Ткаченко Григорий, 4537

Курсовая работа по базам данных.
База данных для пользовательского приложения
“Музыкальный плеер”.

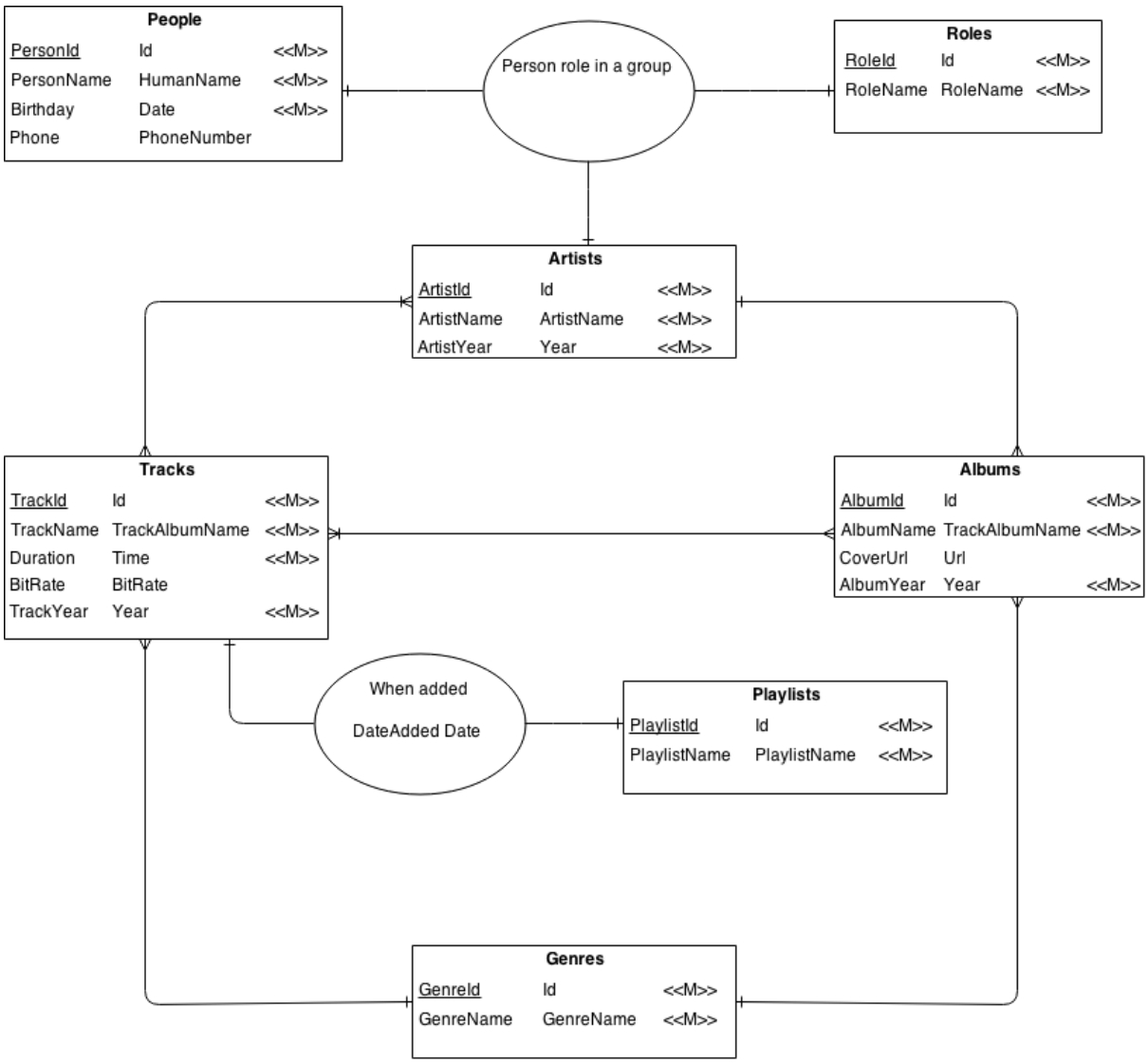
PostgreSQL 9.3.5

Тестирование

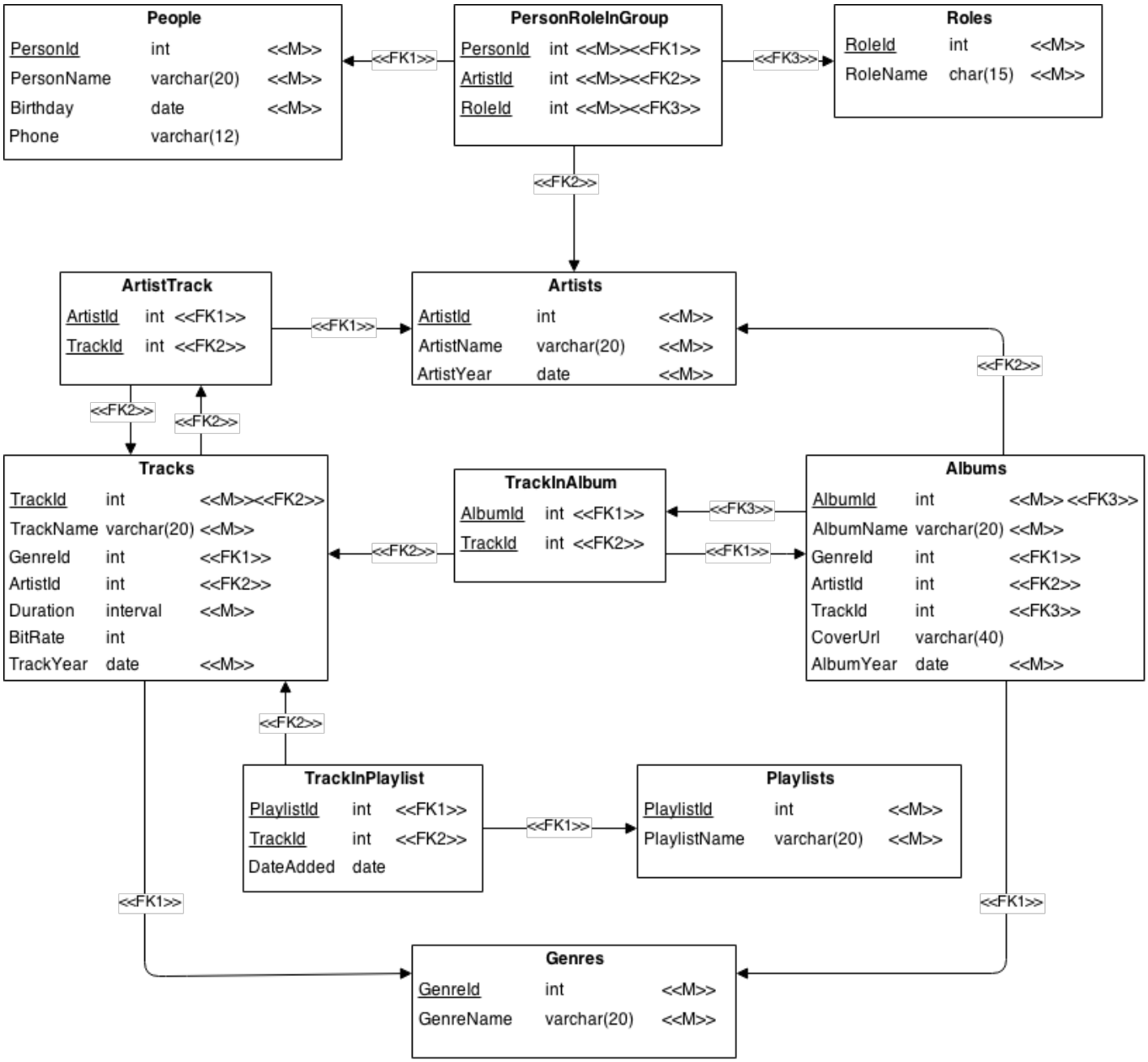
После запуска psql (9.3.5) в интерактивном режиме:

```
\i create_db.sql        -- drop the existing db and create the new one
\i run.sql              -- run init_db.sql and fill_data.sql
\i test.sql             -- some testing stuff
```

ER-model.



