

# Базы данных

## Проект

### Состав команды и описание проекта

Наша команда разработала проект “Музыкальный диктант”, который предназначен для проведения “Всеобщего музыкального диктанта”. Наше веб-приложение является платформой для проведения этого мероприятия в этом и в следующих годах.

Наша команда состоит из четырех человек:

- a. Асташкина Мария Николаевна, БПИ196
- b. Корицкий Александр Юрьевич, БПИ196
- c. Татаринов Никита Алексеевич, БПИ196
- d. Шилова Ксения Алексеевна, БПИ196

### Кто будет использовать нашу программу

Нашей программой уже пользуется вся команда “Всеобщего музыкального диктанта” и участники их мероприятий. На данный момент, на сайте чуть больше 6700 пользователей, 300 из них - это члены команды организатора мероприятий.

Дальнейшее использование нашей программы также планируется, так как организаторы планируют проводить Музыкальный диктант ежегодно. На данный момент наша команда поддерживает это решение и исправляет выявленные недостатки.

Работа с базой данных - неотъемлемая часть нашего приложения, так как там уже сейчас хранится очень много пользовательских данных. Именно поэтому мы хотим рассмотреть это веб-приложение как наш проект по предмету “Базы данных”

## Функциональные требования

В нашем веб-приложении достаточно много сущностей и разделов, для которых нужно определить функциональные требования, именно поэтому мы разобьем их на логические блоки.

### Работа с пользователем

1. Пользователь должен иметь возможность зарегистрироваться в приложении. Для этого он должен ввести имя, фамилию, почту и придумать пароль.
  - a. В случае, если пользователь не введет некоторые из этих данных регистрация не должна пройти;
  - b. Пароли должны совпадать.
2. Возможность авторизации в личном кабинете по почте и паролю.
  - a. При неверном сочетании почты и пароля система не должна пропускать пользователя.
3. У каждого пользователя есть свой личный кабинет, в котором он может использовать некоторые функции:
  - a. Изменение личных данных (аватарка, почта и пароль);
  - b. Подтверждение почты;
  - c. Заполнить анкету с дополнительными личными данными:
    - i. Страна и город;
    - ii. Телефон;
    - iii. Возраст;
    - iv. Пол;
    - v. Псевдоним пользователя для заполнения бланков во время написания диктанта;
    - vi. Образование (по желанию пользователя).
  - d. Увидеть информацию по проверенному диктанту - оценку и скан работы.
4. Каждый пользователь может просматривать страницу с новостями всей платформы.
5. Каждый пользователь может выйти из аккаунта.

## Процесс записи на диктант

1. Каждый пользователь может посещать страницу всех площадок для выбора место написания диктанта [5].
2. Каждый пользователь может посмотреть список всех диктантов, отсортированных по времени (рис. 1).
3. Пользователь может записаться на диктант.
  - a. Заявка на диктант отобразится в личном кабинете;
  - b. Изменение статуса заявки (одобрена или отклонена) также отобразится в личном кабинете.
4. Администратор может осуществлять дополнительные действия с диктантом (рис. 1):
  - a. Редактирование диктанта;
  - b. Удаление диктанта;
  - c. Закрывать регистрацию;
  - d. Получить список участников в текстовом файле.

**Расписание площадки**  
[Описание уровней сложности](#)

Время	Диктант	Аудитория	Иллюстратор	Действия
12:00	Диктант №3 – Детский (стандарт)	Большой зал	Мечин Игорь Юрьевич	<a href="#">Регистрация открыта</a> <a href="#">Удалить диктант</a> <a href="#">Редактировать диктант</a> <a href="#">Записаться!</a> <a href="#">Закреть регистрацию</a> <a href="#">Список участников</a>
	Диктант №2 – Детский (простой)	Малый зал	Сибирякова Екатерина Викторовна	<a href="#">Регистрация открыта</a> <a href="#">Удалить диктант</a> <a href="#">Редактировать диктант</a> <a href="#">Записаться!</a> <a href="#">Закреть регистрацию</a> <a href="#">Список участников</a>
13:00	Диктант №5 – Одногласный (простой)	Малый зал	Любина Марина Вячеславовна	<a href="#">Регистрация открыта</a> <a href="#">Удалить диктант</a> <a href="#">Редактировать диктант</a> <a href="#">Записаться!</a> <a href="#">Закреть регистрацию</a> <a href="#">Список участников</a>
	Диктант №7 – Одногласный (сложный)	Большой зал	Евдокименко Мария Валерьевна	<a href="#">Регистрация открыта</a> <a href="#">Удалить диктант</a> <a href="#">Редактировать диктант</a> <a href="#">Записаться!</a> <a href="#">Закреть регистрацию</a> <a href="#">Список участников</a>

рис 1. Логика работы записи на диктант

## Ролевая модель системы

1. В системе существует несколько разных ролей, каждая из которых отвечает за свою зону ответственности:
  - a. Администратор - включает в себя все права;
  - b. Координатор проекта - отвечает за все регионы, площадки и диктанты, не может писать общие новости и редактировать партнеров проекта;
  - c. Региональный координатор - отвечает за один или несколько регионов;
  - d. Координатор площадки - отвечает за одну или несколько площадок в одном регионе;
  - e. Регистратор - одобряет или отклоняет заявки на диктант на одной площадке, также может загружать работы по этой же площадке;
  - f. Пользователь - обычный пользователь, целевая аудитория системы.
2. Администратор может дать любую роль любому пользователю, координатор проекта может редактировать роль каждого пользователя, кроме администратора. Такая же логика сохраняется везде и каждая роль может редактировать все младшие роли.
3. Администратор, координатор проекта и региональный координатор могут забанить пользователя за нарушения правил системы.

## Работа с площадками для проведения диктантов

1. У пользователя с ролью координатор площадки и выше в личном кабинете есть панель управления площадками, в которой есть несколько функций:
  - a. “Добавить площадку” - страница добавления площадки с некоторыми обязательными полями (сокращенное название, полное название, код и адрес площадки). Важно отметить, что адрес площадки указывается с помощью интеграции Yandex.Map API (рис. 2).

