Курс: Основы реляционных баз данных. MySQL
Урок 4. CRUD-операции
Выполнил: Кузнецов Сергей (Факультет Geek University Python-разработки)

Домашнее задание:

1. Повторить все действия по доработке структуры БД vk и данных.
2. Подобрать сервис который будет служить основой для вашей курсовой работы.
3. (по желанию) Предложить свою реализацию лайков и постов.

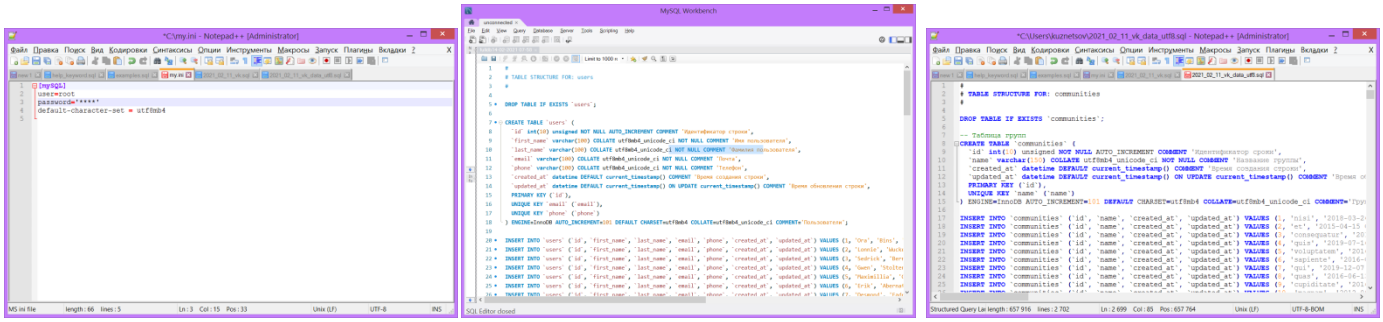
---

1. Повторить все действия по доработке структуры БД vk и данных.

Учитывая работу в классе и собственные изменения структуры БД, удалим все наработки и прогрузим скрипт с данными, полученными в ходе выполнения прошлого ДЗ. Как и рекомендует преподаватель, прогрузим скрипт через консоль (с большими объемами данных он отрабатывает существенно быстрее DBeaver. Заходим в mySQL (в конфигурационном файле прописана кодировка «*default-character-set = utf8mb4*»). Небольшое отступление. Сегодня существуют символы, которые занимают более 3 байт и не помещаются в «*utf8*». Несмотря на то, что начиная с версии MySQL 5.5.3 можно не беспокоится о потере данных в столбцах, все-же рекомендуют использовать именно «*utf8mb4*», а не «*utf8*».

Файл с дампом, полученный входе выполнения ДЗ прошлого урока, с http://filldb.info вместо комментариев в описании таблиц и столбцов имеет кракозябры. Сейчас, после того как с кодировкой «подружился», это ресурс генерит описания с читабельными комментариями (проверил на простых таблицах).

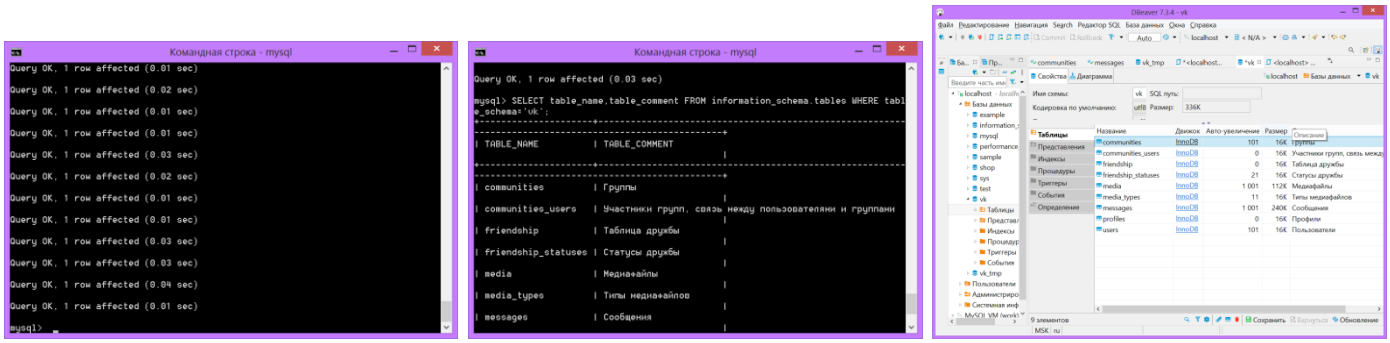
Из природной лени, не стал генерить все по-новой, а просто в файл с дампом подставил комментарии.



