深度解析MySQL的半连接转换

原创 听见风的声音 2025-07-14 204

1 应用场景

半连接转换（semijoin transforming）是MySQL对in和exist（对not in，not exist的优化是反半连接转换–anti semijoin transforming）的优化技术。这种形式的子查询在数据库中经常看到，有其适用的较广泛的应用场景，这种应用场景在各个业务场景中经常会遇到，下面以常见的学生课程管理系统为例，解释一下这个场景。这个系统数据库有下面这样两张表：

```
mysql> select * from class limit 5;--课程表，存储的是开设的课程的信息
+-----------+---------------------------+-----------------+--------------+
| class_num | class_name                | department      | credit_hours |
+-----------+---------------------------+-----------------+--------------+
|       101 | Calculus I                | Mathematics     |            4 |
|       102 | English Composition       | English         |            3 |
|       103 | Introduction to Programming | Computer Science |            3 |
|       104 | General Physics           | Physics         |            4 |
|       105 | World History             | History         |            3 |
+-----------+---------------------------+-----------------+--------------+
5 rows in set (0.00 sec)
mysql> select * from roster limit 5;--课程登记表，存储的是学生登记的课程的信息
+-----------+-----------+----------------+------------+-----------------+
| roster_id | class_num | student_name   | student_id | enrollment_date |
+-----------+-----------+----------------+------------+-----------------+
|         1 |       102 | Mia Brown      | S90872     | 2023-04-14      |
|         2 |       104 | Donald Wright  | S20618     | 2023-07-13      |
|         3 |       102 | Andrew Gonzalez | S70574     | 2023-01-02      |
|         4 |       101 | Emma Smith     | S17402     | 2023-05-08      |
|         5 |       103 | John Hill      | S78404     | 2023-04-20      |
+-----------+-----------+----------------+------------+-----------------+
5 rows in set (0.00 sec)
```

现在有一个需求，要查询所有有学生登记的课程的课程号和课程名称，查询到课程号和名称即可，并不要求查询每个课程等级学生的人数等信息。这样的查询涉及到两个表，在数据库中一般要写成两表连接，从这个查询的逻辑可以看出这里应该是内连接，形式如下：

```
mysql> SELECT class.class_num, class.class_name     FROM class     INNER JOIN roster     WH
+-----------+---------------------------+
| class_num | class_name                |
+-----------+---------------------------+
|       101 | Calculus I                |
|       101 | Calculus I                |
|       101 | Calculus I                |
|       105 | World History             |
|       105 | World History             |
|       105 | World History             |
+-----------+---------------------------+
3000 rows in set (0.00 sec)
```

返回的结果是错误的，因为有大量的重复值，正确的结果应是去重以后的，使用distinct去重

```
mysql> SELECT distinct class.class_num, class.class_name
    ->     FROM class
    ->     INNER JOIN roster
    ->     WHERE class.class_num = roster.class_num;
+-----------+---------------------------+
| class_num | class_name                |
+-----------+---------------------------+
|       101 | Calculus I                |
|       102 | English Composition       |
|       103 | Introduction to Programming |
|       104 | General Physics           |
|       105 | World History             |
+-----------+---------------------------+
5 rows in set (0.00 sec)
```

这条语句的执行计划如下：

听见风的声音
关注
83 文章 38 粉丝 80K+ 浏览量
获得了 179 次点赞
内容获得 48 次评论
获得了 297 次收藏

热门文章
理解model高级语句 2023-03-01 8437浏览
Oracle会话超时设置1：在sqlnet.ora和listener.ora中设置 2023-02-15 8305浏览
Postgresql 15的安装及简单使用 2023-03-06 4736浏览
oracle 数据库中的行锁和死锁 2023-01-12 4211浏览
Oracle --Oracle 11.2.0.4静默安装 2023-01-17 2176浏览

最新文章
MySQL MCP Server--更轻松的连接大模型和MySQL数据库 5天前 135浏览
执行效率提高数十倍，这几个Oracle SQL的优化技巧你一定要掌握 6天前 240浏览
索引条件下推和分区--一条SQL语句执行计划的分析 2025-07-23 196浏览
null和子查询--not in和not exists怎么选择？ 2025-07-21 182浏览
巧用json工具解析MySQL优化器追踪文件 2025-07-16 99浏览

