Курс: Основы реляционных баз данных. MySQL

Урок 2. Введение в проектирование БД

Выполнил: Кузнецов Сергей (Факультет Geek University Python-разработки)

Домашнее задание:

1. Создать структуру БД Вконтакте по скриптам, приложенным в файле примеров examples.sql
2. Используя сервис 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. Загрузить тестовые данные в свою локальную базу данных ВК. Приложить к отчёту полученный дамп с данными.
3. (по желанию) 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.

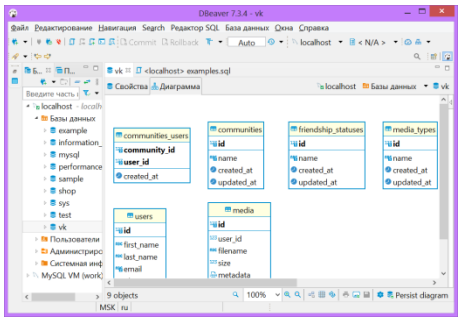
---

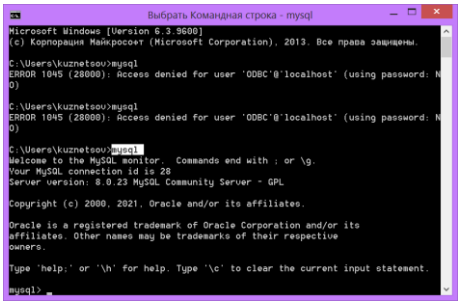
1. *Создать структуру БД Вконтакте по скриптам, приложенным в файле примеров examples.sql.*

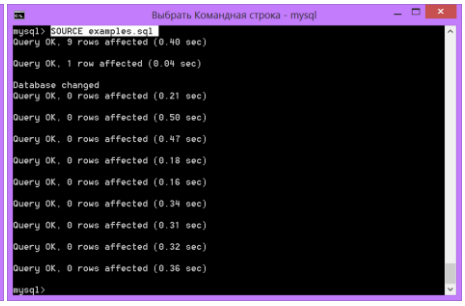
Запускаем DBeaver, подключаем локальное соединение, подгружаем скрипт… Т.к. на уроке базу vk уже частично создал, перед командой создания БД, добавляем в скрипт команду «*DROP DATABASE IF EXISTS vk;*» и выполняем скрипт «Alt+X». Все получилось.

Помня о том, что задача курса «…научиться работать в консольном режиме…», Произведем все то-же самое, но «по честному». Запускаем терминал. Кстати, пришлось все-же выполнить задание с конфигурационным файлом. Переименовал файл «.my.cnf» в «my.ini» и поместил в корень диска «C:\» - все работает!

Для разворачивания дампа БД, введем команду «*SOURCE examples.sql*». Естественно, перед этим мы скопировали файл скрипта в домашний каталог.





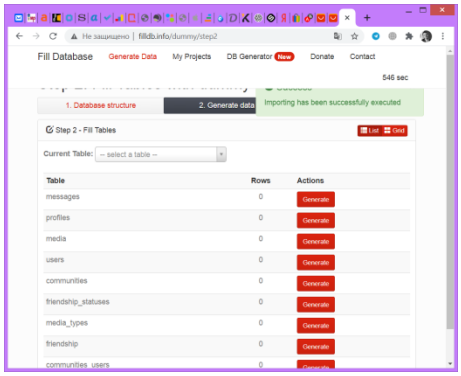
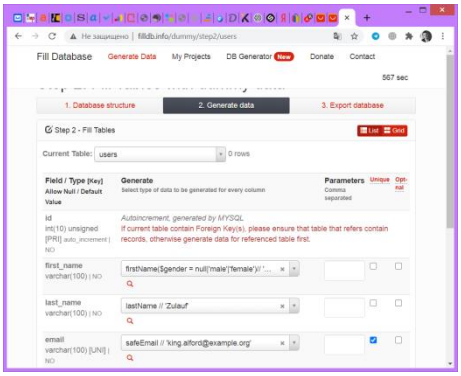
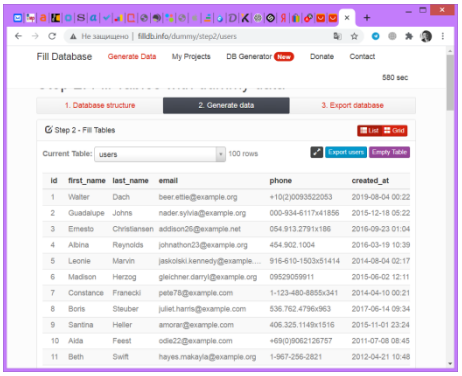


