Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка базы данных курсовых проектов по дисциплине «Базы данных»

Студент группы **ИУ7-66Б** _____ **Виноградов А. О.**

Руководитель курсовой работы _____ **Кузнецов Д. А.**

2023 г.

РЕФЕРАТ

Рассчетно-пояснительная записка содержит 26 с., 6 рис., 3 табл., 12 ист.

Ключевые слова: базы данных, СУБД, реляционная модель, PostgreSQL, индекс.

Целью работы: создание базы данных курсовых проектов по предмету «Базы данных».

В данной работе изучаются принципы работы с базами данных, разрабатывается приложение для хранения, изменения и удаления данных о курсовых проектах.

Результаты: программный продукт, обеспечивающий взаимодействие с реляционной базой данных.

Содержание

ВВЕДЕНИЕ

Курсовая работа[1] — практическая работа по всему курсу соответствующей научной дисциплины, выполняемая студентом в качестве формы контроля полученных и усвоенных знаний. Ежегодно десятки курсовых проектов по предмету «Базы данных» создаются студентами МГТУ им. Н.Э. Баумана.

Электронное хранение, обработка и анализ информации о курсовых проектах могут быть полезными для различных систем, например автоматизированной проверки на плагиат для выявления студентов, недобросовестно выполняющих практическую работу.

Целью данной работы является разработка программного обеспечения для хранения, редактирования и удаления данных о курсовых проектах по предмету «Базы данных».

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) проанализировать существующие методы представления данных и определить подходящий для выполнения работы вариант;

2) провести анализ существующих систем управления базами данных и определить подходящий для выполнения работы вариант;

3) спроектировать базу данных, описать ее структуру сущностей и связей между ними;

4) реализовать описанную базу данных и программное обеспечение для работы с ней;

5) исследовать влияние использования индекса базы данных на эффективность запросов подсчета количества записей таблицы.

1 Аналитический раздел

В данном разделе проводится анализ и классификация существующих систем управления базами данных. На основе анализа предметной области проводится формализация информации, подлежащей хранению, а также составляется список пользователей системы. На основе полученных данных строятся диаграмма сущность-связь в нотации Чена и диаграмма вариантов использования приложения. На основе формализованных данных, подлежащих хранению, выбирается модель базы данных.

1.1 Понятие базы данных

База данных[2] — это упорядоченный набор структурированных данных, хранящихся в электронном виде.

По способу применения базы данных разделяют на два типа.

1) **OLAP**(online analytical processing) [3] — система, используемая для обработки больших объемов данных.

2) **OLTP**(online transactional processing) [4] — система, основной упор которой делается на быструю обработку запросов, операции в режиме реального времени.

В связи с необходимостью работать с базой данных в режиме реального времени в разрабатываемом программной обеспечении будет использована технология OLTP.

1.2 Понятие системы управления базами данных

Система управления базами данных (СУБД) — это приложение, обеспечивающее создание, хранение, обновление и поиск информации в базе данных.

Существует множество классификация СУБД, использующие разные признаки классификации. Далее приведены некоторые из них.

Дореляционные СУБД

1) **Инвертированные списки**. Базы данных, основанные на использовании инвертированных списков представляют собой совокупность файлов, содержащих записи. Инвертированный список —- это организованный в вид списка специального вида индекс файла, позволяющий получить всю совокупность указателей на записи, которым соответствует заданное значение ключа индексации. В общем случае ограничения целостности базы данных отсутствуют. В некоторых системах поддерживаются ограничения уникальности значений некоторых полей, но в основном все возлагается на прикладную программу.

2) **Иерархическая модель**. Иерархические модели имеют древовидную структуру, где каждому узлу соответствует один сегмент, представляющий собой запись (кортеж полей) базы данных. Каждому сегменту может соответствовать несколько дочерних сегментов, однако запись-потомок должна иметь в точности одного предка.

3) **Сетевые СУБД**. Сетевой подход к организации данных является расширением иерархического. На формирование связи ограничений не накладывается, в отличие от иерархической модели, предполагающей ровно одного предка у записи-потомка.