Делаем текущей БД vk «USE vk» и запускаем на исполнение скрипт «*SOURCE 2021_02_11_vk_data_utf8.sql*».

Убедимся, что кодировка не съехала «*SELECT table_name,table_comment FROM information_schema.tables WHERE table_schema='vk';*»

И база наполнена данными.



Теперь, можем приступить к выполнению задания текущего урока. По итогу принятого преподавателем решения, в текущей базе мы реализуем только:

- Вносим изменения в таблицу профилей «*ALTER TABLE profiles MODIFY COLUMN gender ENUM('M', 'F');*»
  Стоит заметить, что впрямую такая операция на данных не прошла «*SQL Error [1265] [01000]: Data truncated for column 'gender' at row 3*». Это вполне логично, что перед ее выполнением, мы должны

привесттии все данные к соответствию этому ограничению. Для этого, не будем создавать временную таблицу (слишком длинная запись, а ограничимся выборкой) «*UPDATE profiles SET gender = (SELECT * FROM (SELECT 'F' AS gender UNION SELECT 'M' AS gender) gender_list ORDER BY rand() LIMIT 1);*»

- Вносим изменение в таблицу групп «*ALTER TABLE communities ADD COLUMN owner_id INT UNSIGNED NOT NULL AFTER id;*»
- Убираем избыточный столбец «*ALTER TABLE friendship DROP COLUMN requested_at;*»

Все комментарии и последовательность действий по приведению БД к более натуральному (достоверному) виду приведены в скрипте «*2021_02_14_vk_convert_utf8.sql*».

Обращаю внимание на то, что обновление дат в «profiles» произвел обычным копированием из таблицы «users»: «*UPDATE profiles p SET p.created_at = (SELECT u.created_at FROM users u WHERE u.id = p.user_id), p.updated_at = (SELECT u.updated_at FROM users u WHERE u.id = p.user_id);*» Честно говоря, в нашем случае «profiles» является расширением «users» и изменению (созданию) подвергается одновременно. Так что необходимость идентичных полей в обеих таблицах сомнительна. Более оптимальным выглядел бы запрос «*UPDATE profiles p SET (p.created_at, p.updated_at) = (SELECT u.created_at, u.updated_at FROM users u WHERE u.id = p.user_id);*»… но он выдает ошибку – разберусь позже.

Что касается сопряжения типа медиа-контента с расширением файла в таблице «media». Временную таблицу создадим также, как и предложено в классе. «*CREATE TEMPORARY TABLE extensions (ext varchar(10));*». Заполним вариантами «*INSERT INTO extensions VALUES ('jpeg'), ('avi'), ('mpeg'), ('png'), ('wav');*». А вот изменение таблицы «media» произведем более сложным способом:

```
UPDATE media SET filename =
        concat('https://dropbox.com/vk/', filename, '.',
        @ext:=(SELECT ext FROM extensions ORDER by rand() LIMIT 1)),
        media_type_id =
        CASE @ext
        WHEN 'avi' THEN (SELECT ID FROM media_types WHERE name='video')
        WHEN 'jpeg' THEN (SELECT ID FROM media_types WHERE name='photo')
        WHEN 'mpeg' THEN (SELECT ID FROM media_types WHERE name='photo')
        WHEN 'png' THEN (SELECT ID FROM media_types WHERE name='photo')
        WHEN 'wav' THEN (SELECT ID FROM media_types WHERE name='audio')
        ELSE 1
        END;
```

