

Вот примеры простых запросов, которые демонстрируют некоторые возможности SQL (MariaDB).

| | |
|---|--|
| SELECT * FROM Article | Вернуть все строки из таблицы Article в случайном порядке |
| SELECT * FROM Article ORDER BY id SELECT * FROM Article ORDER BY creationTime DESC, id | Примеры указания порядка |
| SELECT id, userId FROM Article | Вернуть только столбцы id, userId для всех строк Article |
| SELECT DISTINCT(userId) FROM Article | Вернуть все различные userId из Article - то есть тех, кто хоть что-то написал |
| SELECT * FROM User WHERE id IN (SELECT userId FROM Article) AND creationTime>NOW() - INTERVAL 7 DAY | Вернуть все строки из User, которые соответствуют тем, кто хоть что-то написал и был зарегистрирован не позже недели назад |
| SELECT MAX(creationTime) FROM User | Вернуть время последней регистрации пользователя |
| SELECT * FROM User WHERE openId IS NOT NULL ORDER BY creationTime DESC LIMIT 10 | Вернуть последних 10 зарегистрированных пользователей среди тех, кто указал openId |
| UPDATE Article SET userId=10 WHERE id=7 | Сменить автора у статьи |
| DELETE FROM Article WHERE userId!=1 AND creationTime>NOW() - INTERVAL 7 DAY | Удалить все статьи не первого пользователя, которые одновременно за последнюю неделю |
| INSERT INTO `Article` (`userId`, `text`, `creationTime`) VALUES (4, 'VK Cup 2019', NOW()); | Вставить в таблицу Article |

Задания

0. Скачайте проект с <https://assets.codeforces.com/files/6e78d88da1176e86/5d/wm1.7z>. Перейдите по <http://wp.codeforces.com/phpMyAdmin/> в свою базу данных и накликайте там таблицу User с полями:

- * id (BIGINT до 18 знаков, autoincrement, primary key, not null)
- * login (VARCHAR до 255 знаков, ключ уникальности unique_User_login, not null)
- * passwordSha (VARCHAR до 255 знаков, not null)
- * creationTime (DATETIME, индекс index_User_creationTime, not null)

Запустите проект (поправьте profile.properties), убедитесь, что всё работает - регистрация+вход+выход.

1. Добавить поддержку email (уникальный) при регистрации.

2. Сделать, что входить в систему можно по логину или email (а не только по логину, что хочешь, то и вводишь).
3. Поддержать новую сущность Event - события от пользователя с полями id, userId, type, creationTime. Поле userId надо сделать внешним ключом на User: ALTER TABLE `Event` ADD CONSTRAINT `fk_Event_userId` FOREIGN KEY (`userId`) REFERENCES `User` (`id`). Поле type должно быть enum с пока двумя значениями ENTER, LOGOUT. Вставлять записи в таблицу Event на каждый удачный вход/выход.
4. Сделать якобы подтверждение email для нового аккаунта. Добавить в User новое поле confirmed, по-умолчанию оно false. С таким значением логиниться нельзя. Сделать таблицу EmailConfirmation с полями id, userId, secret (строковый случайный секрет), creationTime. При регистрации добавлять туда запись (и как будто отправлять письмо с просьбой зайти на страницу /confirm?secret=blablabla). Сделать ConfirmPage на которую если зайти с параметром secret и если есть соответствующий EmailConfirmation, то пользователь подтверждается.
5. Сделать сущность Talk (id, sourceUserId, targetUserId, text, creationTime) - сообщение от одного пользователя другому. Сделать страницу /talks (только для авторизованных). Там простая форма с 2 полями "Send Message" и список всех сообщений, где заданный пользователь автор или адресат в порядке от более поздних к более новым.
6. Обратите внимание, что сейчас в RepositoryImpl очень много похожего кода. Проведите рефакторинг (сами придумайте какой), чтобы уменьшить размер кода, переиспользовав его. Возможно, вам понадобится создавать дополнительные удобные методы в DatabaseUtils

