САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

Гордеев Юрий Борисович

Хип-хоп исполнители

(Hip-hop artists)

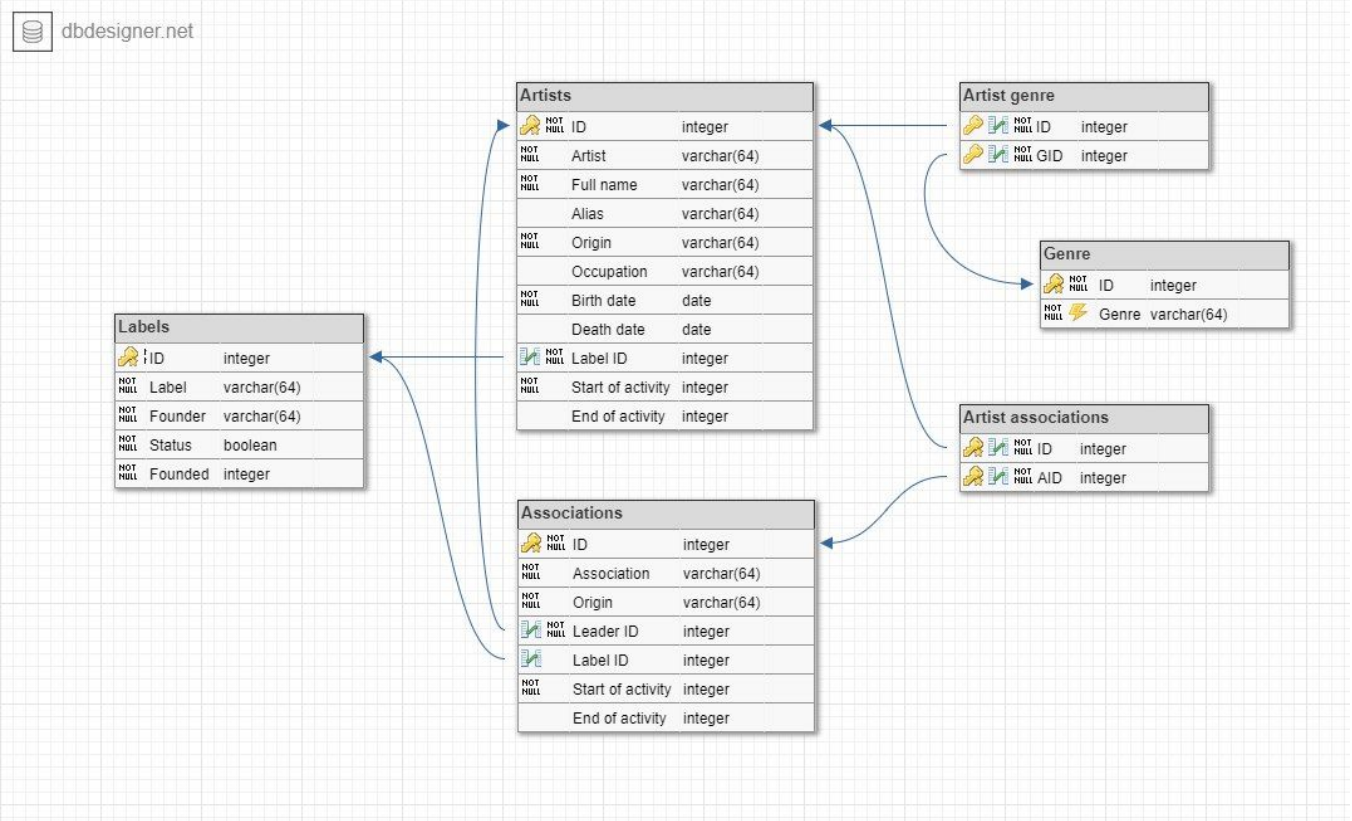
Преподаватель ***Филиппов Р.О.***

Санкт-Петербург

2017

Содержание

1. Схема базы данных



2. Описание базы данных

В базе данных **“Хип-хоп исполнители”** описываются представители хип-хоп культуры, а также связанная с ними информация. За основу взяты исполнители разного формата: разных эпох, жанров, стран, чтобы привнести уникальность и разнообразие отношений в базу данных.