## Добавление площадки

Сокращенное название площадки

ГБОУ №1234

Полное название площадки

Код площадки

Введите код площадки

Адрес точки

удаль

улица Удальцова, Москва, Россия

улица Удальцова, 4, Москва, Россия

деревня Удальцово, Запорожское сельское поселение, Приозерский район, Ленинградская область, Россия

улица Удальцова, 85, Москва, Россия

Удаль, Округ Гавр, Приморская Сена, Нормандия, Франция

рис 2. Добавление площадки с Yandex.Map API строкой

- b. “Все площадки” - страница со списком всех площадок в виде таблицы
- c. “Площадки на карте” - страница со всеми площадками на карте
- d. “Привязать площадки к региону” - страница, где площадки надо привязать к региону. Причем на этой странице реализован фильтр для удобства использования
- e. “Удаление площадок” - удаление площадки, если она создана по ошибке или больше не функционирует
- f. “Добавить команду к площадке” - страница добавления команды к площадке. На ней координатор площадки может добавить любого человека к команде площадки, даже если его нет в системе в качестве пользователя (рис. 3).

The image shows three sequential screenshots of a web application interface for creating a team for a platform. Each screenshot contains a form with the following sections:

- Platform Information:**
  - Полное название (Full name):** Муниципальное бюджетное учреждение дополнительного образования Детская школа искусств муниципального образования "Городской округ Ногликовский"
  - Короткое название (Short name):** МБУ ДО ДШИ пгт.Ноглики
  - Адрес (Address):** 694450, Сахалинская область, пгт. Ноглики, ул. Пограничная, д.5 А
  - Код площадки (Platform code):** 65005
- Текущая команда (Current team):**
  - Киселева Ригина Николаевна
  - Ригина Киселева
  - Лилия Вологодина
  - Марианна Исаева
- Selection and Addition Options:**
  - Выберите команду для этой площадки (Select a team for this platform):** Includes a dropdown menu "Выберите профиль" and a green "Добавить" (Add) button.
  - Добавить нового участника, если его нет на сайте (Add new participant if not on site):** Includes input fields for "Фамилия и имя участника" (Participant's surname and name) and "Должность в команде" (Position in team), and a green "Добавить" button.
- Footer:** A blue link "На страницу площадки" (To platform page).

The second and third screenshots show similar forms but with different data, such as "МБОУ ДО 'Детская школа искусств' г. Холмск" and "МБОУДО 'ДШИ №5' г.Калуги", and different team members.

рис 3. Создание команды для площадки

- г. “Статистика” - страница статистики площадок в системе. На текущий момент, в системе 222 площадки, 169 координаторов площадки и 598 людей в команде разных площадок.
2. После создания площадки создается отдельная страница для нее, которая доступна для всех пользователей в системе. Каждая страница площадки содержит несколько разделов (рис. 4):
  - а. Расписание диктантов
  - б. Регион площадки
  - в. Команда
  - г. Площадка на карте
  - е. Настройки (только для пользователей с ролью координатор площадки и выше)

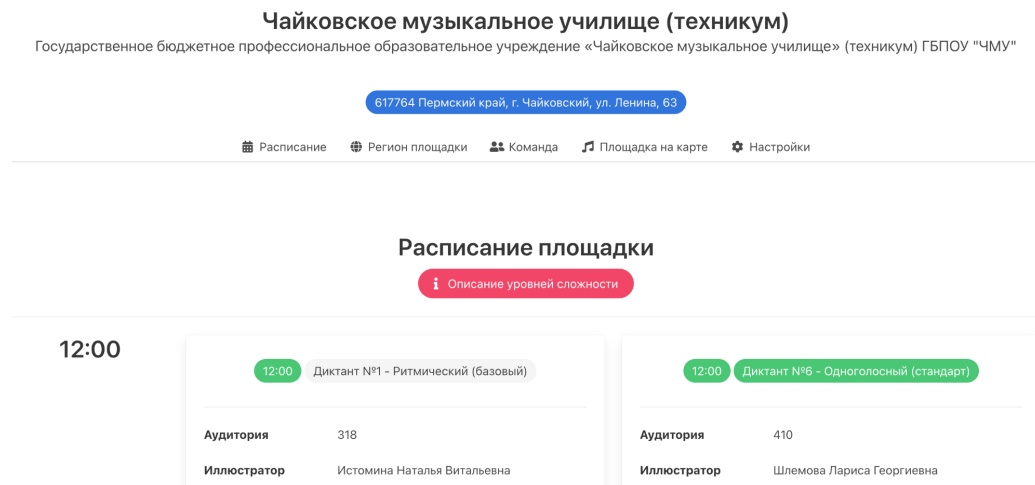


рис 4. Страница площадки [1]

3. В разделе “Настройки” пользователи с ролью координатор площадки и выше могут делать некоторые дополнительные действия:
  - а. Редактирование площадки
  - б. Список заявок на все диктанты на этой площадке в виде таблицы, можно одобрять и отклонять заявки на диктанты на этой странице.

## Работа с регионами для проведения диктанта

1. У пользователя с ролью “Региональный координатор” и выше в личном кабинете есть панель управления регионами с некоторыми функциями:
  - а. “Добавить регион” - страница добавления региона (рис. 5). На не выводится таблица со всеми существующими регионами на сайте, чтобы избежать дублей в базе данных. Для создания региона необходимо его название и фотографию (например, герб).