2. *Используя сервис http://filldb.info или другой по вашему желанию, сгенерировать тестовые данные для всех таблиц, учитывая логику связей. Для всех таблиц, где это имеет смысл, создать не менее 100 строк. Загрузить тестовые данные в свою локальную базу данных ВК. Приложить к отчёту полученный дамп с данными.*

Заходим на http://filldb.info, в предложенном диалоге вставляем скрипт создания БД и нажимаем «Submit shema».

В первую очередь сформируем правила генерации данных для независимых таблиц, имеющих только primary_key. Это users, friendship_statuses, communities и media_types.

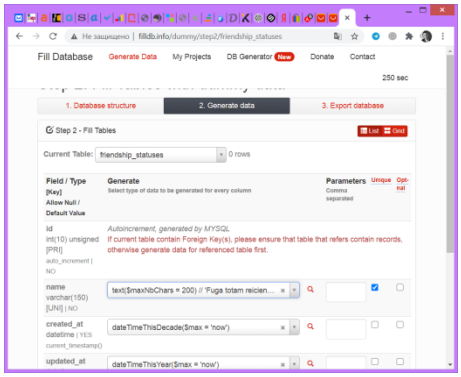
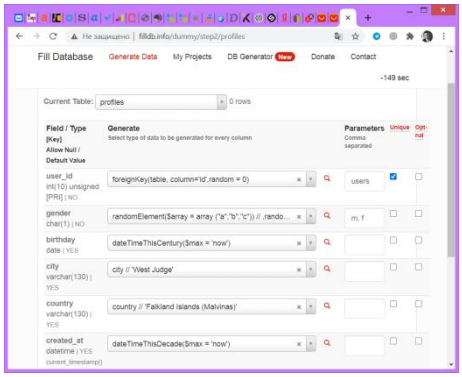
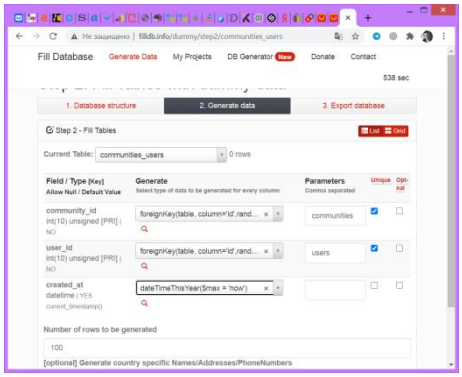
К примеру, в users мы поменяли дату создания на эту декаду и дату изменения на этот год…

В таблице friendship_statuses на поле name – text($maxNbChars…всего 20 статусов. В таблицу media_types внесем 10 типов.

Перейдем к таблицам с Foreign_Key. Начнем с profiles. Здесь система с сайта верно определила ForeignKey из таблицы users. Проставим поле gender на RandomElement($array… с параметрами «m, f». Поле country также, как и в классной работе не подвязалось автоматом – исправим.

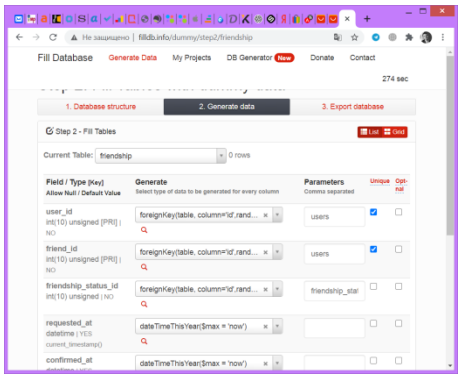
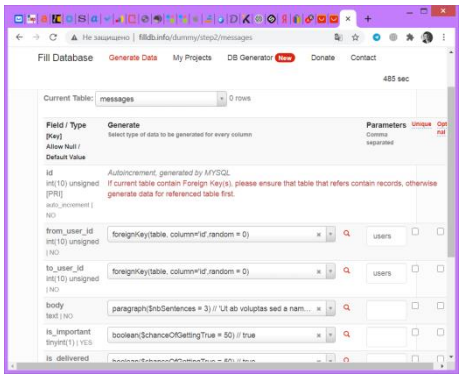
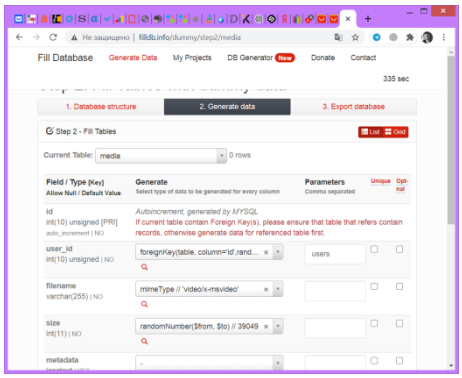
Также поступим и с остальными таблицами communities_users (два ссылочных ключа и дата) – 400 записей.

