計算機科学実験及演習 4 データベース課題 6

1029259152 田中 勝也

1. 1 つの検索質問を設定して, 検索を繰り返して実行し, キャッシュが有効な状況とそうでない状況を示してください. また, キャッシュが効きにくい質問について考えてみてください.

今回は `users(id serial primary key, name char(50), email char(50))` という, ランダムに挿入された 100 万件のデータを持つ関係に対して, 以下の様なクエリを実行した際の予測時間, 実行時間を計測する.

```
select name, email from users where substring(name, 1, 2) = 'ab';
```

これは name 属性の先頭 2 文字が `ab` である組の name と email 属性を取り出すクエリである.
上記のクエリと `EXPLAIN` 文を用いて, キャッシュが無効な状態と有効な状態での予測時間, 実行時間を確認したところ, 以下のような結果になった.

	予測時間 (ms)	実行時間 (ms)
キャッシュ無効	11.420	807.261
キャッシュ有効	0.051	681.511

キャッシュが有効な状況・無効な状況で確かに実行時間に差が生じることが分かった.

またキャッシュが効きにくいようなクエリとして, データの取得ではない, UPDATE 文での更新などのクエリでは, キャッシュを利用できないのではないかと考えられる.
上記のテーブル `users` に対して, 次のクエリを用いて実際に時間を測定した結果, 以下のようになった.

```
update users set name = 'foo';
```

	予測時間 (ms)	実行時間 (ms)
キャッシュ無効	8.633	12393.314
キャッシュ有効	0.066	13165.991

`EXPLAIN` 文の結果から, 予測時間は短くなっているが, データの取得ではなく, 更新作業にかかる時間がオーバーヘッドとなっているため, キャッシュが有効であっ

ても実行時間は改善されないことがわかった.

2. ある属性に対する選択質問について, その属性に索引を構築している場合と構築してない場合の検索時間の違いを, 関係の組数を変化させて求めてください. その結果について考察してください. また, 関係の属性数, 組数や 1 つの組の大きさを変化させ, 索引が効果的となる状況について考察してみること.

今回は`users1(id serial primary key, name char(50), email char(50))`, `users2(id serial primary key, name char(50), email char(50), password char(50), age int, address char(100), zip char(10))` という 2 つの関係について, 組数を 100, 10000, 1000000 と変化させて, インデックスの有無による検索時間の変化を確認した.
検索クエリは

```
select name from users order by name limit 10;
```

となっている. name についてインデックスを作成した時, していない時で比較した結果が以下のようになった.

- `users1`
  単位: ms

組数 \ インデックス	なし	あり
100	0.699	0.099
10000	16.211	0.148
1000000	1173.894	0.069

- `usres2`
  単位: ms

組数 \ インデックス	なし	あり
100	0.487	0.069
10000	15.883	0.123
1000000	1250.230	0.839

結果から, 関係の属性数はあまり検索時間に影響を与えず, また, 組数が多いほどインデックスの影響が大きいと考えられる.

3. 質問を満たす組がただ 1 つの場合と, 満たす組の割合（選択率）$0 < k < 1$ を変化させた場合のそれぞれについて, 索引有無の実行時間を比較する.

今回は `users(id serial primary key, name char(50), email char(50))` という, ランダムに挿入された 100 万件のデータを持つ関係に対して, 次のようなクエリで適合率を変化させていく.
name にはランダムに生成したバイト列を 16 進数にエンコードしたものが入っているので, 質問を満たす組がただ 1 つの場合

```
select name, email from users where name = 'foo';
```

というクエリで, name がユニークなものを指定することにする. このクエリについて, インデックスを作成するとき, しないときで結果は以下のようになった.

	インデックスなし	インデックスあり
予測時間 (ms)	1.887	9.295
実行時間 (ms)	196.787	0.187

また, name の 1 文字目は 0-9, a-f のうちいずれかで, いずれになる確率も同様であると考えられるので, 選択率 $ k / 16$ のときは

```
select name, email from users where ascii(substring(name, 1, 1)) <= 47 + k; -- 0 <= k <= 10
select name, email from users where ascii(substring(name, 1, 1)) <=  86 + k; -- 11 <= k  <=
```

というクエリで表すことができる. k の値を適宜変化させたときの結果は以下のようになった.

- $k = 4$

	インデックスなし	インデックスあり
予測時間 (ms)	0.875	0.538
実行時間 (ms)	853.697	803.789

- $k = 8$

	インデックスなし	インデックスあり
予測時間 (ms)	0.607	0.535
実行時間 (ms)	912.176	725.819

- k = 12

	インデックスなし	インデックスあり
予測時間 (ms)	0.752	0.750
実行時間 (ms)	940.012	912.798

予測時間は適合率が低い方が改善されるが, 実行時間は適合率 0.5 程度で最も改善度が高いという結果となった.

4. (主索引と二次索引) 主索引と二次索引の性能の違いについて調査する.

`users(id serial primary key, name char(50), email char(50))` というランダムに挿入された 1 万件のデータをもつ関係について, id, name それぞれにインデックスを作成し, 以下のクエリの実行時間を調べる.

- id

```
select email from users where id < 5000;
```

- name

```
select name, email from users where ascii(substring(name, 1, 1)) <= 47 + k; -- 0 <= k <
```

どちらも適合率は 0.5 となる. 実行結果は以下のようになった.

	id	name
予測時間 (ms)	0.917	0.482
実行時間 (ms)	3.333	11.118

主索引のほうが性能が高いという結果となった. これは主索引は, 主キーで構築されている索引であるため, キーを指定するとレコードが一意に定まるが, 二次索引ではキーを指定してもレコードが一意に定まることはないため, より検索に時間がかかるからであると考えられる.