К сожалению, ограничение по повторному обращению к временной таблице, в рамках одного запроса, несколько усложнило задачу и не позволило использовать временную таблицу для хранения/изъятия «*media_type_id*»… Кстати, из осторожности «испортить» данные в поле «*filename*», я сначала проверил через SELECT, потом создал временно доп. поле в таблице и присвоил ему и , только после этого, прошелся запросом по реальным данным. Насколько помню, раньше были какие-то механизмы, позволяющие произвести откат транзакции по массовому изменению БД…(вопрос «на после» или к преподавателю).

2. *Подобрать сервис который будет служить основой для вашей курсовой работы.*

Основной целью обучения в GEEKBRAINS является последующее трудоустройство по ИТ специальности. Несмотря на наличие неплохого багажа знаний и опыта работы в сегменте ИТ, многое забыто и/или устарело. Все это делает непростым выбор темы Курсовой работы. Отчасти, согласен с преподавателем, что «*Работодатели будут оценивать не для чего вы сделали проект, а как сделали, основа второстепенна*.»

Но, помня себя и мои принципы набора штата в ИТ,…. порогом к рассмотрению компетенций, была предметная область, в которой претендент себя уверенно позиционирует.

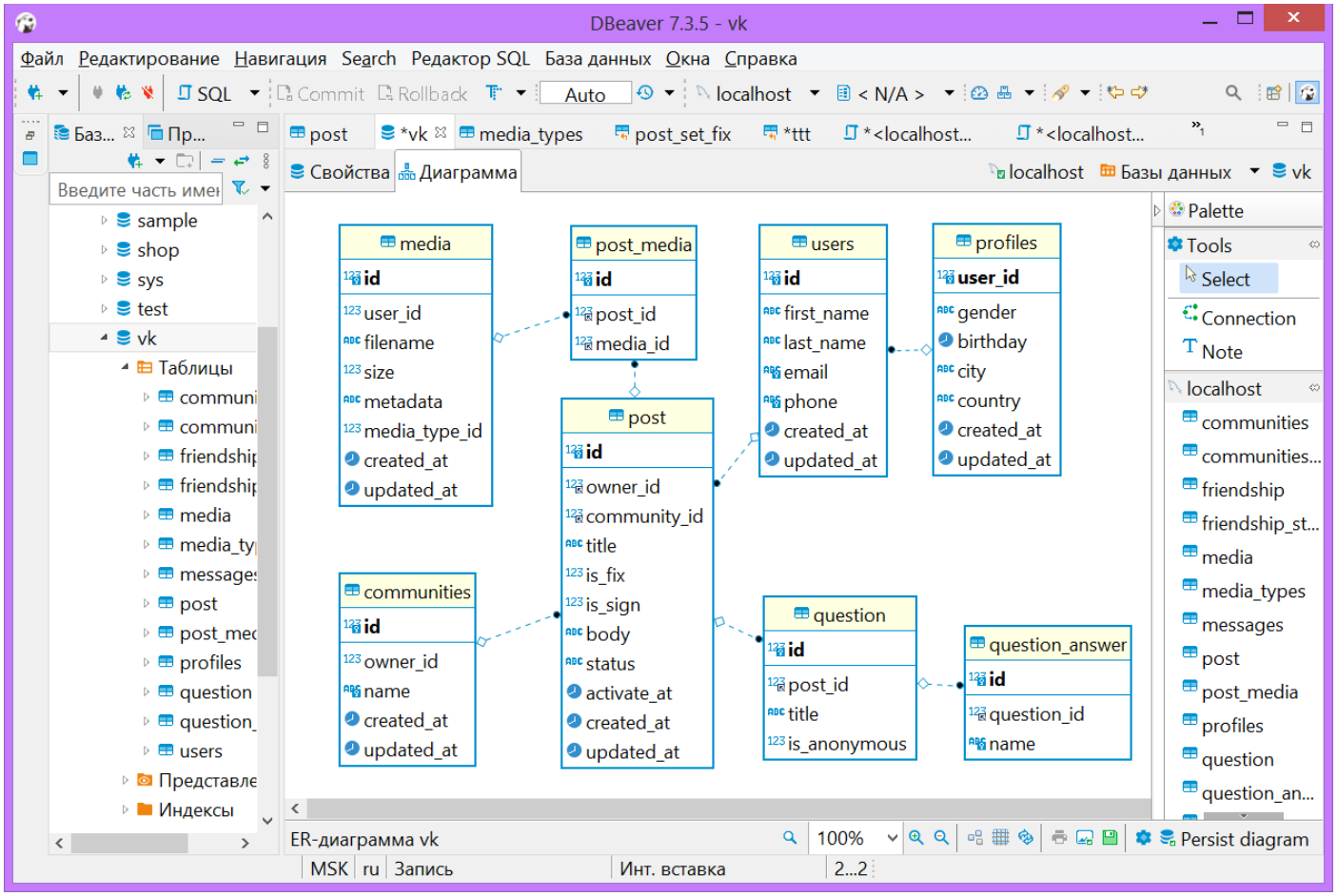
Проанализировав рейтинг ведущих веб-сайтов мира, используя данные SimilarWeb[1], видим, первую строчку рейтинга занимает сайт поисковика google, который имеет более 60 миллиардов визитов ежемесячно. Следом идёт видеохостинг YouTube с 24 миллиардами просмотров в месяц. Ниже размещаются 2 популярные во всём мире соцсети facebook и twitter. Выбрал facebook.

