МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

ОТЧЁТ

по реализации проекта для дисциплины «Базы данных»
по направлению “09.03.01 – Информатика и вычислительная техника”
(профиль: “Технологии разработки программного обеспечения ”)

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

_____
(Жуков Н. Н.)

Преподаватель: ассистент кафедры ИТиЭО

_____
(Иванова Е. А.)

Студенты 2 курса:

Круглов И. С. _____
Веремчук И. О. _____

Санкт-Петербург
2023

Оглавление

Ответственные

Круглов И.С. – разработчик проекта. В обязанности Круглова входил процесс проектирования и нормализации базы данных. При выполнении данного задания были использованы знания по следующим формам нормализации: 1НФ-3НФ.

Веремчук И.О. – разработчик проекта. В обязанности Веремчука входил процесс проектирования и создания ER-диаграммы, отчета проекта.

Предметная область

Мы имеем подробную информацию по пользователю, задаче и категории To-Do листа. В этой модели данных, каждая задача имеет уникальный идентификатор, название, описание, дату создания, дату завершения, приоритет (например, низкий, средний, высокий), статус (например, незавершенная, завершенная) и связь с определенным пользователем, к которому она относится. Также, включена сущность пользователь, которая содержит информацию о пользователях системы, такую как идентификатор, имя пользователя, пароль и адрес электронной почты. Категории могут быть добавлены, если пользователи хотят классифицировать свои задачи по определенным категориям. Каждая категория имеет уникальный идентификатор, название и связь с соответствующим пользователем.

Ход выполнения нормализации

Далее будет описан ход выполнения нормализации не по конкретно каждому пункту, а как это получилось у нас исходя из итераций:

Сначала создали примерный план, что должно быть и какие поля должны быть

На второй итерации сформировали две таблички, которые были описаны в первой итерации и уточнили поля, которые будут у нас в модели, прописали какое поле чем является (primary/foreign/unique)

В третьей итерации мы нормализовали сразу все. Разбили все на 4 таблицы, добавили для даты завершения пометочку (optional).

Первая итерация:

Пользователи - содержит информацию о пользователях системы: имя пользователя, пароль и адрес электронной почты.

Задачи - хранит основные атрибуты задач, такие как название, описание, дату создания, дату завершения, приоритет, статус и связь с определенным пользователем через внешний ключ "User ID".

Вторая итерация:

ПОЛЬЗОВАТЕЛИ:

- Имя пользователя (primary key)
- Адрес электронной почты (unique)
- Id (foreign key)
- Пароль

ЗАДАЧИ:

- Id пользователя (primary key)
- Id задачи (unique)
- Название задачи
- Описание
- Дата создания
- Дата завершения
- статус

Третья итерация:

ПОЛЬЗОВАТЕЛИ

- почта (primary)
- id

ИНФОРМАЦИЯ О ПОЛЬЗОВАТЕЛЕ

- id (primary + foreign)
- имя пользователя
- пароль

ЗАДАЧИ К ПОЛЬЗОВАТЕЛЮ

- id (primary + foreign)
- id задачи

ЗАДАЧИ

- id задачи (primary)
- название
- описание
- дата начала
- дата завершения (optional*)