Реляционные СУБД

Реляционная модель предполагает организацию данных в виде таблиц (отношений), содержащих информацию о сущностях. Каждая запись таблицы содержит уникальный индекс (ключ), используемый для поиска связанных данных в разных таблицах.

Реляционная модель состоит из трех частей:

1) **структурная часть** фиксирует, что база данных использует только одну структуру данных — n-арное отношение;

2) **целостная часть** описывает ограничения, накладываемые на отношения реляционной модели;

3) **манипуляционная часть** описывает два эквивалентных способа манипулирования реляционными данными — реляционную алгебру и реляционное исчисление.

Постреляционные СУБД

Постреляционная модель представляет собой обобщене реляционной модели. Она допускает многозначные поля таблиц, каждое из которых рассматривается как самостоятельное отношение, встроенное в главное отношение.

1.3 Формализация пользователей

В системе присутствуют три уровня пользователей.

1) **Студент** — пользователь, обладающий возможностями просмотра сущностей академических групп и тем курсовых проектов.

2) **Преподаватель** — пользователь, обладающий возможностями просмотра всех сущностей, перечисленных в разделе «Формализация данных».

3) **Администратор** — пользователь, обладающий возможностями изменения сущностей и полей базы данных, просмотра всех сущностей.

На рисунке 1 представлена диаграмма использования приложения

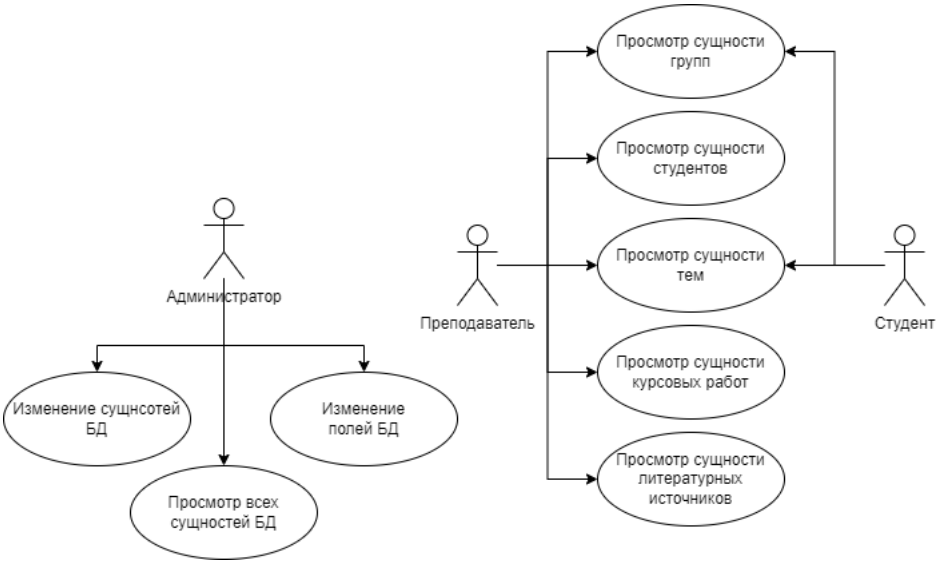

Рисунок 1 – Диаграмма использования приложения

1.4 Формализация данных

База данных состоит из следующих таблиц:

1) таблица курсовых проектов Project;

2) таблица литературных источников Source;

3) таблица академических групп Group;

4) таблица студентов Student;

5) таблица тем курсовых проектов Theme;

Каждая таблица описывает сущность базы данных. В таблицах будут расположены поля с информацией о каждой сущности. На рисунке 2 представ-

лена ER-диаграмма сущностей в нотации Чена.