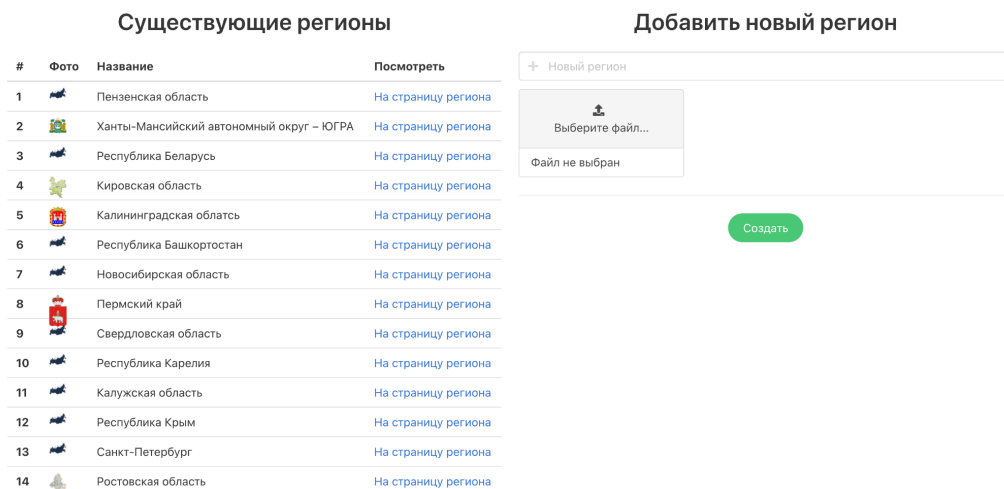


рис 5. Страница добавления региона

- b. “Привязка людей к регионам” - на этой странице можно привязать региональных координаторов к определенному региону.
  - c. “Все регионы” - страница, на которой выводятся все регионы в виде таблицы вместе с координаторами регионов
  - d. “Статистика” - страница для статистики по регионам. Сейчас в системе 46 регионов и 52 региональных координаторов. Это связано с тем, что в некоторых регионах очень много площадок (например, Москва и Московская область).
2. После создания каждого региона появляется отдельная страница региона. На ней пользователи могут увидеть список площадок и выбрать подходящую. Пример странице вы можете увидеть по ссылке [2].
3. Пользователь с ролью региональный координатор и выше может редактировать регион:
  - a. Изменить название (например, в случае ошибки)
  - b. Изменить фотографию региона
  - c. Добавить регионального партнера. У каждого региона есть свои партнеры, которые поддерживали акцию
  - d. Добавить новость на страницу региона. Это необходимо в случае каких-либо непредвиденных обстоятельств или изменений.



## События

В системе реализована сущность событий, чтобы команда “Всеобщего музыкального диктанта” могла спокойно пользоваться системой без технической поддержки. Каждый диктант привязан к событию. Именно поэтому на странице каждой площадки администратор системы может легко выбрать, какое именно событие сейчас активно и показывать корректное расписание.

1. Пользователь с ролью администратора может добавить события с помощью панели управления в личном кабинете.

## Получение результатов диктанта

1. Пользователь с ролью регистратор и выше может загрузить работу участника с помощью панели управления в личном кабинете
  - а. Система позволяет прикрепить проверенную работу к существующему пользователю, который оставлял заявку на участие в этом диктанте. Именно по заявке на диктант регистратор может понять, чья работа у него на проверке.
  - б. Однако есть возможность загрузить работу без профиля. В таком случае регистратор загружает скан работы и все данные, которые есть на бланке (ФИО или почта, или кодовое слово участника)
2. Пользователь сможет увидеть свою работу или в личном кабинете, или по данным, которые он оставил на бланке. Это можно сделать с помощью поиска своей работы на сайте [3].

## Дополнительные возможности пользователя

1. Каждый пользователь может зарегистрировать пользователя на диктант при условии, что он сам записался на определенный диктант. Это нужно в случае, если педагог ведет писать музыкальный диктант своих учеников. Для этого у пользователя в личном кабинете есть специальная кнопка “Зарегистрировать участников” (рис. 6).

## Регистрация других участников

Если Вы хотите зарегистрировать какого-то участника, который пойдет вместе с Вами, но он не может зарегистрироваться сам – эта кнопка как раз для Вас! Здесь Вы сможете зарегистрировать еще одного или нескольких людей к нам на диктант

Зарегистрировать участников +

Статус созданных участников ?

рис 6. Регистрация других участников в личном кабинете

2. Пользователь с ролью администратора имеет еще три дополнительных возможности:
  - а. “Добавить федерального партнера” - это страница добавления партнера, который после добавления отобразится на главной странице системы [4]
  - б. “Закрывать всю регистрацию” - эта функция закрывает возможность подавать заявки на всех площадках на все диктанты. Это нужно для того, чтобы люди не отправляли заявки на диктанты уже после проведения мероприятия.
  - в. “Редактировать главный штаб” - это страница для добавления самых важных фигур акции на главную страницу системы [4].
3. Отображение каждой площадки на карте для удобства пользователя (рис. 7)

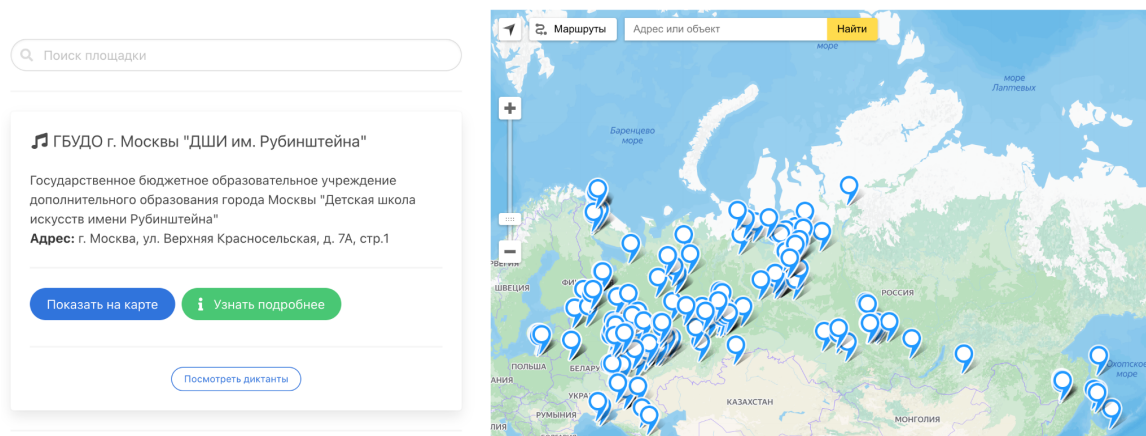


рис 7. Страница всех площадок на карте [5]