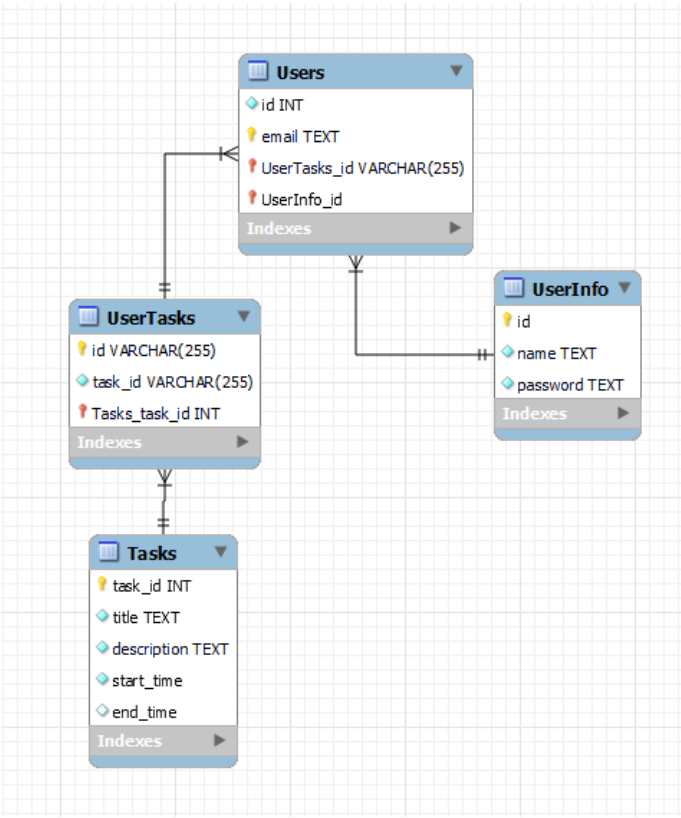
* дата начала это обязательный атрибут, а дата завершения – опциональный и это ОЧЕНЬ важно, т.к. если дата завершения будет обязательной, то модель данных уже не будет в 3НФ, однако, если дата завершения опциональна (что требуется из выдуманного тз в голове, т.к. задача может быть не задачей в привычном понимании, а эдаким «сделаю когда-то в будущем, может завтра, а может и в следующем году. А заставлять пользователя лишний раз кликать дату завершения задачи на n лет вперед – тьху и грех). Следуя из всего вышеперечисленного, дата завершения и дата начала никак не связаны, следовательно, вся модель данных после третьей проведенной итерации находится в 3 нормальной форме.

Объяснение выбранной СУБД

Изначально выбор пал на MySQL или PostgreSQL

Несмотря на то, что PostgreSQL является в последние годы более выбираемой СУБД и де-факто стандартом, однако MySQL известна своей высокой производительностью и способностью обрабатывать большие объемы данных. Она была оптимизирована для рабочих нагрузок с интенсивными операциями чтения данных и имеет быструю систему индексирования, которая помогает улучшить производительность запросов.

ER – диаграмма



Возможность использования NoSQL

NoSQL решение использовать возможно, особенно если мы хотим добиться простой гибкости в наполнении задач и превратить to-do лист во что-то наподобие блокнота, нежели непосредственно to-do листа, например, чтобы хранить множество различных видов задач с различным наполнением и структурой, а так же не очень переживать про сложности масштабирования. Однако в данном случае куда рациональнее будет использовать именно реляционную базу данных, т.к. у нас не очень сложная система с весьма строгим функционалом, да и шаблон для задач весьма типичный, несмотря на то что мы можем его легко расширить. Поэтому, при росте базы данных, добавлении новых фич и потребности в легком масштабировании и производительности (не считая сложных join-запросов, например) возможно и есть смысл миграции на NoSQL.

Исходный текст запросов

Код можно найти тут по созданию таблиц: https://pastebin.com/nc0jaMg1

```
-- Базовая информация о пользователе
CREATE TABLE Users (
   email TEXT NOT NULL
         UNIQUE
         PRIMARY KEY,
   id     UNIQUE
         NOT NULL
);



-- Расширенная информация про пользователя
CREATE TABLE UserInformation (
   id        REFERENCES Users (id)
            UNIQUE
            NOT NULL
            PRIMARY KEY,
```

```
    name     TEXT NOT NULL,
    password TEXT NOT NULL
);



-- Задачи пользователей
CREATE TABLE UserTasks (
    id       REFERENCES Users (id)
             NOT NULL
             PRIMARY KEY,
    task_id  UNIQUE
             NOT NULL
);



-- Задачи
CREATE TABLE Tasks (
    task_id         REFERENCES UserTasks (task_id)
                    UNIQUE
                    NOT NULL
                    PRIMARY KEY,
    title       TEXT NOT NULL,
    description TEXT NOT NULL,
    start_time      NOT NULL,
    end_time
);
```