Санкт-Петербургский государственный университет
факультет прикладной матетматики – процессов управления

Ивкин Кирилл Андреевич

Курсовая работа

Гитарный мир
(guitar world)

Направление 01.03.02
«Прикладная математика и информатика»

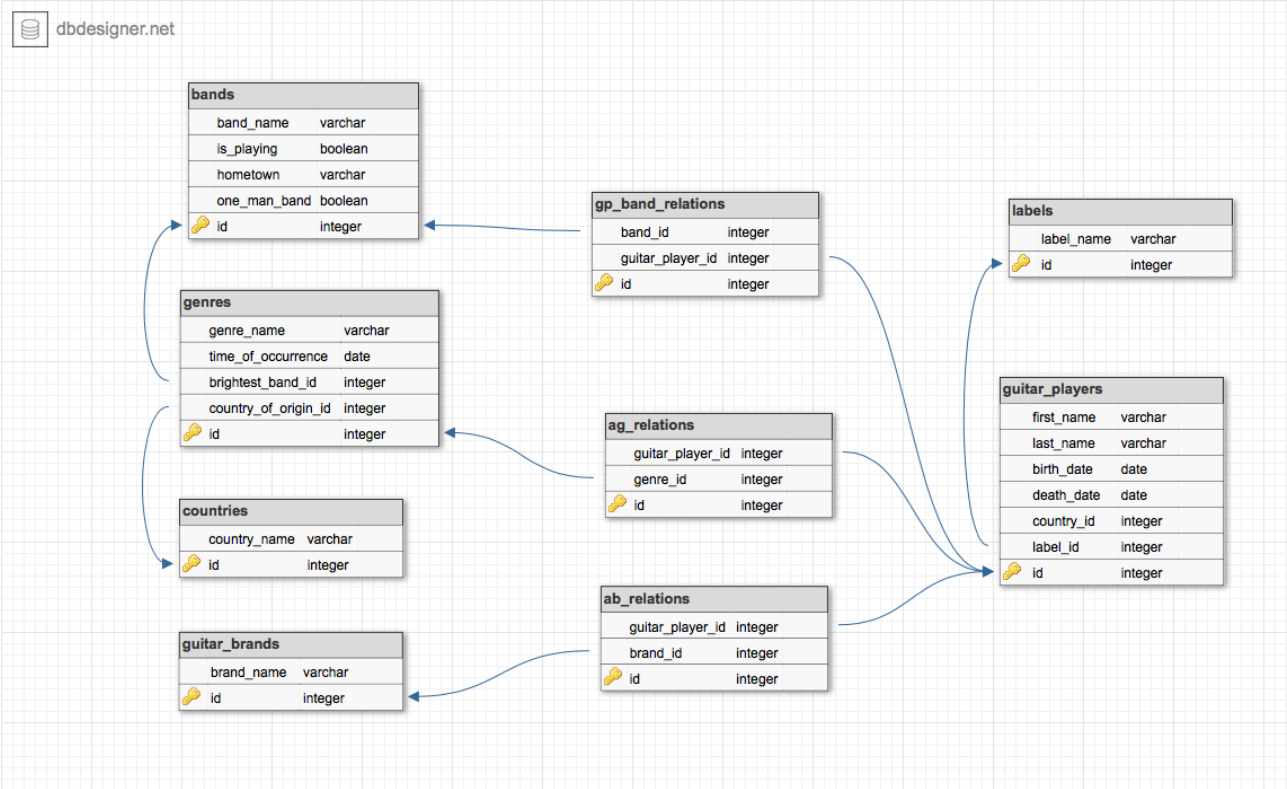
Преподаватель: Филиппов Р.О.

Санкт-Петербург
2017

Оглавление

Глава 1: Схема

Здесь представлена структура базы «Гитарный мир»



Ссылка на GitHub репозиторий: https://github.com/sitar777/guitar_world_db

Глава 2: Описание базы данных

Данная БД представляет собой список гитаристов, которые были признаны лучшими во все времена. Здесь будут представлен список гитаристов, играющих в разных жанрах и разные времена. За подробной информацией об этих людях можно обратиться в Википедию, здесь же представлены лишь некоторые особенности этих гитаристов.

