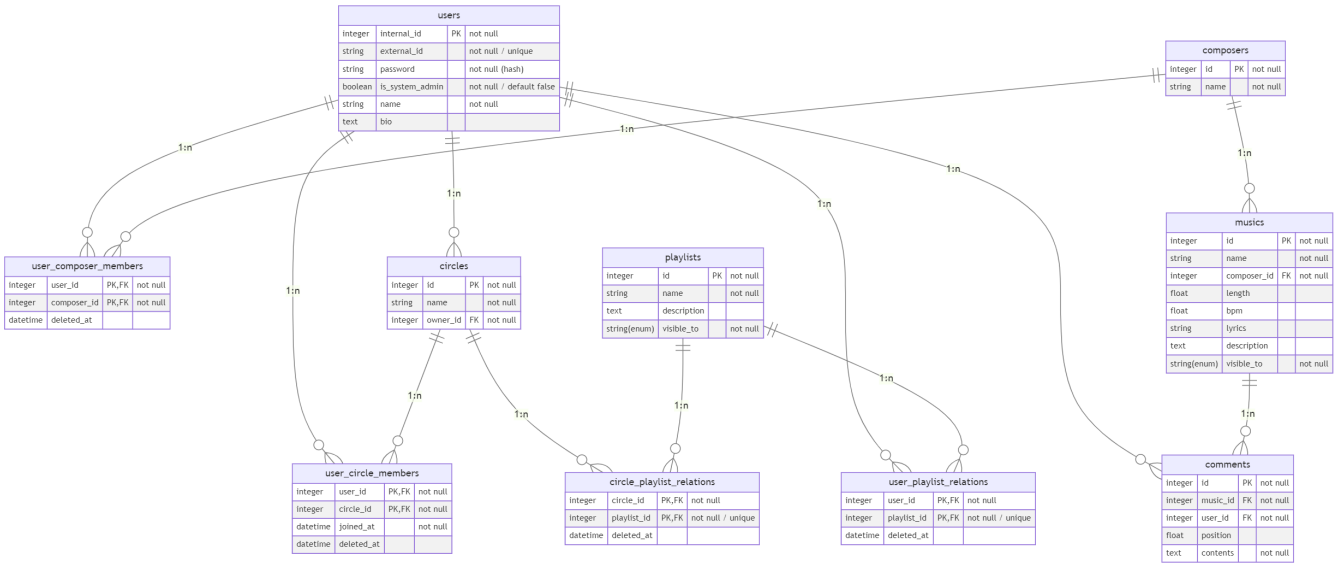
実用的な関係スキーマの正規化の演習及び考察

計算機科学実験及演習 4: データベース 課題 3 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: 2023-10-13

論理モデル

課題 2 にて完成した関係スキーマは下図の通りです。



関係スキーマにおける自明でない関数従属性集合

課題 2 にて、この関係スキーマにおける自明でない関数従属性を、従属性が成立する理由ごとに分類したものが以下の通りです。

主キーから他の属性への関数従属性

主キーはその定義から他の属性を一意に定めるので、自明でない関数従属性を持ちます。

- `users` において: `internal_id -> external_id, password, is_system_admin, name, bio`
- `composers` において: `id -> name`
- `circles` において: `id -> name, owner_id`
- `musics` において: `id -> name, composer_id, length, bpm, lyrics, description, visible_to`
- `playlists` において: `id -> name, description, visible_to`
- `comments` において: `id -> music_id, user_id, position, contents`
- `user_composer_members` において: `user_id, composer_id -> deleted_at`
- `user_circle_members` において: `user_id, circle_id -> joined_at, deleted_at`
- `user_playlist_relations` において: `user_id, playlist_id -> deleted_at`
- `circle_playlist_relations` において: `circle_id, playlist_id -> deleted_at`

非主キーで unique 制約が課されている属性から他の属性への関数従属性

unique 制約が課されている非主キーは候補キーなので、他の属性を一意に定め、自明でない関数従属性を持ちます。

- `users` において: `external_id -> internal_id, password, is_system_admin, name, bio`
- `circle_playlist_relations` において: `playlist_id -> circle_id, deleted_at`
- `user_playlist_relations` において: `playlist_id -> user_id, deleted_at`

その他の候補キーから他の属性への関数従属性

他にも候補キーがあれば、自明でない関数従属性を持ちますが、この関係スキーマには unique 制約で定めたもの以外に候補キーはありません。

候補キーではない属性から他の属性への関数従属性

候補キーではない属性から他の属性への関数従属性もありません。

関係スキーマにおける自明でない多値従属性集合

この関係スキーマにおける自明でない (関数従属性以外の) 多値従属性はありません。

関係スキーマの正規化

本節では、上記の関係スキーマがそれぞれの正規形の条件を満たすか判定して、必要に応じて正規化を行います。

第一正規形 (1NF)

この関係スキーマの各属性のデータ型は、`integer`, `string`, `boolean`, `text`, `datetime`, `float` のみであり、いずれも単純値です。よって、**第一正規形は満たされました**。