Physical model.



Анализ модели

Выпишем все функциональные зависимости для каждого отношения

PersonId -> PersonName, Birthday, Phone
RoleId -> RoleName
RoleName -> RoleId
ArtistId -> ArtistName, ArtistYear
GenreId -> GenreName
GenreName -> GenreId
TrackId -> TrackName, GenreId, Duration, Bitrate, TrackYear
AlbumId -> AlbumName, GenreId, ArtistId, CoverUrl, AlbumYear
PlaylistId -> PlaylistName
PlaylistName -> PlaylistId
PlaylistId, TrackId -> DateAdded

Заметим следующее

- В каждом отношении атрибуты атомарны, нет повторяющихся групп и есть ключ. То имеем 1НФ
- Не существует неключевых атрибутов, зависящих от части ключа. То имеем 2НФ
- Для каждого отношения верно, что не существует неключевых атрибутов, транзитивно зависящих от ключа. То имеем 3НФ

Описание сущностей.

- *People* - описание реальных людей, которые являются частью каких-то музыкальный групп
- *Roles* - описание роли человека в группе
- *Artists* - описание музыкальных групп. При этом человек, выступающий один и под своим именем, будет также иметь свое описание как *Artist*
- *Albums* - описание альбомов. Будем считать, что одному альбому соответствует только *один* исполнитель. (в жизни не всегда так, но для некоторого разнообразия модели будем считать так)
- *Tracks* - описание трека. Будем считать, что трек может быть исполнен совместно несколькими артистами.
- *Playlists* - описание пользовательских плейлистов
- *Genres* - описание жанров (для треков и альбомов будем рассматривать один доминирующий жанр)
- *PersonRoleInGroup* - описание роли реального человека в определенной группе
- *ArtistTrack* - набор записей для треков и их соответствию артистам
- *TrackInAlbum* - набор записей для треков и их соответствию альбомам
- *TrackInPlaylist* - набор записей для треков и их соответствию плейлистам

Реализованные ограничения

- Ограничения на длины всех имен, названий итп
- Проверки на уникальность названий плейлистов и жанров
- Ограничения на значение BitRate
- Количество треков в плейлисте не превышает 50
- Год выпуска альбома должен быть больше года создания группы
- Все треки альбома должны иметь тот же жанр, что и у самого альбома

Прочие триггеры

- Триггер для автоматической установки времени добавления трека в плейлист
- Триггер для обновления materialized view

Полезные функции

- Получение списка треков определённого альбома
- Получение списка реальных людей, принимавших участие в создании альбома
- Получение списка артистов, участвовавших в создании трека
- Получения всех треков определённого плейлиста
- Умное слияние плейлистов. Для двух переданных плейлистов выбирается наиболее релевантный (среднее количество вхождений добавленных недавно треков плейлиста во все плейлисты) и все треки оставшегося плейлиста перекидываются в него.
- Вспомогательные функции для триггеров

View & materialized view

- Представления для недавно добавленных треков (менее 5 дней назад) и их количество вхождения в плейлисты (одно представление с идентификаторами, а другое с именами)
- Материализованное представления для самых популярных треков из недавних (входящих в наибольшее число плейлистов)

Индексы

К нашей можно придумать несколько запросов, которые могут быть частыми (для нужд нашего пользовательского приложения “Плеер”). Например, это будут следующие запросы:

- Все треки определенного исполнителя
- Все треки определенного альбома
- Все треки определенного плейлиста

- Выбрать все треки плейлиста за последние две недели

Заметим, что так как в PostgreSQL для каждого отношения верно, что существует индекс по PRIMARY KEY и по всем его префиксам, то индексы для первых трех запросов уже созданы автоматически. Тогда осталось добавить индекс для последнего запроса.

Поскольку в запросе может фигурировать диапазон, то будем использовать в качестве структуры данных B-tree (по умолчанию в PostgreSQL). Другим аргументом за такой выбор может послужить рекомендация разработчиков PostgreSQL не использовать hash-table (http://www.postgresql.org/docs/9.1/static/indexes-types.html)

То есть имеем:

create index RecentTracksInPlaylist on TrackInPlaylist(DateAdded);