В таблице friendship (проставим родительские таблицы в ForeignKey и даты) – 400 записей.

Таблица messages (проставим родительские таблицы в ForeignKey и дату) – 1000 записей.

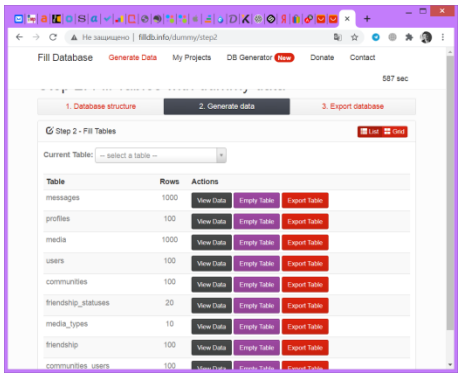
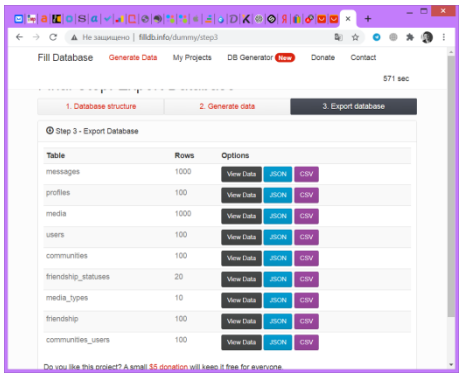
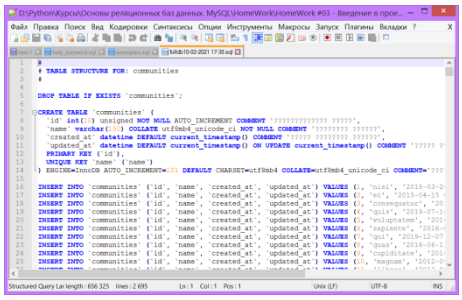
И, последняя таблица media (…) – 1000шт

Данные для всех таблиц получены.

Переходим на вкладку «Export Database» и нажимаем кнопку экспорта.

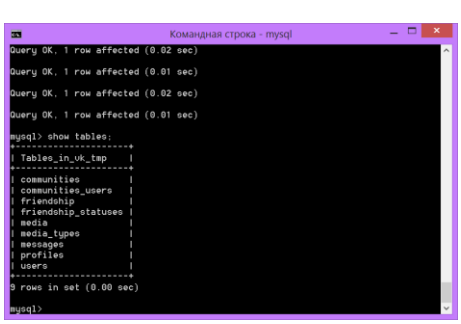
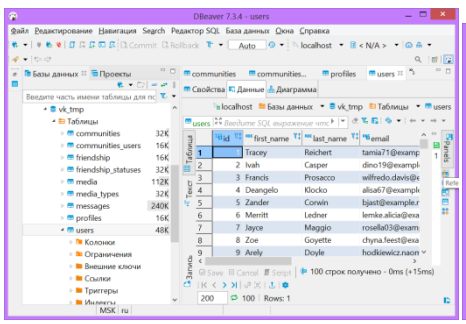
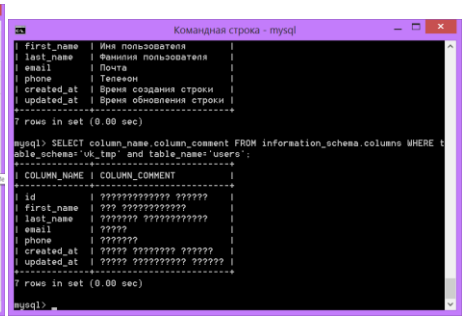
Получен дамп с данными.