第二正規形 (2NF)

この関係スキーマは、課題 2 にて考察した通り、非主キーの属性から他の属性への関数従属性は

- `users` において: `external_id -> internal_id, password, is_system_admin, name, bio`
- `circle_playlist_relations` において: `playlist_id -> circle_id, deleted_at`
- `user_playlist_relations` において: `playlist_id -> user_id, deleted_at`

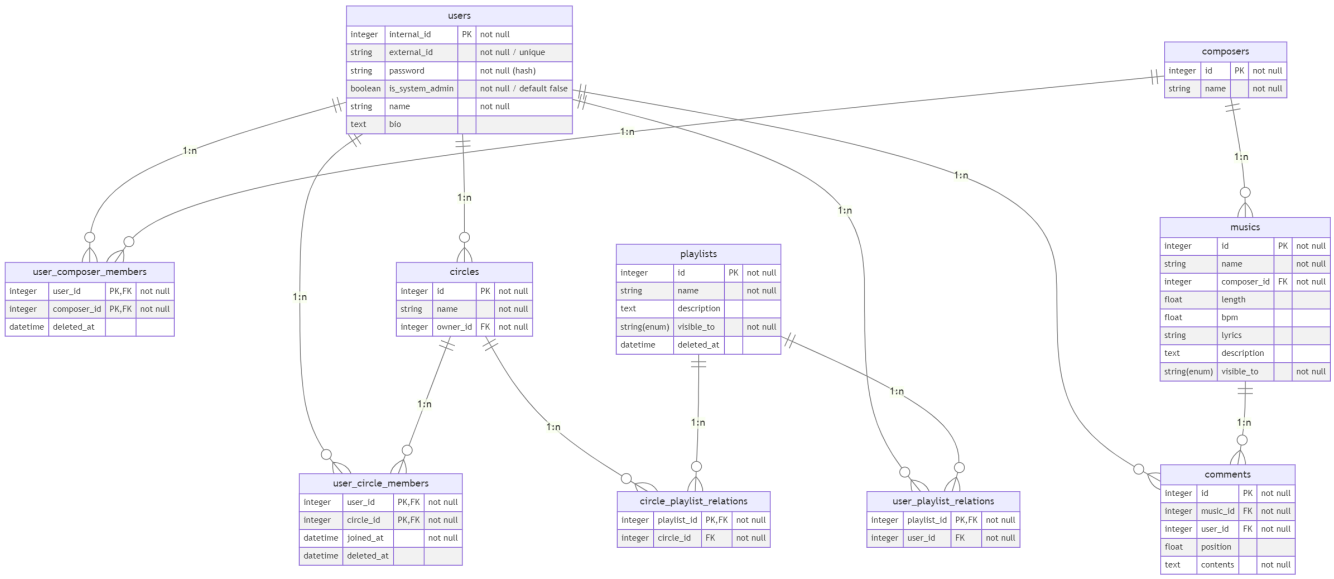
のみです。そのうち、関数従属性の左辺が候補キーの真部分集合であり、右辺が素属性でないものは

- `circle_playlist_relations` において: `playlist_id -> deleted_at`
- `user_playlist_relations` において: `playlist_id -> deleted_at`

の 2 つです。つまり、論理削除のためのフラグ `deleted_at` は、ここでは `circle_playlist_relations` などに保持されるべきではなく、`playlists` にて保持されるべきです。

`circle_playlist_relations` などは、そもそも `playlists` は個人に紐づくプレイリストとサークルに紐づくプレイリストとの多相性を持っていたことを解消するために作った中間テーブルです。よって、高々 1 つの個人あるいはサークルに紐づける以外の役割は担うべきでなく、`playlists` が担うべきです。

`playlists` は必ず `user_playlist_relations` か `circle_playlist_relations` のいずれかを 1 つだけ持つことに注意して、`deleted_at` を `playlists` の属性として持たせ、第二正規形を満たす関係スキーマを作り直したのが以下の通りです。



この関係スキーマにおいて非主キーの属性から他属性への関数従属性は

- `users` において: `external_id -> internal_id, password, is_system_admin, name, bio`

のみとなり、**第二正規形は満たされました**。

第三正規形 (3NF), Boyce-Codd 正規形 (BCNF)

この関係スキーマにおける非主キー属性から他属性への関数従属性は上記に挙げたもののみです。 ところで `external_id` は候補キーなので、超キーです。したがって、**第三正規形、Boyce-Codd 正規形は満たされました**。

第四正規形 (4NF)

この関係スキーマには自明でない多値従属性がありません。したがって、**第四正規形は満たされました**。

第五正規形 (5NF)

この関係スキーマには、自明でない結合従属性が数多く存在する。3 つ以上の属性を持つ関係について、主キーとその他の 1 属性という 2 属性の関係に情報無損失分解できてしまう。しかしそのような分解に意味はないので、**第五正規形は満たされないが、分解はしない**こととします。

以上より、再設計された関係スキーマは上図の通りであり、第四正規形が得られました。