guitar_players – список гитаристов

- first_name (character varying(256) NOT NULL) – имя гитариста
- last_name (character varying(256)) – фамилия гитариста
- birth_date (date NOT NULL) – день рождения гитариста
- death_date (date) – день смерти гитариста
- country_id (integer NOT NULL) – страна, откуда гитарист родом, внешний ключ на поле id таблицы countries
- label_id (integer NOT NULL) – звукозаписывающий лейбл гитариста, внешний ключ на поле id таблицы labels
- id (serial) – является первичным ключом

В данной таблице есть ограничение, что рождение гитариста должно быть раньше, чем его смерть (birth_death_date).

bands – группы в которыхиграют или играли гитаристы

- band_name (character varying(256) NOT NULL) – имя группы
- is_playing (boolean NOT NULL) – играет ли группа сейчас
- hometown (character varying(256)) – родной город группы
- one_man_band (boolean NOT NULL) – является ил группа сольным проектом
- id (serial) – является первичным ключом

genres – жанры в которых играют гитаристы

- genre_name (character varying(256) NOT NULL) – название жанра
- country_of_origin_id (integer NOT NULL) – страна, где зародился жанр, внешний ключ на поле id таблицы countries
- year_of_occurrence (smallint NOT NULL) – примероное время появления жанра
- brightest_band_id (integer NOT NULL) – ярчайший представитель жанра, внешний ключ на поле id таблицы bands
- id (serial) – является первичным ключом

В данной таблице есть ограничение, которое указывает на то, что мы рассматриваем жанры, «родившиеся» не раньше 1850 года (genre_year_of_occurrence_check)

countries – список стран используемых в таблицах

- country_name (character varying(256) NOT NULL) – имя страны
- id (serial) – является первичным ключом

labels – звукозаписывающие лейблы работавшие с гитаристами

- label_name (character varying(256) NOT NULL) – название звукозаписывающего лейбла (студии звукозаписи)
- id (serial) – является первичным ключом

guitar_brands – брэнды гитар используемых гитаристами

- brand_name (character varying(256) NOT NULL) – название брэнда по изготовлению гитар
- id (serial) – является первичным ключом

Перейдем к рассмотрению таблиц, реализущих отошение m:m (многие к многим)

ab_relations – отношения гитарист-бренд гитар (показывает на каких гитарах играет каждый отдельно взятый гитарист)

- guitar_player_id (integer NOT NULL) – внешний ключ на поле id таблицы guitar_players
- brand_id (integer NOT NULL) – внешний ключ на поле id таблицы brands
- id (serial) – является первичным ключом

ag_relations – отношения гитарист-жанр (показывает в каких жанрах играет каждый отдельно взятый гитарист)

- guitar_player_id (integer NOT NULL) – внешний ключ на поле id таблицы guitar_players
- genre_id (integer NOT NULL) – внешний ключ на поле id таблицы genres
- id (serial) – является первичным ключом

gp_band_relations – отношения гитарист-группа (показывает в каких группах играл или играет каждый отдельно взятый гитарист)

- band_id (integer NOT NULL) – внешний ключ на поле id таблицы bands
- guitar_player_id (integer NOT NULL) – внешний ключ на поле id таблицы guitar_players
- id (serial) – является первичным ключом

Глава 3: Простые запросы и оптимизация

1) Выбираем всех гитаристов из США.

```
SELECT g.first_name, g.last_name
FROM guitar_players g
WHERE g.country_id = 1;
```

Для оптимизации запросов был создан индекс country_id_idx (CREATE INDEX country_id_idx ON public.guitar_players (country_id)), так как условием выбора данных является g.country_id = 1. Прироста в производительности сканирование при помощи индекса не дало. Ранее созданные индексы не использовались.

2) Выбираем только живых гитаристов и сортируем их по дате рождения.

```
SELECT g.first_name, g.last_name, g.birth_date
FROM guitar_players g
WHERE death_date IS NULL
ORDER BY g.birth_date;
```

Для оптимизации запросов был создан индекс death_date_idx (CREATE INDEX death_date_idx ON guitar_players (death_date NULLS FIRST)), так как условием выбора данных является death_date IS NULL. Прироста в производительности сканирование при помощи индекса не дало. Ранее созданные индексы не использовались.

