提升子连接（具体流程）

基本函数调用

子连接的提升是在 pull_up_sublinks() 这个函数中完成的，主要就是对 Query 查询树的 rtable 和 jointree 进行修改。同时，我们的 SQL 语句是可以嵌套多层的，因此就可能在各个节点中出现子连接，那么我们必须对所有的子连接都进行提升，因此这是一个递归操作

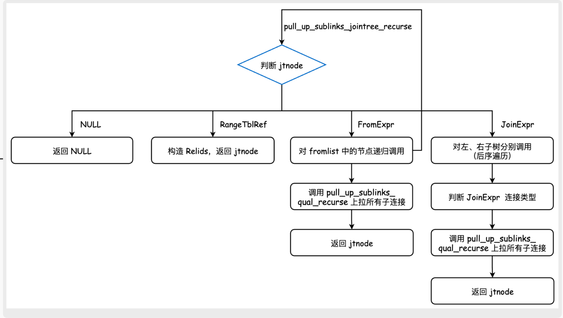
递归函数主要涉及两个，一个是 pull_up_sublinks_jointree_recurse()，另一个则是 pull_up_sublinks_qual_recurse()

static Node * pull_up_sublinks_jointree_recurse(PlannerInfo *root, Node *jtnode, Relids *relids)

- 函数语义非常清晰，就是递归上拉 jointree 中的所有子查询，并返回新的 jointree 节点
- 其中 jtnode 就是 jointree 节点，在递归调用时可能是 FromExpr、JoinExpr 以及 RangeTblRef，relids 则是 RangeTblRef 所组成的 Bitmap 集合，是一个返回值，可以认为就是对孩子节点信息的收集

static Node * pull_up_sublinks_qual_recurse(PlannerInfo *root, Node *node, Node **jtlink1, Relids available_rels1, Node **jtlink2, Relids available_rels2)

- pull_up_sublinks_jointree_recurse 主要用于遍历连接树（jointree）的各个节点，然后进行统筹规划，真正上拉子查询的过程由 pull_up_sublinks_qual_recurse 方法完成。该函数的语义略微复杂一些，因此对其语义的解析贯穿整个 XMind 文件



子连接与连接（JOIN）

当一个连接的约束条件为子连接时，我们就需要做更多的判断，并且在提升子连接时能够引用的关系集合也是不一样的 —— 子连接本质上完全可以看做是一个约束条件

Example

- 比如 SELECT sno FORM student LEFT JOIN course ON student.sno > ANY (SELECT sno FROM SCORE);
- 这是一个左连接，并且子连接的左操作数（student.sno）就是引用的左连接中的 Nonnullable 表，因此如果我们把这个子连接提升的话，将会导致结果和原来不符合

① 对于内连接来说，其连接顺序任意，因此子连接的左操作数既可以引用 LHS 的列属性，又可以引用 RHS 的列属性

- SELECT sno FORM student INNER JOIN course ON student.sno > ANY (SELECT sno FROM SCORE);
- SELECT sno FORM student INNER JOIN course ON course.sno > ANY (SELECT sno FROM SCORE);

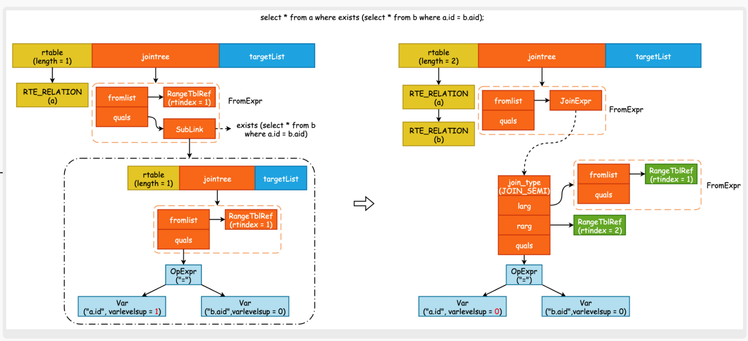
② 对于外连接来说，由于其连接顺序不可任意更改，因此子连接的左操作数必须引用外连接的 nullable 端才可得到提升

比如 A LEFT JOIN B，其中 A 为 Nonnullable 表，B 为 Nullable 表。因此如果存在一个子连接，那么其左操作数应该是 B 表中的列，否则该子连接将无法提升

