САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Прикладной Математики и Процессов Управления

Курсовая работа по дисциплине

«БАЗЫ ДАННЫХ И СЕТЕВЫЕ ТЕХНОЛОГИИ»

на тему:

«ДИСКОГРАФИЯ МУЗЫКАЛЬНЫХ ИСПОЛНИТЕЛЕЙ»

Выполнила: Бронникова В.Е. 205 группа
Преподаватель: Криммель Г.К.

Санкт-Петербург

2018

Содержание

1. Описание и Схема БД

    1.1. Предметная область

    1.2. Сущности и их атрибуты

    1.3. Связи между таблицами

    1.4. Визуализация схемы БД

2. Запросы

    2.1. Простые запросы:

    2.2. Средние запросы:

    2.3. Сложные запросы

1. Описание и Схема БД

1.1. Предметная область

База данных описывает предметную область «Дискография музыкальных исполнителей», которая содержит в себе общую информацию об исполнителях, их альбомах и треках в них, а так же музыкальных жанрах, к которых исполнители выступают. Исполнители характеризуются именем, годом начала творчества, описанием, жанрам, в которых выступают, ссылкой на сайт. Жанры - именем и описанием. Альбомы - названием, автором, описанием, годом, рейтингом, количеством треков. Треки именем, альбомом, длиной, текстом композиции.

1.2. Сущности и их атрибуты

В скобках приведены соответствующие названия таблиц физической модели и их поля:

1) Исполнитель (Artist)
   - ID исполнителя (artistId) - первичный ключ, уникальный номер
   - имя (name) - имя исполнителя или название группы
   - год (year) - год начала активности
   - описание (description) - общая информация об исполнителе
   - сайт (website) - ссылка на сайт исполнителя
2) Жанр (Genre)
   - ID жанра (genreId) - первичный ключ, уникальный номер
   - название (name) – название жанра
   - описание (description) – описание жанра
3) Альбом (Album)
   - ID альбома (albumId) - первичный ключ, уникальный номер
   - ID исполнителя (artistId) - внешний ключ к сущности «Исполнитель»(таблице Artist)
   - название (name) - название альбома
   - описание (description) – общее описание альбома
   - год (year) - год выхода альбома
   - рейтинг(rating) - рейтинг альбома
   - количество треков (numberOfTracks) – количество треков в альбоме
4) Трек (Track)
   - ID трека (trackId) - первичный ключ, уникальный номер
   - ID альбома (albumId) – ID альбома, которому принадлежит трек, внешний ключ к сущности «Альбом» (таблице Album)
   - название (name) – название композиции
   - длительность (trackLength) – длительность трека
   - текст (lyrics) – текст композиции

1.3. Связи между таблицами

Каждый исполнитель может работать в нескольких жанрах, в свою очередь, в каждом жанре могут выступать несколько исполнителей. Это описывает связь многие-ко-многим (m:m), поэтому для корректного формирования такой связи создается дополнительная таблица ArtistGenre, которая состоит из первичных ключей таблиц Artist и Genre и разбивает связь (m:m) на две связи (1:m).
У одного альбома может быть только один автор, но у одного исполнителя может быть несколько альбомов (для уменьшения количества таблиц), поэтому между таблицами «Album» «Artist» и «Album» «Track» поддерживается связь (m:1).

1.4. Визуализация схемы БД

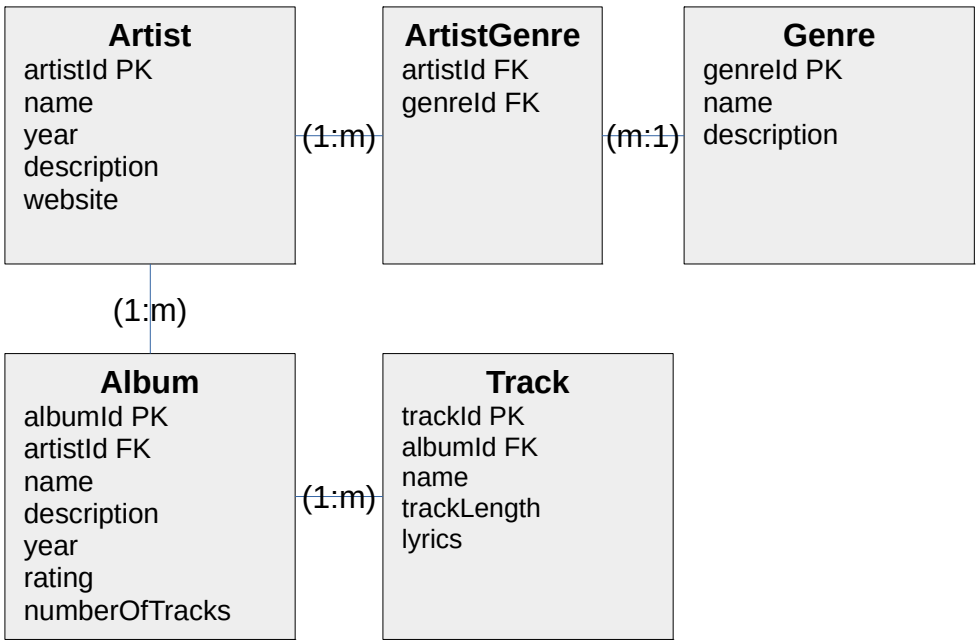

Рис. 1 Схема базы данных

2. Запросы

2.1. Простые запросы:

1. Альбомы, выпущенные за последние 10 лет

```
SELECT albumid AS album_id, name AS album_name, year AS album_year
FROM album WHERE year >= (date_part('year'::text,
CURRENT_DATE))::integer - 10;
```

2. Исполнители, начавшие свою деятельность после 2000 года

```
SELECT artistid AS artist_id, name AS artist_name, year AS artist_year FROM
artist WHERE year > 2000;
```

3. Исполнители, сайты которых расположены на домене .com

```
SELECT artistid AS artist_id, name AS artist_name, website AS artist_website
FROM artist WHERE website LIKE '%.com';
```

4. Треки длинной от 3 до 4 минут

```
SELECT trackid AS track_id, name AS track_name, to_char(tracklength, 'MI:SS')
AS track_length FROM track WHERE tracklength BETWEEN INTERVAL '3
min' AND '4 min';
```

2.2. Средние запросы:

1. Выводит средний рейтинг альбомов каждого исполнителя, который начал свою активность после определенного года (не включительно). Результат выводит, округляя до 1 знака после запятой.

