DBTrain Lab 3 Report

刘泓尊 2018011446 计84

CI job ID #116023

Status	Job	Pipeline
passed	#116023 master 0f131ba2	#62698 by

Join算法

实现了2种Join算法：**Sort Merge Join** 和 **Hash Join**. 执行计划就是根据**数据量**和**是否存在索引**动态选择：

- 如果数据量较大且不存在索引，使用Hash Join
- 如果存在索引或者数据量较小，使用Sort Merge Join

只支持两表Join. 两种算法的实现都特别朴素，下面简要分析一下流程：

Sort Merge Join

Sort Merge Join 适合数据量较小或者列上存在索引的情况。

首先根据传入参数获得每个 PageSlotID 对应的 Record, 因为传入的时候已经对 PageSlotID 做了排序，所以实际上这里可以对Page做缓存，这样可以免去每次获得<PageID, SlotID>时重复构造页面的开销。

然后对所有 Record 对应的字段进行升序排序，并利用两个有序列表上的嵌套循环来得到结果。如果存在列上索引，则不需要进行排序，那么可以将该列作为查询列，根据参考列的 Field 值查询出另一列 PageSlotID 的列表，然后构造出结果。

不存在索引的时候，复杂度是$O(m+n+mlogm+nlogn)$； 存在索引的时候，复杂度是$O(mlogn)$, 进一步优化了性能。

Hash Join

Hash Join适合数据量大且不能利用索引的情况，可以克服Sort Merge Join的排序开销。

同 SortMergeJoin 一样，HashJoin 首先构造 Record* 列表。之后选择数据量较小的一列做 Hash：在我的实现里直接存进了 `unordered_map<Field, vector<Record*>>` 中，表示某一Field对应的 Bucket . 之后遍历另一个表的 Record , 从 HashTable 中拿出对应的 `vector<Record*>` ，构造新的 `Record*` 列表作为结果即可。复杂度是$O(m+n)$的。

测试结果

助教提供的测试`RandomSqlTest`的结果数量是$O(N^2/4)$的，所以无论是Sort Merge Join还是Hash Join, 构造结果（对应于SortMergeJoin的嵌套循环和HashJoin的最后遍历）的时候的复杂度都是平方量级的，这掩盖了排序或Hash操作的时间，使得结果的构造反而成了瓶颈。因此，我写了一个`RandomSqlTest2`,结果的数量是$O(N)$的，这样可以体现两种算法的性能，也可以测试更大的数据量。具体到实现，只需要把`SqlGenerator`的插入`id`改变一下:

```
// initial test
String sql = "INSERT INTO A VALUES(" + std::to_string(i / (data_num / 2)) + "," + "'" + alphanum[generator() % (sizeof(alphanum) - 1)] + "');";
// changed by me
String sql = "INSERT INTO A VALUES(" + std::to_string(data_num-i-1) + "," + "'" + alphanum[generator() % (sizeof(alphanum) - 1)] + "');";
```

之后运行测试，在`data_num = 50000,500000`的数据量下，运行`RandomSqlTest2`的时间如下:

dataNum\Time(ms)	Sort Merge Join	Hash Join
50000	1972	1846
500000	58513	18724

可以看到SortMergeJoin的复杂度基本是$O(nlogn)$增长的，而HashJoin保持线性。两者在50k数据量下速度差不多，因为小数据量下`sort`和`hash`的时间接近；但是500k的数据量明显Hash Join更胜一筹。

全部测试结果

```
Database Init.
Build Finish.
[==========] Running 5 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 5 tests from Lab3
[ RUN      ] Lab3.BasicJoinTest
[       OK ] Lab3.BasicJoinTest (5 ms)
[ RUN      ] Lab3.IndexJoinTest
[       OK ] Lab3.IndexJoinTest (0 ms)
[ RUN      ] Lab3.DuplicateJoinTest
[       OK ] Lab3.DuplicateJoinTest (0 ms)
[ RUN      ] Lab3.RandomTest
[       OK ] Lab3.RandomTest (146 ms)
[ RUN      ] Lab3.RandomTest2
[       OK ] Lab3.RandomTest2 (188 ms)
[----------] 5 tests from Lab3 (340 ms total)

[----------] Global test environment tear-down
[==========] 5 tests from 1 test suite ran. (340 ms total)
[  PASSED  ] 5 tests.

  YOU HAVE 1 DISABLED TEST
```