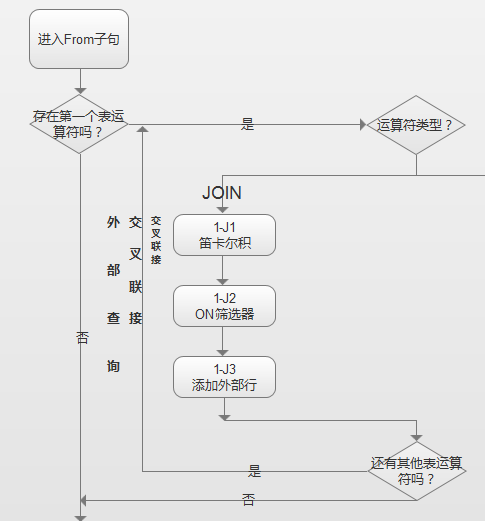
**逻辑查询处理**

SQL与其他编程语言不同的最明显特征是代码的查询顺序。在大多数语言中，代码是按照编写顺序来处理的；但在SQL中，第一个要处理的子句是FROM子句，尽管SELECT是第一个出现，但它几乎都是最后才处理。

每一步都会生成一个虚拟表，该虚拟表会称为下一步的输入，这些虚拟表对于用户是不可见的。如果在查询中没有指定某一个子句，则会跳过相应的步骤。

**步骤1：From阶段**

****

From阶段负责标识要查询的表，如果指定了表运算符，这个阶段还要按照从左到右的顺序对这些运算符进行处理。

**步骤1-J1：执行笛卡尔积（交叉联接）**

这个阶段对联接涉及的两个表执行笛卡尔积运算，生成虚拟表**VT1-J1**。这个虚拟表为左表行和有表行的每一个可能的组合包含一行数据。如果左表包含n行，右表包含m行，**VT1-J1**包含m \* n行。

**步骤1-J2：应用ON筛选器（联接条件）**

ON筛选器是查询中可以指定的三种筛选器（ON，Where，Having）中的第一个。ON筛选器作用于虚拟表（VT1-J1）中的所有行。只有使<on\_predicate>为True的那些行，才会包含在这一步返回的虚拟表（**VT1-J2**）。

**步骤1-J3：添加外部行（Outer Row）**

这一步只在外联接（outer join）中发生。对于外联接，通过为其指定一种外联接类型（LEFT ,RIGHT或FULL），就把一个或两个表标记为保留表（preserved table）。把表标记为**保留表**，即表示希望返回该表的所有行，即使ON筛选器过滤掉了这些行。

**左外联接**是把左表标记为保留表，**右外联接**是把右表标记为保留表，**完全外联接**是把两个表标记为保留表。

步骤1-J3返回VT1-J2中的行，以及保留表在1-J2中过滤掉的行。这些新行被称为**外部行**（outer row）。外部行中非保留表的列值被赋值为NULL，最后生成**VT1-J3**。

注意：*如果From子句中有多个表运算符，则按照从左到右的顺序进行处理。每个表运算符的结果作为下一个表运算符的左输入。最终生成的虚拟表将作为下一阶段的输入。*

**步骤2：Where阶段**

对上一步返回的虚拟表中返回的所有行应用Where筛选器。只有让<where\_predicate>逻辑条件为True的行，才会组成这一步要返回的虚拟表**VT2**。

注意：由于还有对表进行分组，所以在Where子句中不能使用聚合。

对于包含外联接的查询，有一个问题是让人困惑的，**何时使用ON筛选器？何时Where筛选器？**

两者的主要区别是：ON在添加外部行之前执行，而Where在添加外部行之后执行。ON中筛选掉的行不是最终的，因为步骤1-J3会将这些行再添加回来；相反Where筛选器对行的删除是最终的。

需要注意的是，只有在使用外联接时，ON和Where才有这种逻辑区别。当使用内连接时，它们是一致的。

**步骤3：GROUP BY阶段**

在Group By阶段，根据<group\_by\_specification>指定的列表，将上一步返回的虚拟表中的行分配到各个组，这一组列被称之为**分组集**。