## Нефункциональные требования

1. Записаться на диктант самостоятельно можно только при наличии личного кабинета в системе;
2. Диктанты не могут быть привязаны к двум разным событиям;
3. Администратор не может лишать прав администратора других пользователей (от прав администратора можно только отказаться самостоятельно);
4. Система должна полностью соответствовать ролевой модели. Обычные пользователи не могут использовать функции регистратора и любого пользователя с ролью выше;
5. Нельзя записаться на диктант, когда регистрация закрыта (поле `is_registration_open` в таблице `dictation`);
6. Нельзя записаться на два диктанта, которые идут одновременно;
7. Один аккаунт может оставлять только одну заявку (на самого себя) на участие в диктанте. Для регистрации других людей можно использовать массовую регистрацию;
8. Нельзя зарегистрировать ни одного человека в режиме массовой регистрации, пока сам пользователь не подал ни одной заявки на участие в диктанте.
9. Нельзя воспользоваться функцией “Забыли пароль?”, не подтвердив свою почту;
10. На одну почту нельзя зарегистрировать двух пользователей, так как почта должна быть уникальной;
11. Нельзя добавить в команду региона пользователь с ролью, отличной от “Региональный координатор”;
12. Нельзя раздавать роли пользователям, которые выше, чем у текущего пользователя.

# Схема базы данных

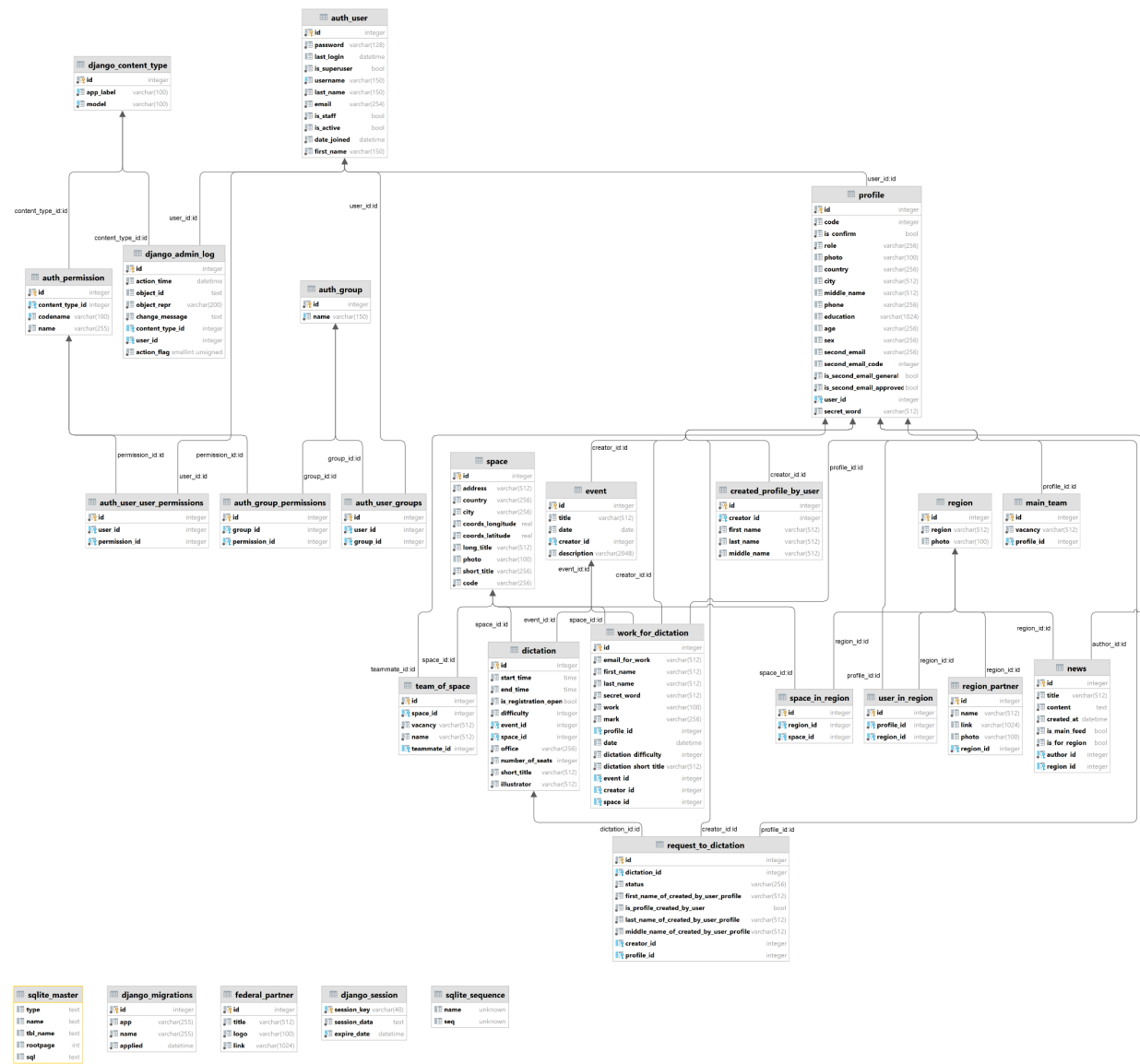


рис 8. UML-схема базы данных [6]

## Текстовые ограничения данных в виде функциональных и многозначных зависимостей

### Таблица auth\_user

1. {username} → {id, password, last\_login, is\_superuser, last\_name, email, is\_staff, is\_active, date\_joined, first\_name}

Таким образом, можно понять, что username это **UNIQUE** атрибут, и можно ограничить авторизацию с username, который уже существует.

### Таблица dictation

1. Так как атрибут space\_id есть в таблице, то можно понять, что диктант однозначно определяет площадку (невозможно сделать диктант на двух площадках на выбор или разбить его на 2 этапа на разных площадках).
2. То же можно сказать про атрибут illustrator. Он должен быть один, который однозначно определяется dictation.id.
3. {space\_id, office} → {number\_of\_seats}

Таким образом, количество мест определено этими двумя полями (то есть и максимальное количество участников)

### Таблица news

1. Каждая новость имеет ровно одного автора и ровно один регион, где она произошла.
2. {is\_main\_feed} → {is\_for\_region}

Тогда, is\_main\_feed однозначно определяет is\_for\_region. Новость на главной странице и региональная новость противоположны.

### Таблица space

1. {coords\_longitude, coords\_latitude} → {country, city, address}

Значит, должно быть соответствие между этими атрибутами.