3) Выбираем играющие сейчас группы (не сольное творчество) и сортируем их по родному городу.

```
SELECT b.band_name, b.hometown
FROM bands b
WHERE b.is_playing = TRUE AND b.one_man_band = FALSE
ORDER BY b.hometown;
```

Для оптимизации запросов были созданы индексы is_playing_idx (CREATE INDEX is_playing_idx ON bands (is_playing) WHERE is_playing) и one_man_band_idx (CREATE INDEX one_man_band_idx ON bands (one_man_band) WHERE NOT one_man_band), так как условием выбора данных являются b.is_playing = TRUE и b.one_man_band = FALSE. Появился прирост в скорости выполнения запроса, но повысилась стоимость запроса. Использовался индекс hometown_indx, созданный ранее.

4) Выбираем первых 15 городов, которые являются родными городами группы.

```
SELECT DISTINCT b.hometown
FROM bands b
WHERE b.hometown IS NOT NULL
ORDER BY b.hometown
LIMIT 15;
```

Для оптимизации запроса был создан индекс hometown_indx (CREATE INDEX hometown_indx ON public.bands (hometown NULLS LAST)), так как условием выбора данных является b.hometown IS NOT NULL. Прироста в производительности сканирование при помощи индекса не дало. Ранее созданные индексы не использовались.

Глава 4: Средние запросы и оптимизация

1) Показывает на какие группы ссылается поле brightest_band_id, заменяет id на имя группы. Удобней смотреть какая группа является ярким представителем жанра.

```
SELECT b.band_name, g.genre_name
FROM bands AS b
RIGHT JOIN genres AS g ON b.id = g.brightest_band_id
ORDER BY b.band_name;
```

Для оптимизации запросов был создан индекс brightest_band_id_idx (CREATE INDEX brightest_band_id_idx ON genres (brightest_band_id)), так как условием выбора данных является b.id = g.brightest_band_id. Прироста в производительности сканирование при помощи индекса не дало. Использовался индекс bands_pk, созданный ранее.

2) Показывает страну, откуда гитарист родом, для первых 10 музыкантов вместо id страны, сортирует по алфавиту.

```
SELECT g.first_name, g.last_name, c.country_name
FROM countries c
LEFT JOIN guitar_players g ON c.id = g.country_id
ORDER BY g.first_name, g.last_name
LIMIT 10;
```

Для оптимизации запросов был создан индекс country_id_idx (CREATE INDEX country_id_idx ON guitar_players (country_id)), так как условием выбора данных является c.id = g.country_id. Прироста в производительности сканирование при помощи индекса не дало. Использовался индекс countries_pk созданный ранее.

3) Показывает звукозаписывающий лейбл для всех гитаристов из США вместо id лейбла

```
SELECT g.first_name, g.last_name, l.label_name
FROM labels l
INNER JOIN guitar_players g ON l.id = g.label_id
WHERE g.country_id = 1
ORDER BY g.first_name, g.last_name;
```

Для оптимизации запросов был создан индекс label_id_idx ((CREATE INDEX label_id_idx ON guitar_players (label_id)), так как условием выбора данных является l.id = g.label_id. Прироста в производительности сканирование при помощи индекса не дало. В то же время ускорилась склейка таблиц. Использовались индексы labels_pk и country_id_idx созданные ранее.

Глава 5: Сложные запросы

1) Показывает в каких группах играют или играли гитаристы

```
SELECT gpb.first_name, gpb.last_name, b.band_name
FROM (SELECT g.first_name, g.last_name, gp.band_id
FROM guitar_players g
RIGHT JOIN gp_band_relations gp ON g.id = gp.id) AS gpb
RIGHT JOIN bands b ON b.id = gpb.band_id
WHERE gpb.first_name IS NOT NULL
ORDER BY gpb.first_name, gpb.last_name;
```

2) Считает, сколько разных жанров музыки записала компания Warner Brosers с гитаристами из таблицы guitar_players

```
SELECT COUNT (*)
FROM (SELECT DISTINCT ag.genre_id
FROM (SELECT g.id, g.first_name, g.last_name, l.label_name
FROM guitar_players g
RIGHT JOIN labels l ON g.label_id = l.id
WHERE l.label_name = 'Warner Bros') AS wbgp
RIGHT JOIN ag_relations ag ON wbgp.id = ag.id
WHERE wbgp IS NOT NULL) as wbg ;
```

3) Считает сколько гитаристов записывали свою музыку у каждого из лейблов

```
SELECT l.label_name, COUNT (g.label_id)
FROM guitar_players g
RIGHT JOIN labels l ON g.label_id = l.id
GROUP BY g.label_id, l.label_name;
```