База данных может быть полезна как обычному обывателю, желающему узнать больше о своем любимом исполнителе, так и сведущему в хип-хоп культуре журналисту или аналитику, ищущему самую необходимую информацию о артистах, лэйблах или ассоциациях.

Все данные скомпонованы и отредактированы так, чтобы любой пользователь, от новичка до эксперта, мог легко и доступно получить все необходимые сведения.

Проект загружен на веб-сервис для хостинга IT-проектов - **GitHub**.
Ссылка на репозиторий: https://github.com/4str0n4ut/Hip-hop-database

Взаимоотношения объектов

Реализованы отношения:
1:1 – *1 artist, 1 association* (в плане того, что один исполнитель является лидером группы/коллаборации);
1:m – *1 label, m artists* (на лейбл может быть подписано несколько артистов);
m:m – *m artists, m genres* (несколько артистов могут писать музыку одного жанра, так и один артист может писать музыку в разных жанрах).

Описание полей таблиц

Таблица Artists (Исполнители):

В этой таблице представлена самая необходимая и важная информация об исполнителях.

- **ID** – идентификатор с параметрами ***primary key*** и ***autoincrement***, использован тип *integer* для возможности дальнейшего дополнения таблицы (поскольку на данный момент в мире существует огромное количество хип-хоп исполнителей);

- **Artist** – имя исполнителя (чаще всего псевдоним), использован тип *varchar(64)* и ***unique key***, т.к. имя исполнителя уникально;

- **Start of activity/End of activity –** годы активности исполнителя (от начала до завершения карьеры, **End of activity** может принимать значение *NULL*), взят тип *smallint*, так как он достаточно прост и удобен для вычислений, чаще всего начало и конец карьеры исполнителей определяются с точностью года, а также его достаточно для хранения дат в формате **year**;

- **Full name** – полное (настоящее) имя артиста типа *varchar(64)*, т.к. большинство исполнителей выступает под псевдонимами;

- **Alias** – псевдонимы (при наличии, т.е. может принимать значение *NULL*), тип *varchar(64),* использовано ограничение ***unique key***; отличие от поля **Artist** в том, что у исполнителя может быть несколько псевдонимов, помимо основного;

- **Origin** – место рождения исполнителя (чаще всего в формате *“город, штат, страна”*, т.к. большинство хип-хоп исполнителей из Соединенных Штатов) типа *varchar(64)*;

- **Occupation(s)** – другие виды деятельности, помимо музыкальной карьеры, тип *varchar(64)*;

- **Birth/Death date** – дата рождения/смерти (последняя также может принимать значение *NULL*), взят тип *date,* т.к. здесь необходим более точный тип данных для хранения дат, в отличие от **Start of activity/End of activity**;

Таблица Labels (Лэйблы):

Лэйблы – брэнд, созданный звукозаписывающими компаниями, на которые подписываются музыкальные исполнители.

- **ID** – идентификатор с параметрами ***primary key*** и ***autoincrement***, использован тип *integer*, так же как и в таблице **Artists**;

- **Label** – наименование лэйбла, использовано ограничение ***unique key*** и тип данных *varchar(64)*, т.к. имена лэйблов уникальны;

- **Founder** – основатель лэйбла, тип данных *varchar(64)*;

- **Founded** – дата основания лэйбла типа *smallint*, поскольку некоторые звукозаписывающие компании были основаны в первой половине XX века, и не имеют точной даты основания помимо года;

- **Status** – статус лэйбла типа *boolean* (булева переменная, принимает значение *«true»* если лэйбл продолжает свою деятельность, и, соответственно, *«false»*, если лэйбл ее уже прекратил);

Таблица Associations (Ассоциации):

Ассоциации – различные творческие объединения исполнителей, наподобие группы, дуэта, коллаборации и т.д.

