音楽管理・配信サービスのデータベースに実用的な索引を張る

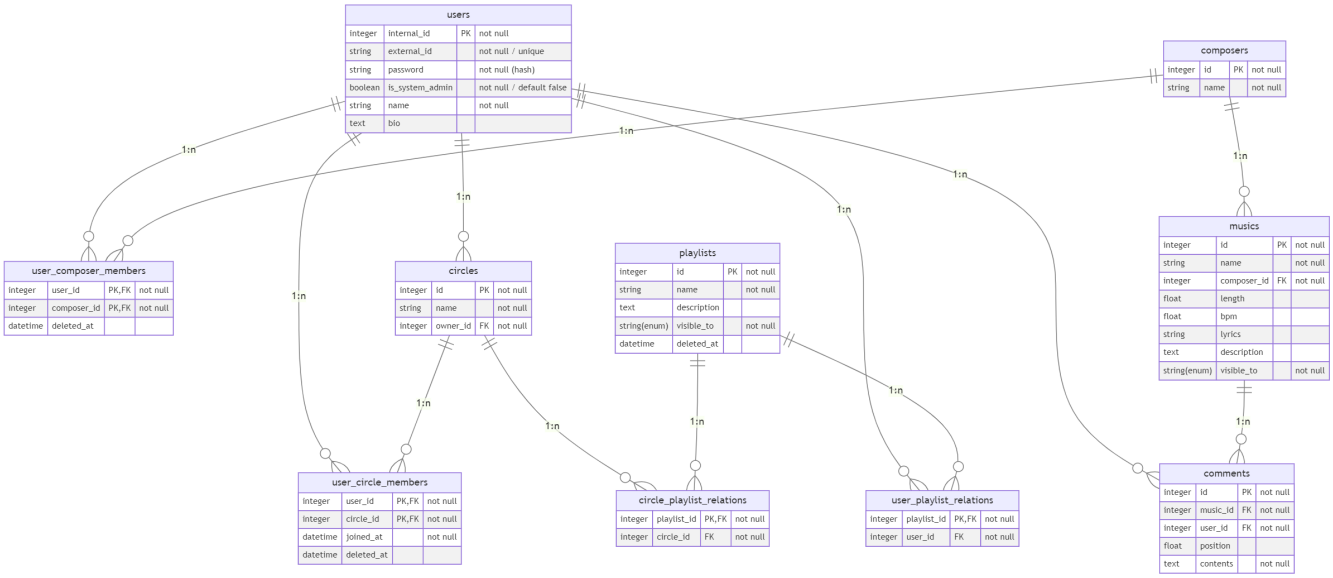
計算機科学実験及演習 4: データベース 課題 6 レポート

京都大学工学部情報学科 計算機科学コース 3 年生 王篤遥

学生番号: 1029332225, 提出日: 2023-10-26

論理モデル

課題 3 にて完成した関係スキーマは下図の通りです。



索引の検討

users 関係への索引

`users` 関係には、主キー以外に `external_id` という unique な属性があります。これはユーザーのログイン用 ID であり、Twitter 上でユーザーを検索する際に用いる ID と同じユースケースが想定されます。また、ユーザー名による検索も、サークルの会員管理やユーザー同士のつながりのために重要です。よって、両者には等号検索に強いハッシュ索引を構築します。

```
CREATE INDEX idx_users_external_id ON users USING hash (external_id);
CREATE INDEX idx_users_name ON users USING hash (name);
```

`string` 型への索引は、前方一致検索に対する検索速度を上げます。具体的には以下のようなクエリに対する処理速度が上がると考えられます。

```
SELECT * FROM users WHERE external_id LIKE 'Bmbo%';
SELECT * FROM users WHERE name LIKE 'Ao%';
```

これらのクエリは、ユーザーが他のユーザーの検索をしたい際に頻繁に実行されます。なお、ハッシュ索引の中でも拡張可能なハッシュや線形ハッシュを用いると、データの変更にも柔軟に対応できます。

user_circle_members 関係への索引

`user_circle_members` 関係は、複合キー `[user_id, circle_id]` が主キーであり、自動的に索引が張られています。しかし、クエリを発行する際には両方を指定した検索を行うケースが少なく、「あるサークルに所属するメンバー」を検索することが多いと想定されます。よって、`circle_id` 単体に対して、動的なデータの範囲検索・等号検索に強い B+ 木索引を構築します。

```
CREATE INDEX idx_user_circle_members_circle_id ON user_circle_members USING Btree
(circle_id);
```

この索引は、あるサークルに所属するメンバーを選択して一定の処理を行うようなクエリ全般に対して処理速度が高いです。以下のようなクエリの速度向上が期待できます。

```
SELECT users.external_id FROM (
    SELECT * FROM user_circle_members WHERE circle_id = 136
) LEFT JOIN users;
```

musics 関係への索引

`musics` 関係は、作曲者に対する検索が頻繁に行われることが想定されます。よって、`composer_id` に対して B+ 木索引を構築します。

```
CREATE INDEX idx_musics_composer_id ON musics USING Btree (composer_id);
```

この索引は、特定の作曲者の作品一覧を表示したい際に処理速度が高くなります。具体的に以下のようなクエリの速度向上が期待できます。

```
SELECT * FROM musics m WHERE m.composer_id IN (
    SELECT c.id FROM composers c WHERE c.name = 'Ao Takeuma'
);
```

効果が期待できない索引の例

効果が期待できない索引として、

- データ長の大きい属性への索引
- Cardinality の低い属性への索引

- 検索頻度が低い属性への索引

などが考えられます。例えば、`users.is_system_admin` に対する索引は、ユーザーがシステム管理者か否かのフィルタリングが頻繁に行われるとしても、索引を張るには至りません。Cardinality が `true` `false` の 2 つしかないためです。また、`users.bio` に対する索引は、`text` 型のデータ長が大きいため、索引を張ろうとしてもサイズが膨大になってしまう割には、ユーザーが自己紹介 (bio) に基づいて検索を行うようなユースケースがあまり想定されません。他にも、`musics.length` に対してハッシュ索引を張る、といったことも効果が期待できません。`float` 型である曲の「長さ」に対しては、等号検索をかけるよりも範囲検索をかけることが多く、ハッシュ索引が得意とするものではないためです。