目录

```
mysql> explain analyze SELECT distinct class.class_num, class.class_name     FROM class
*************************** 1. row ***************************
EXPLAIN: -> Table scan on <temporary>  (cost=7.47..9.84 rows=10) (actual time=0.394..0.394
    -> Temporary table with deduplication  (cost=7.21..7.21 rows=10) (actual time=0.393..0.
        -> Nested loop inner join  (cost=6.21 rows=10) (actual time=0.0944..0.376 rows=5 lo
            -> Table scan on class  (cost=1.25 rows=10) (actual time=0.0206..0.0249 rows=10
            -> Limit: 1 row(s)  (cost=6.4 rows=1) (actual time=0.0347..0.0347 rows=0.5 loop
                -> Covering index lookup on roster using class_num (class_num=class.class_r
1 row in set (0.00 sec)
```

除了这种形式外，也可以写成in或exist形式子查询,下面是in形式的

```
mysql> SELECT class_num, class_name
    ->     FROM class
    ->     WHERE class_num IN
    ->         (SELECT class_num FROM roster);
+-----------+---------------------------+
| class_num | class_name                |
+-----------+---------------------------+
|       101 | Calculus I                |
|       102 | English Composition       |
|       103 | Introduction to Programming |
|       104 | General Physics           |
|       105 | World History             |
+-----------+---------------------------+
5 rows in set (0.00 sec)
```

这条语句执行计划

```
mysql> explain analyze SELECT class_num, class_name     FROM class     WHERE class_num IN
*************************** 1. row ***************************
EXPLAIN: -> Nested loop semijoin  (cost=605 rows=6000) (actual time=0.168..0.62 rows=5 loop
    -> Table scan on class  (cost=1.25 rows=10) (actual time=0.0286..0.0359 rows=10 loops=1
    -> Covering index lookup on roster using class_num (class_num=class.class_num)  (cost=3
```

这条语句执行的是Nested loop semijoin，前面的语句依次执行的Nested loop inner join，Temporary table with deduplication，Table scan on ，可以看到，后面这条语句的执行效率是优于前面那条的，这就是MySQL semijoin transformation（半连接转换）优化的结果。

2 什么是semijoin transformation（半连接转换）

2.1 定义

半连接 (Semijoin) 是一种在SQL语句准备阶段进行的转换，它包含5个执行策略，分别是表拉出 (table pullout)、重复值消除 (duplicate weedout)、首次匹配 (first match)、松散扫描 (loose scan) 和物化 (materialization)。优化器使用这些半连接策略来改进前面所述的子查询的执行。上面的5个策略不仅适用于半连接或反半连接场合，也适用于其它场合，简单解释如下：

1）表拉出（Table Pullout）

如果子查询中的表可以通过唯一键（如主键）与外部查询关联，MySQL 会直接把子查询里的表“拉”到外部查询中，变成普通的 JOIN，避免子查询执行，适用于子查询中的表有主键或唯一索引，且与外部查询的关联条件能确保一对一匹配的情况。

2）重复值消除（Duplicate Weedout）

先执行子查询和外部查询的 JOIN，生成可能有重复的结果集，再通过临时表去重。适用于子查询返回的结果可能有重复（如多对多关系），但最终结果需要去重的场合。

3) 首次匹配（First Match）

对于外部查询的每一行，只要在子查询中找到第一个匹配项就停止扫描，减少不必要的计算。适用于子查询只需要确认“是否存在匹配”，不需要遍历所有结果的场合。

4）松散扫描（Loose Scan）

利用索引快速跳过重复值，只读取子查询中每个分组的第一行，类似于 GROUP BY 的松散索引扫描。适用于子查询的表有合适的索引，且需要按某个字段分组去重的场合。

5）物化（Materialization）

将子查询的结果预先计算并存储到临时表，再通过 JOIN 或索引查找匹配外部查询。适用于子查询结果集较小，且外部查询的表有索引可以高效 JOIN的场合。

在语句的优化阶段，MySQL会根据特定情况，对适用的策略进行评估，选择最终的执行计划。

2.2 半连接适用场景

从MySQL 8.0.16开始支持半连接转换，对exist和in形式的子查询进行半连接转换，从MySQL 8.0.17, 下面形式的子查询被转换反连接（其实是反半连接）

- NOT IN (SELECT ... FROM ...)
- NOT EXISTS (SELECT ... FROM ...).
- IN (SELECT ... FROM ...) IS NOT TRUE
- EXISTS (SELECT ... FROM ...) IS NOT TRUE.

- IN (SELECT ... FROM ...) IS FALSE
- EXISTS (SELECT ... FROM ...) IS FALSE.