Значения: до 2015

```
\set yrvar 2010
SELECT DISTINCT artist.name AS artist_name,
ROUND( CAST(avg(album.rating)
OVER (PARTITION BY artist.name) as numeric), 1) AS album_avg_rating
FROM album
INNER JOIN artist ON artist.artistid=album.artistid
WHERE artist.year > :yrvar;
```

Наиболее ресурсозатратные операции запроса: Seq_Scan по artist (WHERE) и Hash Join (INNER JOIN)

Оптимизация:
Создание индексов
CREATE INDEX artist_artistid_idx ON artist(artistid);
CREATE INDEX artist_year_idx ON artist(year);

2. Считает количество исполнителей, выступающих в определенном жанре.
Значения: от 1 до 8

```
\set gidvar 5
SELECT name AS genre_name, COUNT(artistgenre.genreid) AS
artistgenre_number_of_artists
FROM artistgenre
INNER JOIN genre ON (artistgenre.genreid=genre.genreid)
GROUP BY genre.name, artistgenre.genreid
HAVING (artistgenre.genreid) = :gidvar;
```

Наиболее ресурсозатратные операции запроса: Seq_Scan по genre (HAVING) и Hash Join (INNER JOIN)
Оптимизация:
Создание индексов
CREATE INDEX genre_genreid_idx ON genre(genreid);
CREATE INDEX artistgenre_genreid_idx ON artistgenre(genreid);

3. Выводит максимальный рейтинг альбомов исполнителей, чье имя начинается на определенную букву.
Значения: "'r/b/c/m/j/h/n/f/t/l/a/p/f%'" любой регистр

```
\set namevar "'L%'"
SELECT DISTINCT artist.name AS artist_name, MAX(album.rating)
OVER (PARTITION BY artist.name) AS album_max_rating
FROM artist, album
WHERE artist.artistid = album.artistid AND artist.name ILIKE :namevar;
```

Наиболее ресурсозатратные операции запроса: Seq_Scan по artist для WHERE
Оптимизация:
Создание индекса CREATE INDEX artist_name_idx ON artist(name text_pattern_ops);

2.3. Сложные запросы

1. Считает средний рейтинг альбомов в определенном жанре, Выводит, округляя до одного знака после занятой.

Допустимые значения: от 1 до 8

```
\set gidvar 5
CREATE VIEW genre_album_avg AS
  SELECT arg.genreid as genreid, arg.artistid as artistid, alb.rating AS
album_avg
              FROM album AS alb
              INNER JOIN artistgenre AS arg
            ON arg.artistid=alb.artistid
            WHERE arg.genreid = :gidvar;

SELECT DISTINCT gen.name AS genre_name,
     ROUND(CAST(AVG(album_avg)
                OVER (PARTITION BY genre_album_avg.genreid) as
numeric), 1) AS genre_avg
  FROM genre_album_avg
     INNER JOIN genre AS gen
     ON genre_album_avg.genreid=gen.genreid;
```

2. Выводит список исполнителей, похожих на исполнителя композиции (выступающих в тех же жанрах, что и автор композиции, хотя бы один жанр совпадает)
Допустимые значения: 1-39

```
\set tval 18
SELECT artist.name AS artist_name FROM artist
INNER JOIN artistgenre ON artist.artistid = artistgenre.artistid
WHERE artistgenre.genreid IN (SELECT artistgenre.genreid FROM
artistgenre
WHERE artistgenre.artistid = (SELECT album.artistid
FROM album
INNER JOIN track
ON track.albumid = album.albumid
WHERE track.trackid = :tval));
```

Наиболее ресурсозатратные операции запроса: Seq_Scan по track (WHERE) и Hash Join (JOIN). Для оптимизации создаем индексы
CREATE INDEX track_trackid_idx ON track(trackid);
CREATE INDEX album_albumid_idx ON album(albumid);
CREATE INDEX artistgenre_artistid_idx ON artistgenre(artistid);

CREATE INDEX artistgenre_genreid_idx ON artistgenre(genreid); (уже существует после предыдущего запроса)

3. Отображает количество треков, которые выпустил каждый исполнитель после определенного года.
Допустимые значения: до 2018 (1999, 2005, 2012, 2016)

```
\set yrval 1999
SELECT artist.name as artist_name,
coalesce(SUM(t_count.album_num_tracks) OVER (PARTITION BY
artist.name), 0) as num_of_tracks FROM artist
LEFT JOIN
(SELECT album.artistid, COUNT(track.albumid)  AS album_num_tracks
FROM track
RIGHT JOIN album ON (album.albumid=track.albumid) WHERE
album.year >:yrval
GROUP by album.artistid) AS t_count
ON t_count.artistid = artist.artistid;
```

Наиболее ресурсозатратные операции запроса: Seq_Scan по album (WHERE) и Hash Join (JOIN). Для оптимизации создаем индексы
CREATE INDEX album_year_idx ON album(year);
CREATE INDEX track_albumid_idx ON track(albumid);
CREATE INDEX artist_artistid_idx ON artist(artistid);