接与内连接不同，内连接消除与另一个表任何不匹配的行，外连接会返回From子句中提到的至少一个表或视图所有的行，只要这些行符合任何搜索条件。因为在外链接中参与连接的表中有主次之分，以主表的数据行去匹配从表中的数据行如果符合连接条件则直接返回查询结果中，如果主表中的行在从表中没有找到匹配的行，主表的行任然保留返回查询结果中，相应地表中的行被填上空值后也返回到查询结果中。

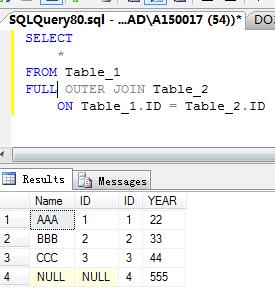
| **链接方式** | **返回数据** |
| --- | --- |
| 左外连接（LEFT OUTER JOIN） | 返回所有匹配行并从关键字JOIN左表中返回所有不匹配发行 |
| 右外连接(RIGHT OUTER JOIN) | 返回所有匹配行并从关键字JOIN右表中返回所有不匹配发行 |
| 完全连接(FULL OUTER JOIN) | 返回两个表中所有匹配行和不匹配行 |

**1. 左外连接**   


**2.右外连接**



**3.完全连接**



**4.交叉连接**

当对两个基表使用交叉连接查询时，将生成来自这两个基表各行所有可能的组合。即在结果集中，两个基表中每两个可能成对的行占一行。在交叉连接中，查询条件一般限定在WHERE子句中，查询生成的结果集分为以下两种情况：

**不使用WHERE子句**

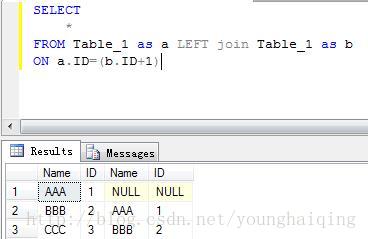
当交叉连接查询语句中没有使用WHERE子句时，返回的结果集是被连接的两个基表所有行的笛卡尔积，即返回到结果集中的行数等于一个基表中符合查询条件的行数乘以另一个基表中的符合条件的行数。

**使用WHERE子句**

当交叉连接查询语句中使用WHERE子句时，返回的结果集是被连接的两个基表所有的行的笛卡尔积减去WHERE子句条件搜索到的数据的行数。

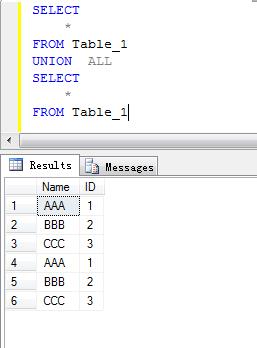
**5.自连接**

自连接是指一个表与自身相连接的查询。自连接操作通过给基表定义别名的方式来实现。



**6.联合查询**

联合查询是指将多个不同的查询结果连接在一起组成一组数据的查询方式。联合查询使用UNION关键字连接SELECT子句，将两个或多个查询结果集组合为单个结果集，该集包含了所有查询结果集汇总的全部行数据。



**注：在使用UNION关键字进行联合查询时，应保证每个联合查询语句的选择列表中具有相同数量的表达式，并且每个查询选择表达式应具有相同的数据类型，或者可以自动将它们转化为相同的数据类型。在自动转换时，对于数值类型，系统将低精度的数据类型转换为高精度的数据类型**

**7.使用子查询**

子查询和连接查询一样提供了使用单个查询操作多个表中的数据的方法。子查询在其他查询结果的基础上提供了一种有效的方式来表示WHERE子句的条件。子查询可以分为返回多行的子查询，返回单值和嵌套子查询3种。

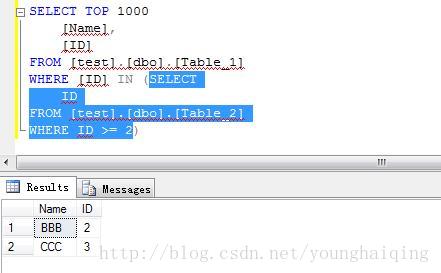
**1.返回多行的子查询**

返回多行子查询是指查询语句获得的结果集中返回了多行数据的子查询。在子查询中可以使用IN,EXISTS和比较运算符来连接表。

* **IN关键字** 用来判断一个表中指定列的值是否包含在已定义的列表或另一个表中。通过使用IN关键字把原表中目标列的值和子查询的返回结果进行比较，如果列值与子查询的结果一致或存在与之匹配的数据行，则查询结果集中就包含该数据行。
* **EXISTS关键字** 表示子查询不需要返回多行数据，而只需要返回一个真值或假值。也就是说，它的作用是在WHERE子句中测试子查询返回的行是否存在。如果存在则返回真值，如果不返回则返回假值。
* **比较运算符** 与使用IN关键字引入的子查询一样，由比较运算符与一些关键字（ANY|ALL|SOME）引入的子查询返回一个值列表。

**2.返回单值的子查询**

例如：使用IN查询–



**3.嵌套子查询**

在SQL中子查询是可以嵌套使用的，并且用户可以在一个查询中嵌套任意多个子查询，即一个子查询中还可以包含另一个子查询。