[1] Это рейтинг крупнейших сайтов. В него не включены онлайн-игры и некоторые приложения, требующие непосредственного интернет-соединения, такие как WeChat, Snapchat, Viber, Telegram и другие.

3. *Предложить свою реализацию постов.*

Сразу оговорюсь, что над архитектурой поработал без фанатизма, учитывая то, что это учебное задания на предмет закрепления навыков. Посмотрев пару статей про Посты в VK, уяснил, что они привязываются к группам, имеют заголовки, могут содержать текст, фото, видео, картинки, …(медиаконтент), опросы, карты и т.д. В группе может быть один закрепленный Пост, который высвечивается вверху, как поплавок. Также, посты могут быть отложенными, т.е. запускаются по таймеру…

Итоговая схема (процесс генерации в файле «*2021_02_14_like_post.sql*»):



Естественно, в описание таблиц заложено поддержание ссылочной целостности при помощи ForeignKey.

Самым затратным с т.з. времени оказалось построение триггеров. То, что они при создании должны запускаться целиком (используем переопределении знака разделителя «*DELIMITER $$*» и возвращение «*DELIMITER ;*») разобрался легко. А вот из-за «клина» по правилам описания «*IF…THEN … END IF;*» потерял достаточно времени. Естественно, попутно разобрался с временными переменными «*SET @ttt = (SELECT…)*» и ограничениями по рекурсивному вызову триггера (попытка изменить таблицу «владельца триггера» внутри собственного триггера). Конечно, учитывая идентичность действий, нужно было создать хранимую процедуру и вызывать ее из обоих триггеров (BEFORE INSERT и DEFORE UPDATE), но это в следующий чуть ниже.

4. Предложить свою реализацию лайков.