*例如：GROUP BY C.customerid中C.customerid就是一个分组集。*

这个阶段将上一步返回的虚拟表中的行按组进行分组。由分组集中的所有属性值的每个唯一组合标识出一个组。最终得到的虚拟表VT3由两部分组成：分组后原来的行(原始信息)，组标识符（组信息）。

|  |  |  |  |
| --- | --- | --- | --- |
| 组信息  C.customerid | 原始信息  C.customerid | C.city | O.orderid |
| Jack | Jack  Jack | ShangHai  ShangHai | 1  2 |
| Lucy | Lucy  Lucy  Lucy | Chongqing  Chongqing  Null | 3  4  5 |
| Ajax | Ajax | Wuhan | 6 |

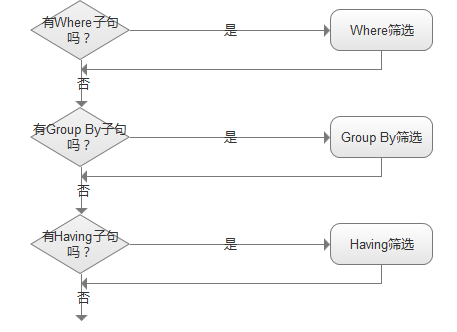
最终，Group By子句的查询将为每一个组生成一行。所以在查询中指定了Group By子句，则后面的所有步骤（Having，Select等）只能指定每个分组上的标量（单个）值表达式。这些表达式可以是Group By列表中的列，表达式（C.customerid），以及聚合行数（Count(C.customerid)）。

这个阶段会认为两个NULL值是相等的。也就是说，所有的NULL值将被分配到同一个组中。

**步骤4：HAVING阶段**

HAVING筛选器用于对上一步返回的虚拟表中的组进行筛选。只有使<having\_predicate>逻辑条件取值为True的组，才会称为这一步返回的虚拟表（VT4）的一部分。

注意：子查询不能作为聚合函数的输入。



**步骤5：SELECT阶段**

虽然SELECT子句出现在查询的最前面，但却放在第五步处理。

**步骤5-1：计算表达式**

SELECT列表中的表达式可以返回上一步得到的虚拟表的基础列，也可以是对这些基础列的操作。如果查询是一个聚合查询，就只能引用上一步返回的虚拟表中的组信息。如果要引用原始信息，就必须对它们进行聚合运算。这个阶段得到的是虚拟表**VT5**。

**步骤5-2：应用Distinct子句**

如果在查询中指定了Distinct子句，将从上一句返回的虚拟表中删除重复的行，并生成虚拟表**VT5-2**。

**步骤5-3：应用Top选项**

Top选项是T-SQL特有的一项功能，允许指定要返回的指定行数或百分比，根据查询的Order by子句来选择指定的数量的行。最终生成虚拟表**VT5-3**。

如果没有指定Order By子句或者未指定With Ties选项，或是根本没有指定Order By子句，那么Top查询就是非确定的。也就是说，返回的行只是SQL Server碰巧在物理上最先访问到的行，因此可能会产生不同的结果，但都可以认为是正确的。

**步骤5：ORDER BY阶段**

这一步按Order By子句中的列名列表对上一步返回的行进行排序，返回游标**VT6**。Order By子句是唯一可以重用SELECT列表中创建列名的步骤。

如果指定了Distinct，则Order By子句中的表达式只能访问上一步返回的虚拟表VT5。如果没有指定Distinct则可以指定任何在Select子句中的表达式。

如果查询中带有Order By子句，则不能用这样的查询来定义表表达式。表表达式包括：视图，内联表值函数，派生表和公用表表达式（CTE）。

除非真的需要有序行，否则不要指定Order By子句，因为排序需要耗费一定的资源。SQL Server需指定有序的索引扫描或sort运算符。

Order By子句认为两个NULL值是相等的。也就是说所有的NULL值是会排序到一起。T-SQL认为NULL值比已知值小。

