[InnoDB系列] -- innodb表如何更快得到count(*)结果

By yejr on 24 六月 2008

作/译者：叶金荣（imysql#imysql.com>），来源：http://imysql.com，欢迎转载。

起因：在innodb表上做count(*)统计实在是太慢了，因此想办法看能不能再快点。
现象：先来看几个测试案例，如下
一、 sbtest 表上的测试

```
show create table sbtest\G
*************************** 1. row ***************************
Table: sbtest
Create Table: CREATE TABLE `sbtest` (
`aid` bigint(20) unsigned NOT NULL auto_increment,
`id` int(10) unsigned NOT NULL default '0',
`k` int(10) unsigned NOT NULL default '0',
`c` char(120) NOT NULL default '',
`pad` char(60) NOT NULL default '',
PRIMARY KEY (`aid`),
KEY `k` (`k`),
KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1000001 DEFAULT CHARSET=latin1

show index from sbtest;
+--------+------------+----------+--------------+-------------+-----------+
-------------+----------+--------+------+------------+---------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation
| Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+--------+------------+----------+--------------+-------------+-----------+
-------------+----------+--------+------+------------+---------+
| sbtest |          0 | PRIMARY  |            1 | aid         | A
|     1000099 |     NULL | NULL   |      | BTREE      |         |
| sbtest |          1 | k        |            1 | k           | A
|          18 |     NULL | NULL   |      | BTREE      |         |
| sbtest |          1 | id       |            1 | id          | A
|     1000099 |     NULL | NULL   |      | BTREE      |         |
+--------+------------+----------+--------------+-------------+-----------+
```

```
+----------+---------+-------+------+-----------+---------+
```

填充了 100万条 记录。

1、 直接 count(*)

```
explain SELECT COUNT(*) FROM sbtest;
+----+-------------+-------+-------+---------------+---------+---------+--
----+---------+-------------+
| id | select_type | table | type  | possible_keys | key     | key_len |
ref  | rows    | Extra       |
+----+-------------+-------+-------+---------------+---------+---------+--
----+---------+-------------+
|  1 | SIMPLE      | sbtest | index | NULL          | PRIMARY | 8       |
NULL | 1000099 | Using index |
+----+-------------+-------+-------+---------------+---------+---------+--
----+---------+-------------+
SELECT COUNT(*) FROM sbtest;
+----------+
| COUNT(*) |
+----------+
|  1000000 |
+----------+
1 row in set (1.42 sec)
```

注意：这个不是绝对的，在其它表结构中默认可能不是使用 primary key！

可以看到，如果不加任何条件，那么优化器优先采用 primary key 来进行扫描。

2、count(*) 使用 primary key 字段做条件

```
explain SELECT COUNT(*) FROM sbtest WHERE aid>=0;
+----+-------------+-------+-------+---------------+---------+---------+--
----+--------+-------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len |
ref  | rows   | Extra                   |
+----+-------------+-------+-------+---------------+---------+---------+--
----+--------+-------------------------+
|  1 | SIMPLE      | sbtest | range | PRIMARY       | PRIMARY | 8       |
NULL | 485600 | Using where; Using index |
+----+-------------+-------+-------+---------------+---------+---------+--
----+--------+-------------------------+
SELECT COUNT(*) FROM sbtest WHERE aid>=0;
+----------+
| COUNT(*) |
```

```
+----------+
|  1000000 |
+----------+
1 row in set (1.39 sec)
```

可以看到，尽管优化器认为只需要扫描 485600 条记录(其实是索引)，比刚才少多了，但其实仍然要做全表(索引)扫描。因此耗时和第一种相当。

3、 count(*) 使用 secondary index 字段做条件

```
explain SELECT COUNT(*) FROM sbtest WHERE id>=0;
+----+-------------+--------+-------+---------------+------+---------+-----
-+--------+-------------------------+
| id | select_type | table  | type  | possible_keys | key  | key_len |
ref  | rows   | Extra                   |
+----+-------------+--------+-------+---------------+------+---------+-----
-+--------+-------------------------+
|  1 | SIMPLE      | sbtest | range | id            | id   | 4       |
NULL | 500049 | Using where; Using index |
+----+-------------+--------+-------+---------------+------+---------+-----
-+--------+-------------------------+
SELECT COUNT(*) FROM sbtest WHERE id>=0;
+----------+
| COUNT(*) |
+----------+
|  1000000 |
+----------+
1 row in set (0.43 sec)
```

可以看到，采用这种方式查询会非常快。

有人也许会问了，会不会是因为 id 字段的长度比 aid 字段的长度来的小，导致它扫描起来比较快呢？先不着急下结论，咱们来看看下面的测试例子。

二、 sbtest1 表上的测试

```
show create table sbtest1\G
*************************** 1. row ***************************
Table: sbtest1
Create Table: CREATE TABLE `sbtest1` (
`aid` int(10) unsigned NOT NULL AUTO_INCREMENT,
`id` bigint(20) unsigned NOT NULL DEFAULT '0',
`k` int(10) unsigned NOT NULL DEFAULT '0',
`c` char(120) NOT NULL DEFAULT '',
```

```
`pad` char(60) NOT NULL DEFAULT '',
PRIMARY KEY (`aid`),
KEY `k` (`k`),
KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1000001 DEFAULT CHARSET=latin1
show index from sbtest1;
+--------+------------+---------+--------------+-------------+-----------
+------------+----------+--------+------+------------+---------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation
| Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+--------+------------+---------+--------------+-------------+-----------
+------------+----------+--------+------+------------+---------+
| sbtest1 |          0 | PRIMARY |            1 | aid         | A
|     1000099 |     NULL | NULL   |      | BTREE      |         |
| sbtest1 |          1 | k       |            1 | k           | A
|          18 |     NULL | NULL   |      | BTREE      |         |
| sbtest1 |          1 | id      |            1 | id          | A
|     1000099 |     NULL | NULL   |      | BTREE      |         |
+--------+------------+---------+--------------+-------------+-----------
+------------+----------+--------+------+------------+---------+
```