Что касается лайков, то тут решил сделать универсальное решение, которое не зависит от архитектуры БД и позволяет привязывать Лайки к любой сущности (таблице)…

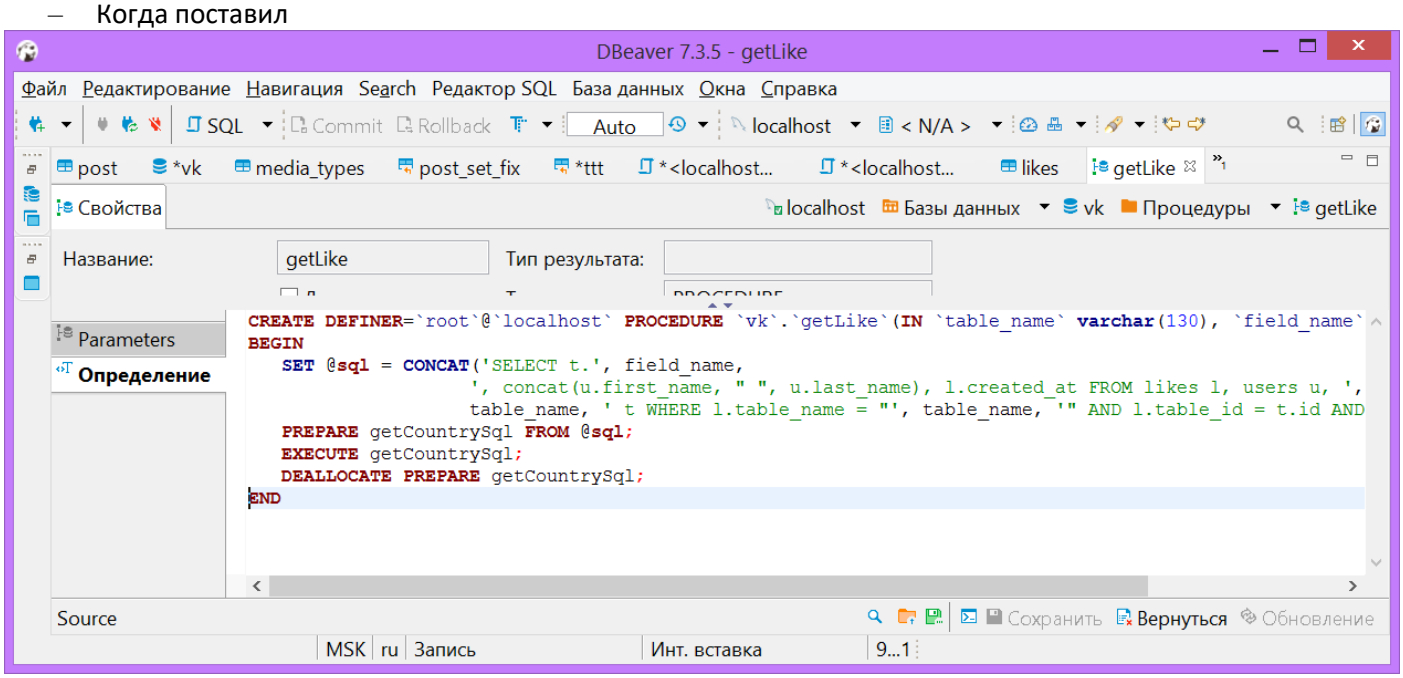
Создаем хранимую процедуру получения всех лайков по выбранной сущности на основе построения динамического SQL-запроса.

Входные параметры:

- имя таблицы, кому поставлены Лайки (users, post, messages, friendship, communities…, в общем – любая)
- имя столбца в этой таблице, чтобы было можно идентифицировать (кому поставлен Лайк)

На выходе итоги запроса с указанием:

- Кому поставили Лайк
- Кто поставил
- Когда поставил



-- Примеры вызова процедуры для различных сущностей (таблиц)

CALL getLike('users', 'first_name');

CALL getLike('post', 'title');