- **ID** – идентификатор с параметрами ***primary key*** и ***autoincrement***, использован тип *integer*, ак же как и в таблице **Artists**;

- **Association** – наименование ассоциации с ограничением ***unique key***, тип данных *varchar(64)*;

- **Origin** – место основания ассоциации (чаще всего в формате *"город, штат, страна"*) типа *varchar(64)*;

- **Start of activity/End of activity –** годы активности ассоциации (**End of activity** может принимать значение *NULL*), взят тип *smallint*, так как он достаточно прост и удобен для вычислений, а также его достаточно для хранения дат в формате **year**;

- **Leader ID –** идентификатор лидера группы типа *integer*, который ссылается при помощи ограничения ***foreign key*** на **ID** артиста в таблице **Artists**;

- **Label ID** – идентификатор лэйбла типа *integer* (группы, так же, как и исполнители, могут записываться на лэйбле, хотя есть и такие, которые предпочитают развиваться самостоятельно, именно поэтому поле может принимать значение *NULL*);

Таблица Artist associations (Ассоциации артистов):

Ассоциации артистов – данная таблица устанавливает отношения между артистами и и ассоциациями, в которых они состоят:

- **Artist ID** – идентификатор исполнителя с параметрами ***primary key*** и ***autoincrement*** типа *smallint*;

- **Associations ID** – идентификатор ассоциации с параметрами ***primary key*** и ***autoincrement*** типа *smallint*;

<u>Таблицы Genre/Artist genre (Жанры/Жанры артистов):</u>

Жанры/Жанры артистов – здесь таблица «Жанры» присваивает жанрам идентификаторы, а «Жанры артистов» - устанавливает отношения между артистами и жанрами:

- **Artist ID** – идентификатор исполнителя с параметрами ***primary key*** и ***autoincrement*** типа *smallint*;

- **Genre ID** – идентификатор жанра с параметрами ***primary key*** и ***autoincrement*** типа *smallint*;

- **Genre** – название музыкального жанра, тип *varchar(64)* и ограничение ***unique key***.

3. Легкие запросы

1. Выводит информацию о лейблах, которые всё еще продолжают свою деятельность (здесь “TRUE” значит “активен”) и сортирует их по дате основания.

```
SELECT label, founder, founded

FROM labels AS l

WHERE l.status = 'TRUE'

ORDER BY l.founded;
```

Необходимость:
Запрос позволяет получить информацию о активных лэйблах - это может быть полезно при сборе информации о исполнителях и группах, ведущих свою музыкальную деятельность, а также принесет пользу начинающему исполнителю, если он захочет заключить контракт с какой-либо звукозаписывающей корпорацией.

Оптимизация:
Были добавлены индексы labels_founded_idx и labels_status_idx для фильтрации лэйблов по дате их основания и статусу соответственно.

```
CREATE INDEX ON labels(status);
CREATE INDEX ON labels(founded);
```

Используется labels_pkey для оптимизации объединения таблиц.

2. Выводит информацию об артистах, которые начали свою карьеру в 1980-2000ых годах и всё ещё продолжают свою деятельность.

```
SELECT artist, full_name, origin, occupation, birth_date,
start_of_activity
FROM artists AS a
WHERE (a.start_of_activity BETWEEN '1980' AND '2000') AND
(a.end_of_activity ISNULL)
ORDER BY a.start_of_activity;
```

Необходимость:

Данный запрос будет полезен журналистам, т.к. позволяет собрать информацию об исполнителях, начавших свою деятельность в определенной эпохе, ставших движущей силой хип-хоп культуры в это время. Это поможет отследить их карьерный рост, рост популярности, определить влияние как на хип-хоп, так и на культуру в целом.

Оптимизация:

Были добавлены индексы artists_end_of_activity_idx, artists_start_of_activity_idx для фильтрации артистов по дате старта и окончания карьеры.

```
CREATE INDEX ON artists(start_of_activity);
CREATE INDEX ON artists(end_of_activity);
```

Используется artists_pkey для оптимизации объединения таблиц.