2.3 控制半连接优化

在MySQL8.0.16及以后版本，半连接转换默认是打开的，它的5个子策略默认也是打开的

```
mysql> select @@optimizer_switch like '%semijoin=on%';
+-----------------------------------------+
| @@optimizer_switch like '%semijoin=on%' |
+-----------------------------------------+
|                                       1 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

MySQL的优化器参数可以支持全局、会话和参数文件设置，会话级设置立即生效，会话级关闭的命令如下：

```
mysql> set @@optimizer_switch= 'semijoin=off';
```

相关子策略也可以单独关闭，如下

```
set @@optimizer_switch='loosescan=off';
```

半连接转换的策略中，duplicateweedout 有点特殊，如果这个策略被关闭, 除非其他的策略都关闭了，否则不会生效.

2.4 半连接转换开启前后执行计划比较

关闭半连接转换

```
mysql> SET optimizer_switch = 'semijoin=off';
Query OK, 0 rows affected (0.00 sec)
```

语句的执行计划如下

```
mysql> explain analyze SELECT class_num, class_name     FROM class     WHERE class_num IN
*************************** 1. row ***************************
EXPLAIN: -> Filter: <in_optimizer>(class.class_num,<exists>(select #2))  (cost=1.25 rows=16
    -> Table scan on class  (cost=1.25 rows=10) (actual time=0.0327..0.0403 rows=10 loops=1
    -> Select #2 (subquery in condition; dependent)
        -> Limit: 1 row(s)  (cost=60.4 rows=1) (actual time=0.0619..0.0619 rows=0.5 loops=1
            -> Covering index lookup on roster using class_num (class_num=<cache>(class.clc
```

关闭之后，不再执行半连接转换，语句的执行时间好像和执行半连接转换时所差无几，还稍微低点，这是因为MySQL执行了另一种优化（子查询物化），从最后一个操作里的(class.class_num)可以看出这一点。如果使用不使用MySQL 8.0的explain新选项，而是之前的选项，看到的是下面的结果。

```
--关闭半连接优化
mysql> explain SELECT class_num, class_name FROM class WHERE class_num IN (SELECT class_num
+----+--------------------+--------+------------+----------------+---------------+---------
| id | select_type        | table  | partitions | type           | possible_keys | key
+----+--------------------+--------+------------+----------------+---------------+---------
|  1 | PRIMARY            | class  | NULL       | ALL            | NULL          | NULL
|  2 | DEPENDENT SUBQUERY | roster | NULL       | index_subquery | class_num     | class_nu
+----+--------------------+--------+------------+----------------+---------------+---------
--开启半连接优化
mysql> explain SELECT class_num, class_name     FROM class     WHERE class_num IN         (
+----+-------------+--------+------------+------+---------------+----------+---------+----
| id | select_type | table  | partitions | type | possible_keys | key      | key_len | ref
+----+-------------+--------+------------+------+---------------+----------+---------+----
|  1 | SIMPLE      | class  | NULL       | ALL  | PRIMARY       | NULL     | NULL    | NUL
|  1 | SIMPLE      | roster | NULL       | ref  | class_num     | class_num | 4       | air
+----+-------------+--------+------------+------+---------------+----------+---------+----
```

3 半连接的工作机制

半连接的工作机制可以从MySQL优化器的跟踪文件里看到，对MySQL优化器的跟踪默认是关闭的，可以在当前会话中打开，语句如下
打开之后，运行要跟踪的SQL语句，查询 视图，和半连接优化有关的部分如下

3.1 半连接转换

在跟踪文件的join_preparation部分里面可以看到下面的内容，只截取主要的内容，略去冗长的细节信息的信息

```
{
  "steps": [
    {
      "join_preparation": {
        "select#": 1,
        "steps": [
          {
            "join_preparation"
            } /* join_preparation */
          },
          {
            "expanded_query": "/* select#1 */ select `class`.`class_num` AS `class_num`,`cl
          },
          {
            "transformation": {
              "select#": 2,
              "from": "IN (SELECT)",
              "to": "semijoin"
              } /* transformation_to_semi_join */
            } /* transformation */
          },
          {
            "transformations_to_nested_joins": {
              "transformations": [
                "semijoin"
              ] /* transformations */,
              "expanded_query": "/* select#1 */ select `class`.`class_num` AS `class_num`,`
            } /* transformations_to_nested_joins */
          }
        ] /* steps */
      } /* join_preparation */
```

优化器执行的第一步是查询准备（join_preparation），这个工作的操作对象是select#": 1，查询准备下面有四个步骤，连接准备（join_preparation）、扩展查询（expanded_query）和转换（transformation），转换成嵌套链接（transformations_to_nested_joins），我们要关注的是后面两个转换操作，第三个操作操作对象是select#": 2，把它转换成半连接，第四个操作的对象是select#": 1，将它转成成嵌套链接（nested_joins），这里可以看到转成了半连接，也可以看到转换后查询的最终形式。

3.2 半连接优化

优化器执行的第二步是连接优化