这个表里，把 aid 和 id 的字段长度调换了一下，也填充了 100万条 记录。

1、 直接 count(*)

```
explain SELECT COUNT(*) FROM sbtest1;
+----+-------------+--------+-------+---------------+---------+---------+-
-----+---------+-------------+
| id | select_type | table  | type  | possible_keys | key     | key_len |
ref  | rows    | Extra       |
+----+-------------+--------+-------+---------------+---------+---------+-
-----+---------+-------------+
|  1 | SIMPLE      | sbtest1 | index | NULL          | PRIMARY | 4       |
NULL | 1000099 | Using index |
+----+-------------+--------+-------+---------------+---------+---------+-
-----+---------+-------------+
SELECT COUNT(*) FROM sbtest1;
+----------+
| COUNT(*) |
+----------+
|  1000000 |
```

```
+----------+
1 row in set (1.42 sec)
```

可以看到，如果不加任何条件，那么优化器优先采用 primary key 来进行扫描。

2、count(*) 使用 primary key 字段做条件

```
explain SELECT COUNT(*) FROM sbtest1 WHERE aid>=0;
+----+-------------+--------+-------+---------------+---------+---------+-
-----+--------+-------------------------+
| id | select_type | table  | type  | possible_keys | key     | key_len |
ref  | rows   | Extra                   |
+----+-------------+--------+-------+---------------+---------+---------+-
-----+--------+-------------------------+
|  1 | SIMPLE      | sbtest1 | range | PRIMARY       | PRIMARY | 4       |
NULL | 316200 | Using where; Using index |
+----+-------------+--------+-------+---------------+---------+---------+-
-----+--------+-------------------------+
1 row in set (0.00 sec)
SELECT COUNT(*) FROM sbtest1 WHERE aid>=0;
+----------+
| COUNT(*) |
+----------+
|  1000000 |
+----------+
1 row in set (1.42 sec)
```

可以看到，尽管优化器认为只需要扫描 316200 条记录(其实是索引)，比刚才少多了，但其实仍然要做全表(索引)扫描。因此耗时和第一种相当。

3、 count(*) 使用 secondary index 字段做条件

```
explain SELECT COUNT(*) FROM sbtest1 WHERE id>=0;
+----+-------------+--------+-------+---------------+------+---------+----
--+--------+-------------------------+
| id | select_type | table  | type  | possible_keys | key  | key_len |
ref  | rows   | Extra                   |
+----+-------------+--------+-------+---------------+------+---------+----
--+--------+-------------------------+
|  1 | SIMPLE      | sbtest1 | range | id            | id   | 8       |
NULL | 500049 | Using where; Using index |
+----+-------------+--------+-------+---------------+------+---------+----
--+--------+-------------------------+
```

```
1 row in set (0.00 sec)
SELECT COUNT(*) FROM sbtest1 WHERE id>=0;
+----------+
| COUNT(*) |
+----------+
|  1000000 |
+----------+
1 row in set (0.45 sec)
```

可以看到，采用这种方式查询会非常快。 **以便让“查询缓冲区”失效**

上面的所有测试，均在 mysql 5.1.24 环境下通过，并且每次查询前都重启了 mysqld。

可以看到，把 aid 和 id 的长度调换之后，采用 secondary index 查询仍然是要比用 primary key 查询来的快很多。看来主要不是字段长度引起的索引扫描快慢，而是采用 primary key 以及 secondary index 引起的区别。那么，为什么用 secondary index 扫描反而比 primary key 扫描来的要快呢？我们就需要了解innodb的 *clustered index* 和 *secondary index* 之间的区别了。 **使用“二级索引”扫描比“主键”扫描快的原因：**

innodb 的 clustered index 是把 primary key 以及 row data 保存在一起的，而 secondary index 则是单独存放，然后有个指针指向 primary key。因此，需要进行 count(*) 统计表记录总数时，利用 secondary index 扫描起来，显然更快。而primary key则主要在扫描索引，同时要返回结果记录时的作用较大，例如：

```
SELECT * FROM sbtest WHERE aid = xxx;
```

那既然是使用 secondary index 会比 primary key 更快，为何优化器却优先选择 primary key 来扫描呢，**Heikki Tuuri** 的回答是：

```
in the example table, the secondary index is inserted into in a perfect
order! That is
very unusual. Normally the secondary index would be fragmented, causing
random disk I/O,
and the scan would be slower than in the primary index.
I am changing this to a feature request: keep 'clustering ratio'
statistics on a secondary
index and do the scan there if the order is almost the same as in the
primary index. I
doubt this feature will ever be implemented, though.
```

详情请看：这个 bug，以及这篇文章：InnoDB Row Counting using Indexes。

最后感谢老杨的帮助。

评论

学习了，以前就知道

By 游客 *on 07* 二月 *2010 at about 21:44.*

学习了，以前就知道这个问题，只是不知道为啥用second快。。。这回长知识了

上面的测试信息为100万，不是1000万的吧

By 路过 *on 10* 七月 *2013 at about 11:42.*

上面的测试信息为100万，不是1000万的吧．另外现在的这个情况在最新版的5.5版本中仍然是这种现象么？

好细心，是100万没错，笔误了。

By yejr on 11 七月 *2013 at about 13:26.*

好细心，是100万没错，笔误了。
count(*)慢是InnoDB引擎MVCC特性导致，暂时还没有优化，不过可以从information_schema读取大致的记录，不建议做精确的count(*)

不错的建议！