3. Выводит информацию об ассоциациях, расположившихся в 'Los Angeles, California, U.S.' или 'Long Beach, California, U.S.', которые завершили свою деятельность.

```
SELECT association, origin, start_of_activity

FROM associations AS a

WHERE (a.origin = 'Los Angeles, California, U.S.' OR a.origin = 'Long Beach,

California, U.S.')

AND a.end_of_activity NOTNULL;
```

Необходимость:

Запрос позволяет получить информацию о так называемых “отцах” хип-хопа Восточного побережья (East coast hip-hop), т.е. позволит узнать о основателях этого жанра, их карьерном пути и влиянии на хип-хоп.

Оптимизация:

Был добавлен индекс associations_leader_id_idx, associations_origin_idx для осуществления фильтрации ассоциаций по их лидеру и месту основания

```
CREATE INDEX ON associations(origin);

CREATE INDEX ON associations(end_of_activity);
```

Используется associations_pkey для оптимизации объединения таблиц.

4. Выводит информацию о живых исполнителях, родившихся раньше 90`ых годов, а также являющихся звукозаписывающими продюсерами.

```
SELECT *

FROM artists AS a

WHERE a.occupation = ('record producer') AND a.death_date ISNULL AND

a.birth_date <= '1990-1-1';
```

Необходимость:

Исполнители, выводимые этим запросом, имеют немалый опыт в хип-хоп среде, и могут быть полезны для новичков в качестве звукозаписывающих менторов. Запрос позволит получить всю необходимую информацию о таких исполнителях.

Оптимизация:

Были добавлены индексы artists_birth_date_idx, artists_death_date_idx, artists_occupation_idx для фильтрации исполнителей по дате рождения/смерти и роду деятельности.

```
CREATE INDEX ON artists(occupation);

CREATE INDEX ON artists(death_date);

CREATE INDEX ON artists(birth_date);
```

Используется artists_pkey для оптимизации объединения таблиц.

4. Средние запросы

Представляемые далее запросы будут полезны и интересны человеку, разбирающемуся в хип-хоп культуре.

1. Выводит исполнителей в жанре гэнгста-рэп (gangsta rap).

```
SELECT a.id, a.artist, g.genre
FROM artists a
 INNER JOIN artist_genre ON a.id = artist_genre.artist_id
 INNER JOIN genre g ON g.id = artist_genre.genre_id
WHERE g.genre = ('gangsta rap')
ORDER BY a.artist;
```

Необходимость:

В зависимости от выбранного жанра (в данном случае - гэнгста-рэп), запрос позволяет отобразить всех исполнителей в данном жанре. Это может быть полезно человеку, ищущему для прослушивания новых исполнителей в определенном жанре.

Оптимизация:

Помимо уже существовавших индексов genre_pkey, genre_genre_key, artist_genre_pkey, artists_pkey, новых добавлено не было.

2. Выводит информацию об исполнителях с лэйблов, продолжающих свою деятельность, и которые начали свою деятельность до основания лэйбла.

```
SELECT a.artist, a.start_of_activity, l.label, l.founder, l.founded
FROM artists a
  JOIN labels l ON a.id = l.id
WHERE l.status NOTNULL AND a.start_of_activity < l.founded
ORDER BY l.founded;
```

Необходимость:

Запрос интересен тем, что отображает исполнителей, начавших свою карьеру “с низов”, т.е. не подписавших с самого начала карьеры контракт со звукозаписывающей студией.

Оптимизация:

Были добавлены индексы labels_founded_idx, labels_status_idx для фильтрации лэйблов по дате основания и статусу.

```
CREATE INDEX ON labels(status);

CREATE INDEX ON labels(founded);
```

Также используются уже созданные индексы artists_pkey, artists_start_of_activity_idx, labels_pkey.