## Таблица request\_to\_dictation

1. is\_profile\_created\_by\_user определяет какие значения будут в first\_name\_of\_created\_by\_user\_profile, last\_name\_of\_created\_by\_user\_profile, middle\_name\_of\_created\_by\_user\_profile. Если is\_profile\_created\_by\_user = 1, то поля, соответствующие именам не являются пустыми.

## Таблица team\_of\_space

1. Атрибут teammate\_id определяет, зарегистрирован ли пользователь на сайте или нет. Если он не является NULL, то пользователь зарегистрирован.



## Нормализация

База данных удовлетворяет всем свойствам 1NF, 2NF, 3NF, так как в атрибутах нет списков, значения которых могли бы использоваться в таблицах, все атрибуты зависят от ключа полностью, таблицы максимально разделены для исключения дублирования данных, таким образом, аномалии UPDATE, INSERT, DELETE исключены. Тогда, усиленная 3-я форма не выполняется, потому что в таблице auth\_user есть атрибут username, помеченный как UNIQUE. Из-за этого, он мог бы стать потенциальным ключом, и требование “ключевые атрибуты не зависят от неключевых” не выполняется. Таким образом, выполняются условия 1,2 и 3 нормальной формы.

## \*Аномалии при использовании денормализации

Предположим, что некоторые таблицы не нормализованы.

### Пример 1

Пусть таблица **space** содержит **region.region** и **region.photo** (вместо того, чтобы пары **space.id**, **region.id** были вынесены в отдельную таблицу, и таблица **region** содержала **region** и **photo**). Тогда, при добавлении новой площадки могут возникать INSERT аномалии, UPDATE аномалии и DELETE аномалии (например, при заполнении имени региона в различных строках таблицы **space** может возникнуть неточность ("Москва", "Мск", "г. Москва"...), а разбиение на таблицы решает эту проблему, потому что для каждого индекса площадки мы храним только индекс региона, название которого заполняется один раз. Если возникнет неточность в имени региона, то возникнут ошибки при операциях UPDATE и DELETE.

### Пример 2

Пусть таблица **news** содержит **first\_name**, **last\_name**, **middle\_name** автора новости (вместо того, чтобы содержать **author\_id**). Тогда, если пользователь захочет сменить имя, возникнет UPDATE anomaly, потому что старые новости, написанные им придется изменять, чего бы не случилось при нормализации.

Таким образом, нормализация приводит к исключению аномалий, что благоприятно скажется на минимизации ошибок записи.



## SQL DDL

```
CREATE TABLE "auth_group"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"name" varchar(150) NOT NULL UNIQUE);
```

```
CREATE TABLE "auth_group_permissions"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED,  
"permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "auth_permission"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"content_type_id" integer NOT NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED,  
"codename" varchar(100) NOT NULL,  
"name" varchar(255) NOT NULL);
```

```
CREATE TABLE "auth_user"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"password" varchar(128) NOT NULL,  
"last_login" datetime NULL,  
"is_superuser" bool NOT NULL,  
"username" varchar(150) NOT NULL UNIQUE,  
"last_name" varchar(150) NOT NULL,  
"email" varchar(254) NOT NULL,  
"is_staff" bool NOT NULL,  
"is_active" bool NOT NULL,  
"date_joined" datetime NOT NULL,  
"first_name" varchar(150) NOT NULL);
```

```
CREATE TABLE "auth_user_groups"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED,  
"group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "auth_user_user_permissions"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED,
"permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "created_profile_by_user"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"creator_id" integer NOT NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,
"first_name" varchar(512) NOT NULL,
"last_name" varchar(512) NOT NULL,
"middle_name" varchar(512) NOT NULL);
```

```
CREATE TABLE "dictation"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"start_time" time NOT NULL,
"end_time" time NOT NULL,
"is_registration_open" bool NOT NULL,
"difficulty" integer NOT NULL,
"event_id" integer NOT NULL REFERENCES "event" ("id") DEFERRABLE INITIALLY DEFERRED,
"space_id" integer NOT NULL REFERENCES "space" ("id") DEFERRABLE INITIALLY DEFERRED,
"office" varchar(256) NOT NULL,
"number_of_seats" integer NOT NULL,
"short_title" varchar(512) NOT NULL,
"illustrator" varchar(512) NOT NULL);
```

```
CREATE TABLE "django_admin_log"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"action_time" datetime NOT NULL,
"object_id" text NULL,
"object_repr" varchar(200) NOT NULL,
"change_message" text NOT NULL,
"content_type_id" integer NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED,
"user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED,
"action_flag" smallint unsigned NOT NULL CHECK ("action_flag" >= 0));
```

```
CREATE TABLE "django_content_type"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"app_label" varchar(100) NOT NULL,
"model" varchar(100) NOT NULL);
```

```
CREATE TABLE "django_migrations"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"app" varchar(255) NOT NULL,  
"name" varchar(255) NOT NULL,  
"applied" datetime NOT NULL);
```

```
CREATE TABLE "django_session"  
("session_key" varchar(40) NOT NULL PRIMARY KEY,  
"session_data" text NOT NULL,  
"expire_date" datetime NOT NULL);
```

```
CREATE TABLE "event"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"title" varchar(512) NOT NULL,  
"date" date NOT NULL,  
"creator_id" integer NOT NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,  
"description" varchar(2048) NOT NULL);
```

```
CREATE TABLE "federal_partner"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"title" varchar(512) NOT NULL,  
"logo" varchar(100) NOT NULL,  
"link" varchar(1024) NOT NULL);
```

```
CREATE TABLE "main_team"  
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,  
"vacancy" varchar(512) NOT NULL,  
"profile_id" integer NOT NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "news"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"title" varchar(512) NOT NULL,
"content" text NOT NULL,
"created_at" datetime NOT NULL,
"is_main_feed" bool NOT NULL,
"is_for_region" bool NOT NULL,
"author_id" integer NOT NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,
"region_id" integer NULL REFERENCES "region" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "profile"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"code" integer NOT NULL,
"is_confirm" bool NOT NULL,
"role" varchar(256) NOT NULL,
"photo" varchar(100) NULL,
"country" varchar(256) NULL,
"city" varchar(512) NULL,
"middle_name" varchar(512) NULL,
"phone" varchar(256) NULL,
"education" varchar(1024) NULL,
"age" varchar(256) NULL,
"sex" varchar(256) NULL,
"second_email" varchar(256) NULL,
"second_email_code" integer NULL,
"is_second_email_general" bool NOT NULL,
"is_second_email_approved" bool NOT NULL,
"user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED,
"secret_word" varchar(512) NOT NULL);
```

```
CREATE TABLE "region"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"region" varchar(512) NOT NULL,
"photo" varchar(100) NULL);
```

```
CREATE TABLE "region_partner"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"name" varchar(512) NOT NULL,
"link" varchar(1024) NULL,
"photo" varchar(100) NULL,
"region_id" integer NOT NULL REFERENCES "region" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "request_to_dictation"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"dictation_id" integer NOT NULL REFERENCES "dictation" ("id") DEFERRABLE INITIALLY DEFERRED,
"status" varchar(256) NOT NULL,
"first_name_of_created_by_user_profile" varchar(512) NOT NULL,
"is_profile_created_by_user" bool NOT NULL,
"last_name_of_created_by_user_profile" varchar(512) NOT NULL,
"middle_name_of_created_by_user_profile" varchar(512) NOT NULL,
"creator_id" integer NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,
"profile_id" integer NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "space"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"address" varchar(512) NOT NULL,
"country" varchar(256) NOT NULL,
"city" varchar(256) NOT NULL,
"coords_longitude" real NOT NULL,
"coords_latitude" real NOT NULL,
"long_title" varchar(512) NOT NULL,
"photo" varchar(100) NULL,
"short_title" varchar(256) NOT NULL,
"code" varchar(256) NOT NULL);
```

```
CREATE TABLE "space_in_region"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"region_id" integer NOT NULL REFERENCES "region" ("id") DEFERRABLE INITIALLY DEFERRED,
"space_id" integer NOT NULL REFERENCES "space" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE sqlite_sequence(name,seq);
```

```
CREATE TABLE "team_of_space"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"space_id" integer NOT NULL REFERENCES "space" ("id") DEFERRABLE INITIALLY DEFERRED,
"vacancy" varchar(512) NOT NULL,
"name" varchar(512) NOT NULL,
"teammate_id" integer NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED);
```

