Практическое задание по теме “Управление БД”

1. Проанализировать структуру БД vk, которую мы создали на занятии, и внести предложения по усовершенствованию (если такие идеи есть). Напишите пожалуйста, всё-ли понятно по структуре.

По структуре вопросов нет.

Предложение по совершенствованию структуры:

1. Таблица media\_types носит явно справочнный характер. Она заполняется/редактируется либо администраторами, либо разработчиками крайне редко. В этом смысле не очевидна необходимость присутствия полей:
   * Created\_at – можно легко вычислить примерную дату появления того или иного типа медиа-контента с помощью запроса к таблице медиа по минимальной дате ее поля created\_at, отфильтровав по интересующему media\_type\_id;
   * Updated\_at – это поле вообще ошибочно, т.к. по сути апдейтов не предполагается.

Изменения в исходном скрипте:

Комментарии внутри описания таблицы media\_types

*-- created\_at DATETIME DEFAULT NOW(),*

*-- updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP*

1. Не очевидна необходимость создания промежуточной таблицы photos, практически являющейся представлением таблиц photo\_albums и media. С моей точки зрения возможна оптимизация структуры посредством исключения таблицы photos (или преобразования ее к представлению) и добавления поля media\_id в таблицу photo\_albums, с соответствующим ограничением целостности. Одновременно будет достигнута унификация создания альбомов, например, можно создать альбом видео-, аудио- и смешанного контента.

Изменения в исходном скрипте:

Помещен в комментарий блок создания таблицы photos:

*/\**

*DROP TABLE IF EXISTS `photos`;*

*CREATE TABLE `photos` (*

*id SERIAL PRIMARY KEY,*

*`album\_id` BIGINT unsigned NOT NULL,*

*`media\_id` BIGINT unsigned NOT NULL,*

*FOREIGN KEY (album\_id) REFERENCES photo\_albums(id),*

*FOREIGN KEY (media\_id) REFERENCES media(id)*

*);*

*\*/*

Помимо этого добавлено:

***alter******table*** *vk\_sol.photo\_albums* ***add******column*** *media\_id* ***BIGINT******unsigned******NOT******NULL****;*

***alter******table*** *vk\_sol.photo\_albums* ***add******CONSTRAINT******FOREIGN******KEY*** *(media\_id)* ***REFERENCES*** *vk\_sol.media(id);*

1. В порядке фантазии: в сети VK присутствует только один тип лайков. К сети FB присутствует 6 типов лайков. С точки зрения структуры БД такой функционал нетрудно получить следующим образом:
   * добавить в таблицу likes поле like\_type\_id
   * добавить таблицу like\_types со следующими полями:
     + - ID (SERIAL, primary key)
       - Like\_Types (ENUM ‘like’, ‘heart’, ‘funny’, ‘wow’, ‘cry’, ‘dislike’)
   * Добавить ограничение целостности по полям like.like\_types\_id и like\_types.id

Изменения в исходном скрипте:

**DROP** **TABLE** **IF** **EXISTS** like\_types;

**CREATE** **TABLE** like\_types(

id SERIAL **PRIMARY** **KEY**,

like\_type **ENUM** ('like', 'heart', 'funny', 'wow', 'cry', 'diclike')

);

**alter** **table** vk\_sol.likes **add** **column** like\_type\_id **BIGINT** **unsigned** **NOT** **NULL**;

**alter** **table** vk\_sol.likes **add** **CONSTRAINT** **FOREIGN** **KEY** (like\_type\_id) **REFERENCES** vk\_sol.like\_types(id);

1. Серьезная логическая ошибка, обнаруженная на фазе заполнения тестовыми данными:

user\_id SERIAL **PRIMARY** **KEY**,

не может быть SERIAL так как связано с аналогичным первичным ключом в таблице USERS. Поправлено на:

**BIGINT** **UNSIGNED** **NOT** **NULL** **UNIQUE** **PRIMARY** **KEY**,

1. Добавить необходимую таблицу/таблицы для того, чтобы можно было использовать лайки для медиафайлов, постов и пользователей.

