課題6

1029289895 尾崎翔太

2018/12/27

1　キャッシュ

1.1　高速化

id, name, price からなる表 test のデータを 10000000 件用意して

```
EXPLAIN ANALYSE SELECT id, name FROM test LIMIT 100;
```

という文を繰り返し実行した. 結果は table1.1 のようになった. 1 回目とそれ

	Planning Time	Execution Time
1 回目	268.038ms	16.364ms
2 回目	0.189ms	0.116ms
3 回目	0.167ms	0.107ms
4 回目	0.121ms	0.095ms
5 回目	0.150ms	0.090ms

表 1: 実行結果

以降では明らかに差があることが見て取れる.

1.2　キャッシュが効かないクエリ

様々なクエリを試してみた結果, キーではない属性について選択するようなクエリはあまり早くならないことがわかった. これは, 選択は条件式のチェックが入るので, データがキャッシュされていても時間がかかるのかなと思った. キー属性が早くなるのは, 索引が作られていることと関係があるのかなと思った.

2　索引の有無

section1 と同様のデータを 10000 行格納した表 test10000, 100000 行格納した表 test100000, 1000000 行格納した表 test1000000 を用いて

```
EXPLAIN ANALYSE SELECT * FROM (* 各表名 *) WHERE name = 'AGB';
```

という文を, 索引の有無を変えて実行した. 結果は table2.1 のようになった. 次に, 属性数が 2, 5, 10, 50 である表 test2, test5, test10, test50 を用いて, section1 と同じ文を実行した. 結果は table2.1 の通りである. 行数の場合のように多い方がむしろ早くなるということはなかったが, どれくらい早くなるかという点においては, 列数が多い方が効果が見られる.

	test10000	test100000	test1000000
索引なし Planning Time	199.770ms	108.441ms	207.552ms
索引なし Execution Time	245.271ms	2163.386ms	13707.996ms
索引あり Planning Time	243.257ms	162.324ms	388.317ms
索引あり Execution Time	138.596ms	357.031ms	71.504ms

表 2: 実行結果 (行数)

	test2	test5	test10	test50
索引なし Planning Time	224.698ms	94.312ms	145.610ms	537.425ms
索引なし Execution Time	2165.536ms	1258.916ms	1620.546ms	40167.935ms
索引あり Planning Time	216.681ms	182.589ms	210.753ms	834.038ms
索引あり Execution Time	53.714ms	67.667ms	67. 776ms	86.820ms

表 3: 実行結果 (列数)

3 選択率

次の文によって生成された表を用いる.

```
CREATE TABLE test_rate (id INTEGER, data INTEGER, PRIMARY KEY (id));
```

data に 0 から 999999 までの整数が重複なくランダムに格納されたような 1000000 行のデータを用いる. この表に対して, 以下の文を実行する.

```
EXPLAIN ANALYSE SELECT * FROM test_rate WHERE data < 200000;
EXPLAIN ANALYSE SELECT * FROM test_rate WHERE data < 400000;
EXPLAIN ANALYSE SELECT * FROM test_rate WHERE data < 600000;
EXPLAIN ANALYSE SELECT * FROM test_rate WHERE data < 800000;
```

上から順に, 選択率が 0.2, 0.4, 0.6, 0.8 の場合に対応している. 結果は table3.1 の通りである. 索引を付けた方が時間がかかるというよくわからない結果に

	0.2	0.4	0.6	0.8
索引なし Planning Time	92.786ms	92.751ms	170.530ms	226.060ms
索引なし Execution Time	2305.741ms	2217.160ms	2350.566ms	4772.773ms
索引あり Planning Time	678.754ms	478.278ms	989.707ms	689.508ms
索引あり Execution Time	9386.464ms	4876.238ms	12009.779ms	8265.277ms

表 4: 実行結果 (選択率)

なった. これまでと違って範囲選択であることが影響しているのかとも思ったが, B-tree は範囲選択にも対応しているはずなので, よくわからない. 選択

率については, 索引なしでは 0.8 の場合だけ時間がかかっている一方, 索引ありでは時間はバラバラなので, 結局選択率による差はあまりないと思った.

4　主索引と二次索引

主索引は主キーに関する索引で, 二次索引はそれ以外の属性に関する索引である. それだけの差なので, 性能には違いがなさそうであるが, 主キーは必ずユニークであるので, その点で性能がよい. 二次索引もユニークであったり, 適切なインデックスを定めたりすることで性能がよくなることが期待される.

5　参考文献

- 「二次索引」, ”https://www.kunihikokaneko.com/cc/db/7.html”, 2018/12/27