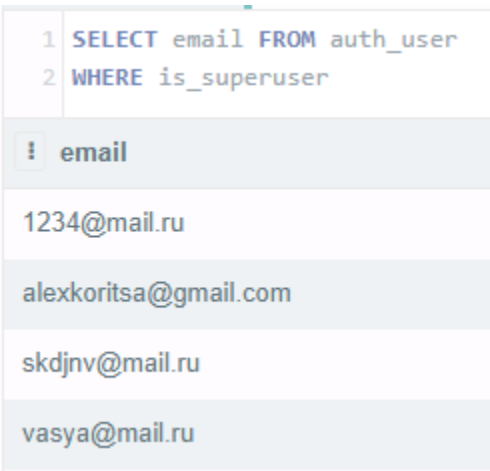
```
CREATE TABLE "user_in_region"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"profile_id" integer NOT NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,
"region_id" integer NOT NULL REFERENCES "region" ("id") DEFERRABLE INITIALLY DEFERRED);
```

```
CREATE TABLE "work_for_dictation"
("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"email_for_work" varchar(512) NOT NULL,
"first_name" varchar(512) NOT NULL,
"last_name" varchar(512) NOT NULL,
"secret_word" varchar(512) NOT NULL,
"work" varchar(100) NOT NULL,
"mark" varchar(256) NOT NULL,
"profile_id" integer NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,
"date" datetime NULL, "dictation_difficulty" integer NOT NULL,
"dictation_short_title" varchar(512) NOT NULL,
"event_id" integer NULL REFERENCES "event" ("id") DEFERRABLE INITIALLY DEFERRED,
"creator_id" integer NULL REFERENCES "profile" ("id") DEFERRABLE INITIALLY DEFERRED,
"space_id" integer NULL REFERENCES "space" ("id") DEFERRABLE INITIALLY DEFERRED);
```

*рис 9-34. Создание всех таблиц базы данных*

## SQL DML

1. Выбрать почту пользователей, которые являются superuser (рис. 35):



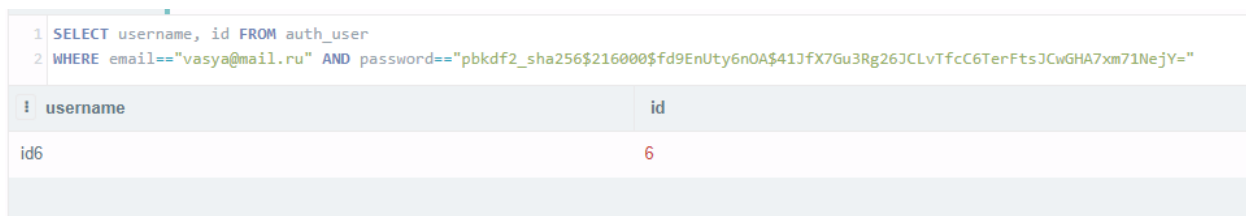
```
1 SELECT email FROM auth_user
2 WHERE is_superuser
```

email
1234@mail.ru
alexkoritsa@gmail.com
skdjnv@mail.ru
vasya@mail.ru

рис 35. SQL-запрос по условию

2. Выбрать id и username по какой-нибудь почте и паролю:

2.1 Случай, когда пользователь найден (рис. 36);



```
1 SELECT username, id FROM auth_user
2 WHERE email="vasya@mail.ru" AND password="pbkdf2_sha256$216000$fd9EnUty6n0A$41JfX7Gu3Rg26JCLvTfcC6TerFtsJCwGHA7xm71NejY="
```

username	id
id6	6

рис 36. SQL-запрос по условию

## 2.2 Случай, когда пользователь не найден (рис. 37).

```
1 SELECT username, id FROM auth_user
2 WHERE email=="vasya@mail.ru" AND password=="123"
```