Загрузим дамп в базу данных «*SOURCE fulldb10-02-2021 17-38.sql*» (зашел в mysql, создал новую БД «*CREATE DATABASE vk_tmp;*», сделал текущей «*USE vk_tmp*» и запустил скрипт). Проверим «*SHOW TABLES;*» - таблицы созданы.

Убедимся через DBeaver, что БД создана, заполнена Непустыми таблицами.

Единственная проблема (разбор оставил на «потом»), это путаница с кодировкой. Дело в том, что с http://filldb.info получен скрипт с кодировкой «CHARSET=utf8mb4». При загрузке в БД кодировка съехала «*SELECT column_name,column_comment FROM information_schema.columns WHERE table_schema='vk_tmp' and table_name='users';*» Можно и комментарии к таблицам посмотреть – картина аналогична «*SELECT table_name,table_comment FROM information_schema.tables WHERE table_schema='vk_tmp';*»

3. *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.*

Создадим правила целостности, поддерживаемые на уровне БД.

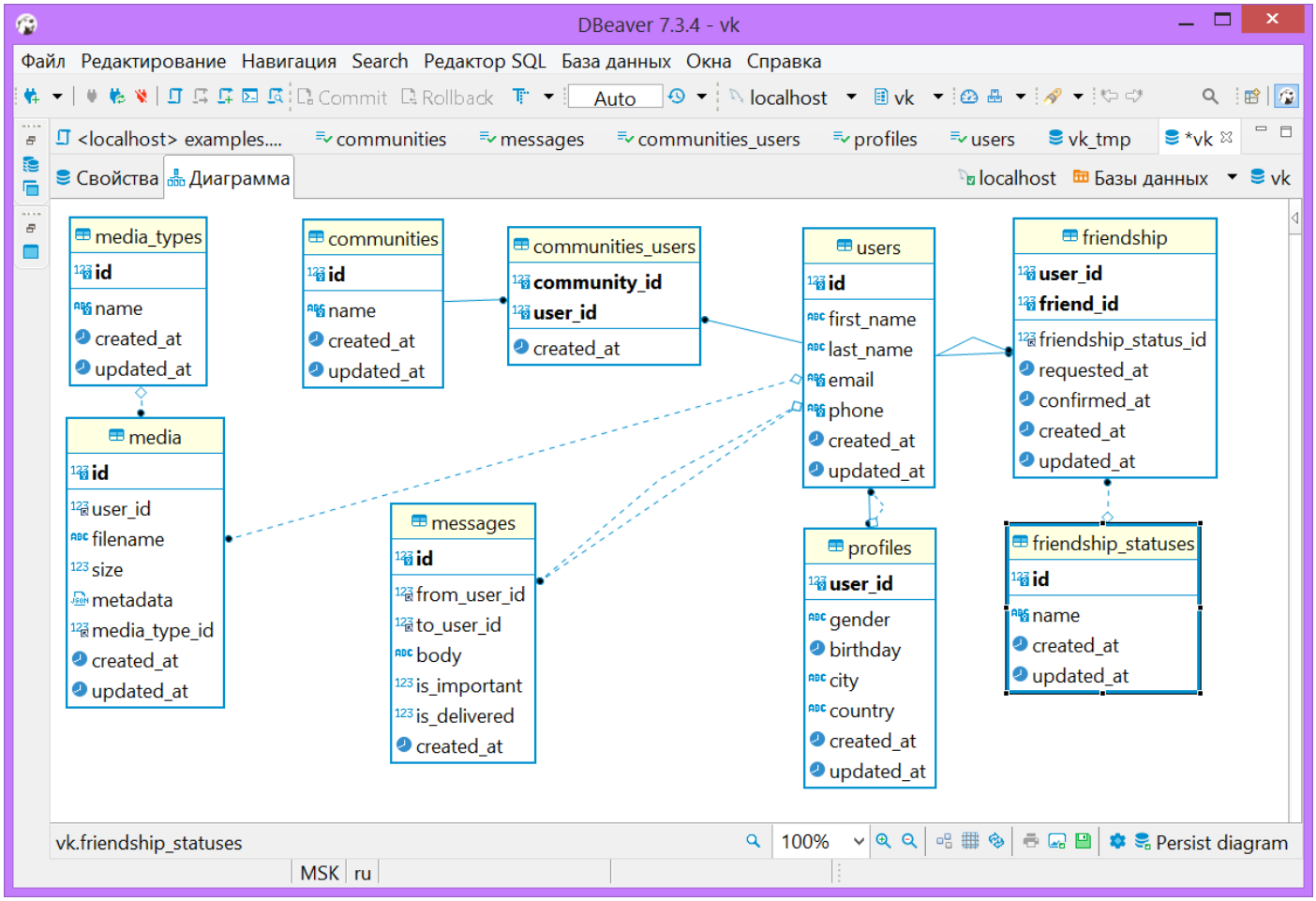
1) Для начала, определимся со связями таблиц. Будем придерживаться правила, что при наличии записей в подчиненной таблице, ссылающихся на родительскую запись, система должна блокировать ее удаление или производить каскадное удаление в подчиненной таблице.
   - «profiles» - зададим внешние ключи и укажем, как поступать с записями. При удалении родительской записи – каскадное удаление и ограничение в изменении ключа записей родительской таблицы:
     FOREIGN KEY (user_id) REFERENCES users (id) ON UPDATE RESTRICT ON DELETE CASCADE

