数据库内核月报 — 2019 / 05

当期文章

MySQL · 引擎特性 · Skip Scan Range

MySQL从8.0.13版本开始支持一种新的range scan方式，称为Loose Skip Scan。该特性由Facebook贡献。我们知道在之前的版本中，如果要使用到索引进行扫描，条件必须满足索引前缀列，比如索引idx(col1,col2), 如果where条件只包含col2的话，是无法有效的使用idx的, 它需要扫描索引上所有的行，然后再根据col2上的条件过滤。

新的优化可以避免全量索引扫描，而是根据每个col1上的值+col2上的条件，启动多次range scan。每次range scan根据构建的key值直接在索引上定位，直接忽略了那些不满足条件的记录。

示例 下例是从官方文档上摘取的例子:

```
root@test 11:03:28>CREATE TABLE t1 (f1 INT NOT NULL, f2 INT NOT NULL, PRIMARY KEY(f1, f2
Query OK, 0 rows affected (0.00 sec)

root@test 11:03:29>INSERT INTO t1 VALUES
->   (1,1), (1,2), (1,3), (1,4), (1,5),
->   (2,1), (2,2), (2,3), (2,4), (2,5);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

root@test 11:03:29>INSERT INTO t1 SELECT f1, f2 + 5 FROM t1;
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

root@test 11:03:29>INSERT INTO t1 SELECT f1, f2 + 10 FROM t1;
Query OK, 20 rows affected (0.00 sec)
Records: 20  Duplicates: 0  Warnings: 0

root@test 11:03:29>INSERT INTO t1 SELECT f1, f2 + 20 FROM t1;
Query OK, 40 rows affected (0.00 sec)
Records: 40  Duplicates: 0  Warnings: 0

root@test 11:03:29>INSERT INTO t1 SELECT f1, f2 + 40 FROM t1;
Query OK, 80 rows affected (0.00 sec)
Records: 80  Duplicates: 0  Warnings: 0

root@test 11:03:29>ANALYZE TABLE t1;
+---------+---------+----------+----------+
| Table   | Op      | Msg_type | Msg_text |
+---------+---------+----------+----------+
| test.t1 | analyze | status   | OK       |
+---------+---------+----------+----------+
1 row in set (0.00 sec)

root@test 11:03:29>EXPLAIN SELECT f1, f2 FROM t1 WHERE f2 > 40;
+----+-------------+-------+------------+-------+---------------+---------+---------+---
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | re
+----+-------------+-------+------------+-------+---------------+---------+---------+---
|  1 | SIMPLE      | t1    | NULL       | range | PRIMARY       | PRIMARY | 8       | NU
+----+-------------+-------+------------+-------+---------------+---------+---------+---
1 row in set, 1 warning (0.00 sec)
```

也可以从optimizer trace里看到如何选择的skip scan:

```
"skip_scan_range": {
  "potential_skip_scan_indexes": [
  {
    "index": "PRIMARY",
      "tree_travel_cost": 0.4,
      "num_groups": 3,
      "rows": 53,
      "cost": 10.625
  }
  ]
},
  "best_skip_scan_summary": {
```

```
    "type": "skip_scan",
    "index": "PRIMARY",
    "key_parts_used_for_access": [
      "f1",
    "f2"
      ],
    "range": [
      "40 < f2"
      ],
    "chosen": true
  },
```

我们从innodb的角度来看看这个SQL是如何执行的，我们知道每个index scan都会走到ha_innobase::index_read来构建search tuple，上述查询的执行步骤：

- 第一次从Index left side开始scan
- 第二次使用key(1,40) 扫描index，直到第一个range结束
- 使用key(1), find_flag =HA_READ_AFTER_KEY, 找到下一个Key值2
- 使用key(2,40)，扫描Index， 直到range结束
- 使用Key(2)，去找大于2的key值，上例中没有，因此结束扫描

笔者在代码注入了日志，打印search_tuple(**dtuple_print()**)

```
STEP 1: no search_tuple

STEP 2:
DATA TUPLE: 2 fields;
0: len 4; hex 80000001; asc     ;;
1: len 4; hex 80000028; asc    (;;

STEP 3:
DATA TUPLE: 1 fields;
0: len 4; hex 80000001; asc     ;;

STEP 4:
DATA TUPLE: 2 fields;
0: len 4; hex 80000002; asc     ;;
1: len 4; hex 80000028; asc    (;;

STEP 5:
DATA TUPLE: 1 fields;
0: len 4; hex 80000002; asc     ;;
```

从上述描述可以看到使用skip-scan的方式避免了全索引扫描，从而提升了性能，尤其是在索引前缀列区分度比较低的时候

条件 skip scan可以通过Hint或者optimizer_switch来控制(skip_scan)，默认是打开的。根据worklog的描述，对于如下query:

```
SELECT A_1,...,A_k, B_1,...,B_m, C
FROM T
WHERE
EQ(A_1,...,A_k)
AND RNG(C);
```

需要满足如下条件才能使用 skip scan:

```
A) Table T has at least one compound index I of the form:
I = <A_1,...,A_k, B_1,..., B_m, C ,[D_1,...,D_n]>
Key parts A and D may be empty, but B and C must be non-empty.
B) Only one table referenced.
C) Cannot have group by/select distinct
D) Query must reference fields in the index only.
E) The predicates on A_1...A_k must be equality predicates and they need
to be constants. This includes the 'IN' operator.
F) The query must be a conjunctive query.
In other words, it is a AND of ORs:
(COND1(kp1) OR COND2(kp1)) AND (COND1(kp2) OR ...) AND ...
G) There must be a range condition on C.
H) Conditions on D columns are allowed. Conditions on D must be in
conjunction with range condition on C.
```

ref: get_best_skip_scan()

当skip scan拥有更低的cost时，会被选择，计算cost的函数是cost_skip_scan()，由于索引统计信息中已经基于不同的前缀列值估算了distinct value的个数(rec_per_key), 可以基于此去预估可能需要读的行数。 更具体的可以参考wl#11322中的描述，笔者对此不甚了解，故不做笔墨 ref: cost_skip_scan()

参考

官方文档：Skip Scan Range Access Method

WL#11322: SUPPORT LOOSE INDEX RANGE SCANS FOR LOW CARDINALITY Bug#88103 相关代码