EXISTS 子连接

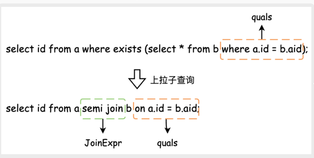
提升条件

Compare



1. 不可包含通用表达式 —— EXISTS 子连接中若存在通用表达式（CTE），那么子连接将不可以提升
2. 子连接必须是"简单"的
   - 通过 `simplify_EXISTS_query` 函数判断 EXISTS 子连接是否"简单"，如果子句中包含集合、聚集操作，或者 HAVING 子句等情况时，子连接也不能提升。同时，当 EXISTS 子连接被判断为"简单"时，还会尝试对其进行优化，即去除 LIMIT/DISTINCT/ORDER BY 等子句，因为我们只需要判断存在性，不在意数据是否完整、数据返回的顺序
   - select * from a where exists (select max(b.aid) from b where a.id = b.aid): 因子查询中包含了 max() 聚合函数，故其不能提升
3. EXISTS 子连接必须是相关子连接
   - EXISTS 子句中，如果约束条件（WHERE/ON）中不包含上层的列属性（Var），也就是说，这是一个非相关的 EXISTS 子连接，那么也不会得到提升，此时将其转换成 InitPlan 是一个更好的选择
   - select * from a where exists (select max(b.aid) from b); 这是一个非相关子连接
4. 不允许在子连接的约束条件之外引用父查询属性 —— EXISTS 类型的子连接的子句中如果在约束条件之外包含了对上层父查询列属性（Var）的引用的话，子连接也不会得到提升
5. 子连接约束条件不可包含易失性函数
   - 最后，如果子连接中的约束条件包含了不稳定函数（VOLATILE）的话，PostgreSQL 为了保证其稳定性，也不会尝试提升该 EXISTS 子连接
   - select * from a where exists (select b.aid from b where a.id = b.aid and b.aid > random()); random() 是 VOLATILE 类型的函数，因此子查询也不会得到提升

提升过程



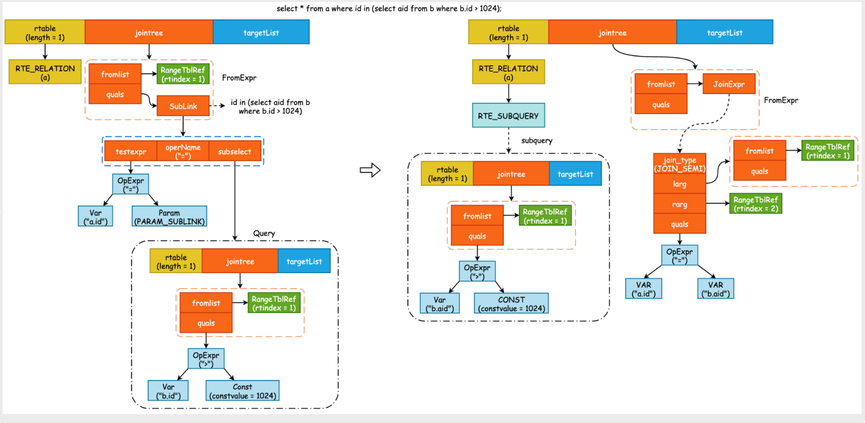
使用一个示例来表明 EXISTS 类型的子连接在提升前和提升后的 SQL 语句，注意本来是没有 semi join 这个语法的，只是为了演示方便

1. 首先，可以看到，EXISTS 类型的子连接中的约束条件，就是提升之后连接的约束条件，因此将 subselect->jointree->quals 保存在 whereCluase 中，并置空子查询中的约束
2. 对 subselect 和 whereClause 中所有 varlevelsu 为 0 的变量，调整其 varnos，使得其引用新的 RangeTblEntry
3. 子连接提升之后，子连接中引用的上层查询的 Var 变量，其 varlevelsup 需要相应地减 1
4. 将子连接的 rtable 追加到父查询的 rtable 中
5. 创建 JoinExpr 结构体，其连接类型是 JOIN_SEMI（EXISTS）或 JOIN_ANTI（NOT EXISTS），连接的RHS是子连接的jointree，LHS 暂时置为NULL，后面会由 pull_up_sublinks_qual_recurse 设置，连接的约束条件就是调整之后 whereClause

ANY 子连接

提升条件

Compare



1. 必须是非相关子连接
   - ANY 类型的子连接中不能引用父查询中的列属性，即 ANY 类型的相关子连接无法提升
   - select * from a where id in (select aid from b where a.id = b.aid); ANY 类型的相关子查询，无法提升
2. 约束条件必须和父查询相关
   - ANY 子连接中的 sublink->testexpr 必须引用了父查询中列属性，也就是说，我们的约束条件得和父查询有关联
   - select * from a where 10 in (select aid from b); 子连接的约束条件和父查询没有任何关系，自然也就不能改写为 Semi Join，所以也不能提升
3. 只能在 testexpr 中引用父查询的列属性 —— ANY 类型子连接中的 sublink->testexpr 不能引用available_rels之外的表
4. 子连接约束条件不可包含易失性函数 —— ANY 类型子连接中的 sublink->testexpr 中不能有易失性函数，即不能出现 Volatile 函数，比如 Random

提升过程

可以看到，子连接将会作为一个子查询和父查询进行连接操作，同时原有的 sublink->testexpr 将会作为连接条件

1. 把子连接中的查询树（sublink->subselect）转变成 RTE_SUBQUERY 类型的 RangeTblEntry，并把它加入到父查询的 rtable 中
2. 获取这个新的 RangeTblEntry 的 rtindex，并生成对应的 RangeTblRef，这样就可以在 jointree 中通过该 RangeTblRef 来引用提升后的子查询了
3. 提取子连接中的查询树（sublink->subselect）的投影列，把它们变成 Var 结构体表示的变量，其中 varno 就是第 2 步中生成的 rtindex
4. 用第 3 步中生成的Var结构体替换 sublink->testexpr 中的 Param 类型的变量
5. 创建新的 JoinExpr 结构体，其中连接类型是 JOIN_SEMI，连接的约束条件是第 4 步替换完成的quals，连接的 RHS 是第 2 步生成的 RangeTblRef，它表示的是提升之后的子查询对应的 RangeTblEntry，连接的 LHS 目前是 NULL，后面会由函数 pull_up_sublinks_qual_recurse 去填充