3. Выводит информацию об исполнителе ассоциации и её лидере, с условием что исполнитель родился в том же городе, в котором была основана ассоциация.

```
SELECT art.artist, ass.association, ass.origin, art2.artist AS leader
FROM artists art
 INNER JOIN artist_associations ON art.id = artist_associations.artist_id
 INNER JOIN associations ass ON artist_associations.associations_id = ass.id
 INNER JOIN artists art2 ON ass.leader_id = art2.id
WHERE art.origin = ass.origin;
```

Необходимость:

Запрос интереснее тем, кто знаком с хип-хоп культурой и тем, как много значит для исполнителей и фанатов “лояльность” своей родине. Так как хип-хоп тесно связан с преступностью и бандами, переезд в другой город или другую часть страны (например, с Западного побережья на Восточное) могли счесть предательством. Данный запрос выводит именно таких, “лояльных” исполнителей.

Оптимизация:

Был добавлен индекс associations_leader_id_idx для фильтрации ассоциаций по их лидеру.

```
CREATE INDEX ON associations(leader_id);
```

Также используются уже созданные индексы artists_pkey, artist_associations_pkey, associations_pkey, labels_pkey, associations_origin_idx.

5. Сложные запросы

1. Выводит жанры с числом исполнителей в этом жанре больше одного и сортирует их по убыванию (здесь должно быть большее число, но в таблице недостаточно записей с исполнителями одного жанра).

```
SELECT COUNT(a.id) members, g.genre genre

FROM artists a

    INNER JOIN artist_genre ag ON (a.id = ag.artist_id)

    INNER JOIN genre g ON (ag.genre_id = g.id)

WHERE a.end_of_activity ISNULL

GROUP BY g.id

HAVING COUNT(a.id) > 1 ORDER BY members DESC;
```

Необходимость:

Данный запрос позволяет отследить определенную статистику в жанрах, установив определенную планку в количестве исполнителей. Какой-нибудь журналист при помощи данного запроса сможет оценить самые популярные жанры среди исполнителей, при этом не включая совсем редкие и непопулярные жанры.

2. Выводит средний возраст исполнителей (для живых исполнителей); средний возраст, в котором исполнители начинают свою карьеру; средний возраст смерти исполнителей (для скончавшихся исполнителей); среднюю длительность существования ассоциаций (для ассоциаций, прекративших свою деятельность).

```
SELECT (SELECT floor(avg(date_part('year', localtimestamp) - date_part('year',
a.birth_date))) AS avg_age FROM artists a WHERE a.death_date ISNULL),

floor(avg(a.start_of_activity - date_part('year', a.birth_date))) AS
avg_career_start_age,

(SELECT floor(avg(date_part('year', a.death_date) - date_part('year', a.birth_date)))
FROM artists a WHERE a.death_date NOTNULL) AS avg_death_age,

(SELECT floor(avg(ass.end_of_activity - ass.start_of_activity)) FROM associations
AS ass WHERE ass.end_of_activity NOTNULL) AS avg_association_existing

FROM artists AS a

 INNER JOIN artist_associations AS artas ON (a.id = artas.artist_id)

 INNER JOIN associations AS ass ON (artas.associations_id = ass.id);
```

Необходимость:

Чисто статистический запрос, интересен тем, что позволяет определить некоторые показатели, из которых можно сделать интересные выводы, например, что хип-хоп исполнители умирают в раннем возрасте.

3. Выводит информацию об артистах старше 30 лет, которые являются звукозаписывающими продюсерами, и которые состоят в группах, существующих более 10 лет и продолжающих свою деятельность.

```
SELECT a.artist, a.birth_date, b.* from (SELECT occupation, birth_date, artist, id
FROM artists WHERE floor(date_part('year', localtimestamp) - date_part('year',
artists.birth_date)) > 30 AND occupation = 'record producer') AS a

 INNER JOIN artist_associations AS aras ON a.id = aras.artist_id

 INNER JOIN (SELECT id AS ass_id, association, start_of_activity FROM
associations WHERE floor(date_part('year', localtimestamp) -
associations.start_of_activity) > 10 AND associations.end_of_activity ISNULL) AS
b ON b.ass_id = aras.associations_id;
```

Необходимость:

Результатом запроса будут самые видные и значимые деятели хип-хоп культуры, которые продолжают свою деятельность в качестве музыкальных продюсеров. Таких исполнителей можно назвать “отцами” современного хип-хопа, именно они создали и продолжают совершенствовать этот музыкальный стиль.