рис 37. SQL-запрос по условию

## 3. Изменение пароля (рис. 38):

```
1 UPDATE auth_user
2 SET password='123'
3 WHERE email=="vasya@mail.ru"
4
5 SELECT password FROM auth_user
6 WHERE email=="vasya@mail.ru"
```

password
123

рис 38. SQL-запрос по условию



4. Просмотр username автора новости с выбором новости по определенной теме (рис. 39):

<pre>1 SELECT content, username FROM news 2 JOIN auth_user ON news.author_id==auth_user.id 3 WHERE title == 'Образ линукса'</pre>	
! content	username
<h2><strong>ПРИВЕТ!</strong></h2> <p><strong>2<sup>3 </sup>= 8</stro...	koritsa
<h2 style="box-sizing: inherit; margin: 0px; padding: 0px; font-size: 16px; font-...	koritsa
<h3>Супер новость!</h3>	koritsa

рис 39. SQL-запрос по условию

5. Изменение графы “Образование” в анкете для определенного пользователя (рис. 40):

<pre>1 UPDATE profile 2 SET education='HSE лучший вуз в мире' 3 WHERE id==( 4   SELECT auth_user.id FROM auth_user 5   WHERE username=='koritsa' 6 ) 7 8 SELECT education FROM profile 9 WHERE id==( 10  SELECT auth_user.id FROM auth_user 11  WHERE username=='koritsa' 12 )</pre>	
! education	
HSE лучший вуз в мире	

рис 40. SQL-запрос по условию

6. Выбор всех диктантов по городу с сортировкой сложности (рис. 41):

<pre>1 SELECT dictation.short_title, dictation.difficulty, space.city FROM dictation 2 JOIN space ON space.id==dictation.space_id 3 WHERE city=='Москва' 4 ORDER BY difficulty DESC</pre>		
! short_title	difficulty	city
Двухголосный (сложный)	10	Москва
Двухголосный (стандарт)	9	Москва
Двухголосный (простой)	8	Москва
Двухголосный (простой)	8	Москва
Одноголосный (стандарт)	6	Москва
Одноголосный (стандарт)	6	Москва
Одноголосный (простой)	5	Москва
Одноголосный (простой)	5	Москва
Ритмический (базовый)	1	Москва
Ритмический (базовый)	1	Москва

рис 41. SQL-запрос по условию

## 7. Просмотр диктантов, отсортированных по времени (рис. 42):

```

1 SELECT short_title, difficulty, illustrator, start_time, end_time FROM dictation
2 ORDER BY start_time

```

!	short_title	difficulty	illustrator	start_time	end_time
	Ритмический (базовый)	1	Иван Иванович Иванов	10:00:00	11:00:00
	Двухголосный (простой)	8	Александр Александрович...	10:00:00	11:00:00
	Двухголосный (простой)	8	Иван Иванович Иванов	10:00:00	11:00:00
	Двухголосный (стандарт)	9	Иван Иванович Иванов	10:00:00	11:00:00
	Одноголосный (стандарт)	6	Иван Иванович Иванов	12:00:00	13:00:00
	Одноголосный (простой)	5	Иван Иванович Иванов	12:00:00	13:00:00
	Ритмический (базовый)	1	Иван Иванович Иванов	13:00:00	14:00:00
	Одноголосный (стандарт)	6	Иван Иванович Иванов	15:00:00	16:00:00
	Двухголосный (сложный)	10	Иван Иванович Петухов	16:00:00	17:00:00
	Ритмический (базовый)	1	Александр Александрович...	16:00:00	17:00:00
	Одноголосный (простой)	5		16:00:00	17:00:00

рис 42. SQL-запрос по условию

## 8. Запись участника на самый легкий диктант (рис. 43):

```

1 INSERT INTO request_to_dictation (dictation_id, status, first_name_of_created_by_user_profile,
2                                   is_profile_created_by_user, last_name_of_created_by_user_profile,
3                                   middle_name_of_created_by_user_profile, profile_id)
4 VALUES ((SELECT id FROM dictation
5           ORDER BY difficulty
6           LIMIT 1),
7          'REVIEW', 'firstname',1,'lastname','midname',
8          (SELECT id FROM profile
9           WHERE user_id==1))
10
11 SELECT * FROM request_to_dictation

```

!	id	dictation_id	status	first_name_of_created_by_use...	is_profile_...	last_name_...	middle_r
43		2	REVIEW	Александр	1	ловам	Корицкий
44		12	REVIEW	Новый	1	XXAXA	Тест
45		12	APPROVE	Новый участник	1	ывмыв	Нвый
46		12	APPROVE	Еще	1	Участник	один
47		2	REVIEW	firstname	1	lastname	midname

рис 43. SQL-запрос по условию

9. Удалить диктант, который давно прошел (рис. 44):

```
1 DELETE FROM dictation
2 WHERE id==(
3     SELECT id FROM dictation
4     ORDER BY start_time
5     LIMIT 1)
```

рис 44. SQL-запрос по условию

10. Добавление новой площадки (рис. 45):

```
INSERT INTO space (address, country, city, coords_longitude,coords_latitude,
                  long_title, short_title,code)
VALUES ('Россия, г.Москва, Покровский б-р, 11', 'Россия', 'Москва', 55,37,
        'НИУ ВШЭ Покровский бульвар лучший вуз в мире', 'лучший вуз в мире', 1)
```

рис 45. SQL-запрос по условию

11. Добавить участника в команду (рис. 46):

```
INSERT INTO team_of_space (space_id,vacancy,name,teammate_id)
VALUES((SELECT id FROM space
        WHERE code==1), 'участник', 'Ксюша', 1)
```