```
{
  "join_optimization": {
    "select#": 1,
    "steps": [
      {
        "condition_processing": {
          "condition": "WHERE",
          ------------
          } /* condition_processing */
        },
        {
          "substitute_generated_columns": {
          } /* substitute_generated_columns */
        },
        {
          "table_dependencies": [
            ---------------
          ] /* table_dependencies */
        },
        {
          "ref_optimizer_key_uses": [
            -------------------
          ] /* ref_optimizer_key_uses */
        },
        {
          "pulled_out_semijoin_tables": [
          ] /* pulled_out_semijoin_tables */
        },
        {
          "rows_estimation": [
            ------------
```

```
            ] /* rows_estimation */
          },
          {
            "execution_plan_for_potential_materialization": {
           -----------
            } /* execution_plan_for_potential_materialization */
          },
          {
            "considered_execution_plans": [
              {
                ----------
                "semijoin_strategy_choice": [
                ] /* semijoin_strategy_choice */,
                "rest_of_plan": [
                  {
                    ----------
                    "semijoin_strategy_choice": [
                    {
                      "strategy": "FirstMatch",
                      "recalculate_access_paths_and_cost": {
                        "tables": [
                        ] /* tables */
                      } /* recalculate_access_paths_and_cost */,
                      "cost": 605.211,
                      "rows": 10,
                      "chosen": true
                    },
                    {
                      "strategy": "MaterializeLookup",
                      "cost": 606.25,
                      "rows": 10,
                      "duplicate_tables_left": false,
                      "chosen": false
                    },
                    {
                      "strategy": "DuplicatesWeedout",
                      "cost": 1207.21,
                      "rows": 10,
                      "duplicate_tables_left": false,
                      "chosen": false
                    }
                    ] /* semijoin_strategy_choice */,
                    "chosen": true
                  }
                ] /* rest_of_plan */
              },
              {
                ----另一个执行计划
              }
            ] /* considered_execution_plans */
          },
          {
            "attaching_conditions_to_tables": {
            } /* attaching_conditions_to_tables */
          },
          {
            "finalizing_table_conditions": [
            ] /* finalizing_table_conditions */
          },
          {
            "refine_plan": [
            ] /* refine_plan */
          }
        ] /* steps */
      } /* join_optimization */
    },
```

这部分内容进行了大量的删减，只保留了主干以及于半连接相关的内容，可以看到优化器先执行了pulled_out_semijoin_tables操作，在后面对考虑的执行计划评估（considered_execution_plans)时对半连接的各个策略的成本进行了评估，策略FirstMatch的chosen值为true，策略MaterializeLookup，DuplicatesWeedout的chosen值为false。

3.3 查询执行

贴上这个只是为了保持跟踪文件的完整，和半连接无关。

```
    {
      "join_execution": {
        "select#": 1,
        "steps": [
        ] /* steps */
      } /* join_execution */
    }
  ] /* steps */
}
```

4 对半连接的其它优化措施

在8.0.41+, 8.4.4+ and 9.2+，,MySQL对半连接和反连接提供了提供了进一步的优化措施，通过使用 group skip scan table 访问方法，减少产生访问结果集所需要的主表的行数，详情参考 Improving Semi-join Performance in MySQL。

墨力计划 mysql 性能优化

评论

分享你的看法，一起交流吧～

... LV.4

深度解析MySQL的半连接转换深度解析MySQL的半连接转换

20天前 点赞 评论

相关阅读

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/25期）
墨天轮小助手 469次阅读 2025-07-25 15:54:18

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/17期）
墨天轮小助手 436次阅读 2025-07-17 15:31:18

墨天轮「实操看我的」数据库主题征文活动启动
墨天轮编辑部 379次阅读 2025-07-22 16:11:27

【GaussDB】深入剖析Insert Select慢的定位全过程
DarkAthena 305次阅读 2025-07-27 01:28:24

MySQL 9.4.0 正式发布，支持 RHEL 10 和 Oracle Linux 10
严少安 199次阅读 2025-07-23 01:21:32

索引条件下推和分区--一条SQL语句执行计划的分析
听见风的声音 196次阅读 2025-07-23 09:22:58

null和子查询--not in和not exists怎么选择？
听见风的声音 182次阅读 2025-07-21 08:54:19

MySQL数据库SQL优化案例(走错索引)
陈举超 164次阅读 2025-07-17 21:24:40

使用 MySQL Clone 插件为MGR集群添加节点
黄山谷 162次阅读 2025-07-23 22:04:19

MySQL 8.0.40：字符集革命、窗口函数效能与DDL原子性实践
shunwah 141次阅读 2025-07-15 15:27:19