     В принципе, можно было решить эту задачу созданием триггера, запускающегося при каждом удалении записи в таблице «users»:
     ```
     CREATE TRIGGER auto_del BEFORE DELETE ON users
     FOR EACH ROW BEGIN
         DELETE FROM profiles WHERE user_id = OLD.id;
     END;
     ```
     Дело в том, что современные подходы к написанию БД требуют сохранения полной истории и к.л. удаление запрещено. Как правило, статусы записей переводятся в состояние «архив» или они просто переносятся в др. БД или таблицы для хранения устаревших и/или удаленных данных. Естественно, все логгируется (кто, когда и что изменил/удалил,…) – все делается для возможности последующего «разбора полетов». Поэтому, часто приходится прибегать к помощи триггеров, как более гибкому инструменту.
   - «messages» - просто укажем связь с родительской таблицей «users»:
     FOREIGN KEY (from_user_id) REFERENCES users (id),
     FOREIGN KEY (to_user_id) REFERENCES users (id)

     По умолчанию, mySQL поддерживаете ограничение в удалении записей родительских таблиц, имеющих «детей», т.е. можно не указывать в явном виде «*ON UPDATE RESTRICT ON DELETE RESTRICT*».
   - «friendship» - то-же самое
     FOREIGN KEY (user_id) REFERENCES users (id),
     FOREIGN KEY (friend_id) REFERENCES users (id),
     FOREIGN KEY (friendship_status_id) REFERENCES friendship_statuses (id)
   - «communities_users»
     FOREIGN KEY (community_id) REFERENCES communities (id),
     FOREIGN KEY (user_id) REFERENCES users (id)
   - «media»
     FOREIGN KEY (user_id) REFERENCES users (id),
     FOREIGN KEY (media_type_id) REFERENCES media_types (id)
2) Предусмотрим проверку на допустимость вводимых значений.
   - «profiles»
     gender CHAR(1) NOT NULL CHECK (gender in ('m', 'M', 'f', 'F', 'м', 'М', 'ж', 'Ж')) COMMENT "Пол",

   Можно было и просто поменять тип поля (считаю это более логичным вариантом):
     gender ENUM('m', 'M', 'f', 'F', 'м', 'М', 'ж', 'Ж')) COMMENT "Пол",

   В общем нет предела совершенству… нужен вектор, что именно мы хотим получить от БД…
   Измененный скрипт в файле «*2021_02_11_vk.sql*» При проверке, выяснилось, что описание некоторых таблиц (media_types, friendship_statuses) выше описание «родителей» - выдается ошибка…переставил – все Ок.


DBeaver 7.3.4 - vk
Файл Редактирование Навигация Search Редактор SQL База данных Окна Справка
Commit Rollback Auto localhost vk
<localhost> examples.... communities messages communities_users profiles users vk_tmp *vk
Свойства Диаграмма localhost Базы данных vk
media_types
id
name
created_at
updated_at
communities
id
name
created_at
updated_at
communities_users
community_id
user_id
created_at
users
id
first_name
last_name
email
phone
created_at
updated_at
friendship
user_id
friend_id
friendship_status_id
requested_at
confirmed_at
created_at
updated_at
media
id
user_id
filename
size
metadata
media_type_id
created_at
updated_at
messages
id
from_user_id
to_user_id
body
is_important
is_delivered
created_at
profiles
user_id
gender
birthday
city
country
created_at
updated_at
friendship_statuses
id
name
created_at
updated_at
vk.friendship_statuses
100%
Persist diagram
MSK ru