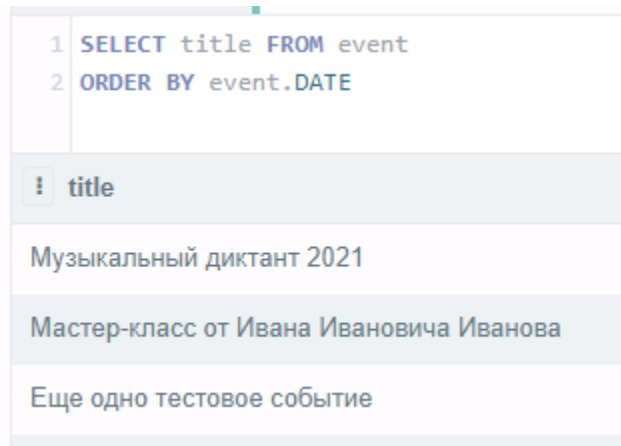
рис 46. SQL-запрос по условию

12. Присоединить площадку к региону (рис. 47):

```
INSERT INTO space_in_region (region_id,space_id)
VALUES ((SELECT id FROM region
        WHERE region.region=='Москва'),
        (SELECT id FROM space
        WHERE code == 1))
```

рис 47. SQL-запрос по условию

13. Просмотреть все события, отсортированные по дате (рис. 48):

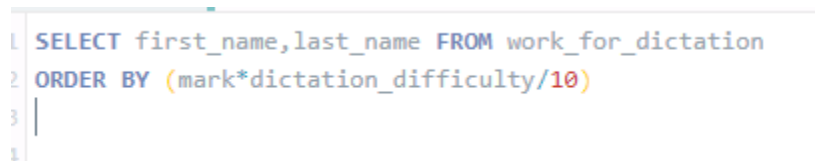


```
1 SELECT title FROM event
2 ORDER BY event.DATE
```

! title
Музыкальный диктант 2021
Мастер-класс от Ивана Ивановича Иванова
Еще одно тестовое событие

рис 48. SQL-запрос по условию

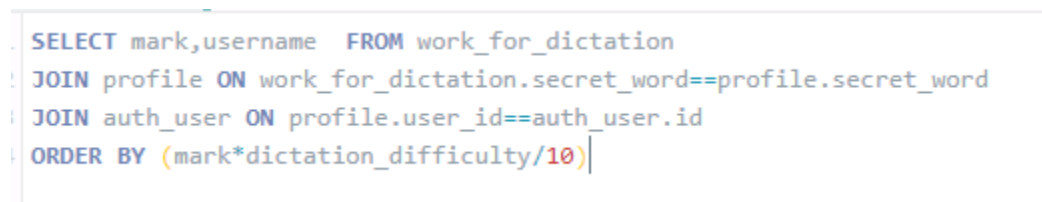
14. Распределить места участников по баллам по всем диктантам (с учетом сложности диктанта) (рис. 49):



```
1 SELECT first_name,last_name FROM work_for_dictation
2 ORDER BY (mark*dictation_difficulty/10)
3
4
```

рис 49. SQL-запрос по условию

15. Выбрать все username участников, сортированных по оценкам с учетом сложности диктанта (рис. 50):



```
1 SELECT mark,username FROM work_for_dictation
2 JOIN profile ON work_for_dictation.secret_word==profile.secret_word
3 JOIN auth_user ON profile.user_id==auth_user.id
4 ORDER BY (mark*dictation_difficulty/10)
```

рис 50. SQL-запрос по условию

16. Закрыть регистрацию на диктанты, которые уже начались (рис. 51):

```
UPDATE dictation
SET is_registration_open = 0
WHERE start_time < (SELECT TIME("now"))
```

*рис 51. SQL-запрос по условию*

## Группировка запросов в транзакции

1. Добавление записи в таблицу space и в таблицу space\_in\_region (рис. 52):

```
BEGIN TRANSACTION;
INSERT INTO space
VALUES (101, 'Россия, г. Нижний Новгород, ул. Б.Покровская', 'Россия',
        'Нижний Новгород', 56.19, 44, 'Самая большая пешеходная улица Нижнего Новгорода',
        NULL, 'Покровка', 0023);
INSERT INTO space_in_region
VALUES(201, 101,5);
END TRANSACTION;
```

рис 52. Группировка SQL-запросов в транзакции

2. Добавление в auth\_user, и в user\_in\_region, аналогично (рис. 53):

```
BEGIN TRANSACTION;
INSERT INTO auth_user
VALUES (100, '123', NULL, 0, 'username1', 'Ivanov', '123@yandex.ru', 0, 1, TIME('now'), 'Vanya');
INSERT INTO user_in_region
VALUES(200, 100, 5);
COMMIT;
```

рис 53. Группировка SQL-запросов в транзакции

3. Создание записи в таблице profile одновременно с созданием записи в таблице auth\_user. Это происходит во время регистрации пользователя (две записи создаются одновременно) (рис. 54):

```
BEGIN TRANSACTION;
INSERT INTO auth_user(id,password, is_superuser,username,last_name,
                      email,is_staff,is_active,date_joined,first_name)
VALUES(102, 'password', 1, 'username2', 'lastname',
        '123@asdkasasd', 0, 1, TIME('now'), 'firstname');
INSERT INTO profile(code, is_confirm, role,is_second_email_general,
                    is_second_email_approved,user_id,secret_word)
VALUES(123,1, 'ADMIN', 1,1,102, 'secret');
COMMIT;
```

рис 54. Группировка SQL-запросов в транзакции

4. Поставить оценку, если secret\_word при сдаче работы (work\_for\_dictation.secret\_word) совпало с profile.secret\_word (рис. 55):

```
BEGIN TRANSACTION;  
UPDATE work_for_dictation  
SET mark=10  
WHERE profile_id=1;  
COMMIT; --если секретное слово подтвердилось  
ROLLBACK; --если секретное слово не подтвердилось
```

*рис 55. Группировка SQL-запросов в транзакции*





## Материалы и ресурсы

1. Страница площадки на изображении - <https://musdict.ru/spaces/1/>.
2. Страница региона - <https://musdict.ru/regions/5/>.
3. Поиск своей работы по данным из бланка - <https://musdict.ru/dictation/result/>.
4. Главная страница системы - <https://musdict.ru/>.
5. Отображение всех площадок на карте - <https://musdict.ru/spaces/all/map/>.
6. Ссылка на схему базы данных в формате PDF -  
[https://drive.google.com/drive/u/0/folders/1HolMr8J\\_wQY45f3TVge-DLcpvzDnRf\\_5?ths=true](https://drive.google.com/drive/u/0/folders/1HolMr8J_wQY45f3TVge-DLcpvzDnRf_5?ths=true).