Описание скрипта приводится в файле создания БД. Сам скрипт ниже:

**DROP** **TABLE** **IF** **EXISTS** like\_users;

**CREATE** **TABLE** like\_users(

id SERIAL **PRIMARY** **KEY**,

user\_id **BIGINT** **UNSIGNED** **NOT** **NULL**,

whom\_id **BIGINT** **UNSIGNED** **NOT** **NULL**, -- кому лайк

like\_type\_id **BIGINT** **UNSIGNED** **NOT** **NULL**,

created\_at **DATETIME** **DEFAULT** **NOW**(),

**FOREIGN** **KEY** (user\_id) **REFERENCES** users(id),

**FOREIGN** **KEY** (whom\_id) **REFERENCES** users(id),

**FOREIGN** **KEY** (like\_type\_id) **REFERENCES** like\_types(id)

);

**DROP** **TABLE** **IF** **EXISTS** like\_messages;

**CREATE** **TABLE** like\_messages(

id SERIAL **PRIMARY** **KEY**,

user\_id **BIGINT** **UNSIGNED** **NOT** **NULL**,

message\_id **BIGINT** **UNSIGNED** **NOT** **NULL**,

like\_type\_id **BIGINT** **UNSIGNED** **NOT** **NULL**,

created\_at **DATETIME** **DEFAULT** **NOW**(),

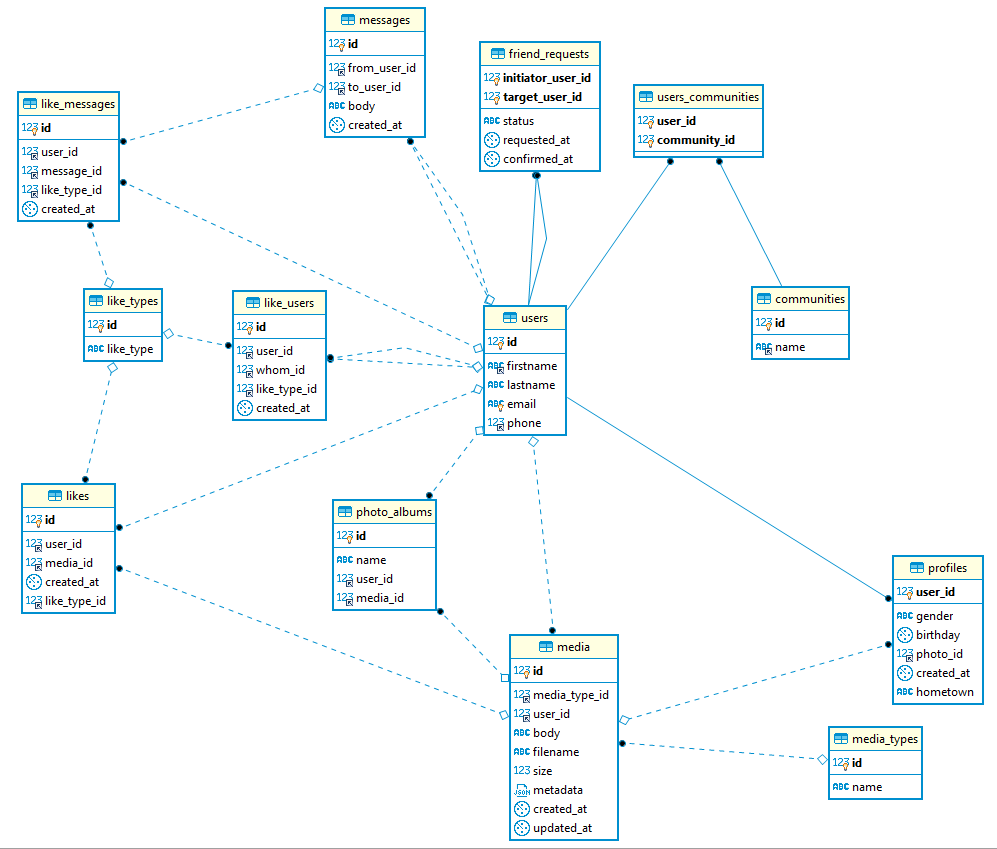
**FOREIGN** **KEY** (user\_id) **REFERENCES** users(id),

**FOREIGN** **KEY** (message\_id) **REFERENCES** messages(id),

**FOREIGN** **KEY** (like\_type\_id) **REFERENCES** like\_types(id)

);

Итоговая структура БД:



1. Используя сервис [http://filldb.info](http://filldb.info/) или другой по вашему желанию, сгенерировать тестовые данные для всех таблиц, учитывая логику связей. Для всех таблиц, где это имеет смысл, создать не менее 100 строк. Создать локально БД vk и загрузить в неё тестовые данные.

Данные загружены. Для загрузки схемы данных в генератор использован файл VK\_DB\_for\_load.sql

Для загрузки данных в локальную базу использован сгенерированный файл

fulldb10-11-2019 12-51.sql

Выложено на Githab.