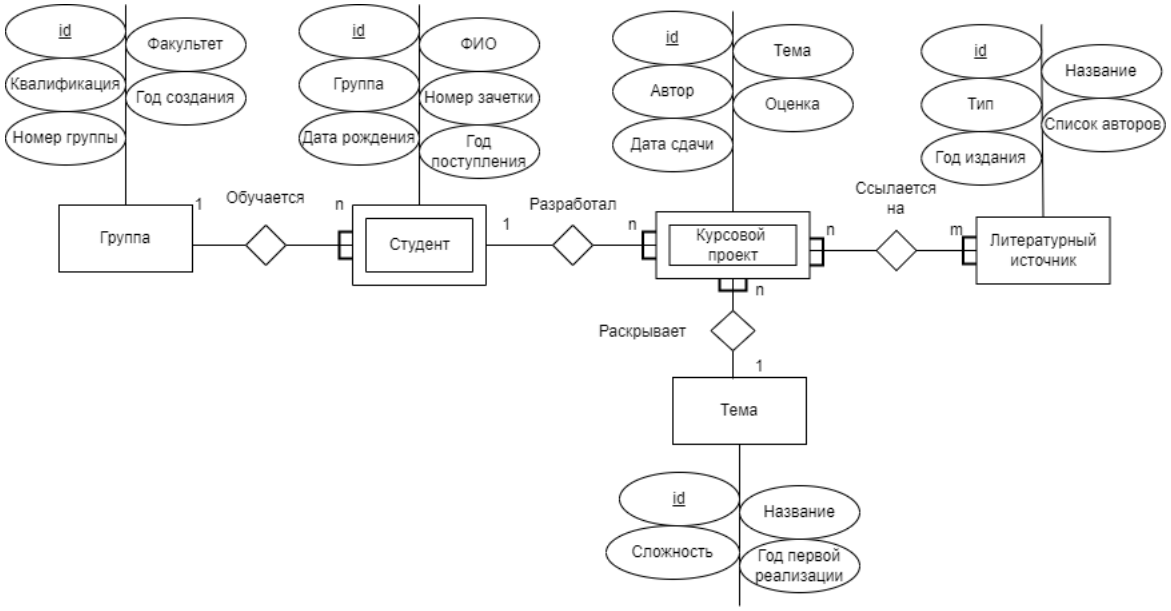

Рисунок 2 – ER-диаграмма в нотации Чена

1.5 Анализ существующих решений

Среди уже существующих проектов были выделены 3 аналога, частично решающие поставленную задачу. Их сравнительный анализ представлен в таблице 1.

Для сравнения были выбраны следующие критерии:

1) группы — возможность просмотра информации об академических группах студентов;

2) антиплагиат — невозможность приобретения готового проекта другого человека;

3) темы — возможность просмотра тем уже существующих работ для поиска вдохновения.

Из таблицы можно сделать вывод, что ни один из перечисленных аналогов не обладает функционалом, удовлетворяющим требованиям или обладает лазейками для недобросовестного выполнения курсовой работы.

Таблица 1 – Существующие решения поставленной задачи

Название проекта	**Группы**	**Антиплагиат**	**Темы**
ЭУ МГТУ	+	-	-
Studynote	-	+	+
Workspay	-	-	+

Создаваемое программное обеспечение будет предоставлять описанный функционал, являясь некоммерческим продуктом и не предоставляя возможности для плагиата.

1.6 Выбор модели базы данных

Для реализации программного продукта была выбрана реляционная модель данных, так как она обладает рядом преимуществ в рамках проекта:

1) позволяет реализовать связи между выделенными сущностями и исключить дублирование за счет использования механизмов первичного и внешнего ключа;

2) подразумевает хранение в виде таблиц, понятных конечному пользователю;

3) имеет возможность произвольного доступа ко всем элементам сущностей.

1.7 Вывод

В данном разделе были проведены анализ и классификация существующих систем управления базами данных. На основе анализа предметной области была проведена формализация информации, подлежащей хранению, а также составляется список пользователей системы. На основе полученных данных были построены диаграмма соответствующие диаграммы. На основе формализованных данных, подлежащих хранению, была выбрана реляционная модель базы данных.

2 Конструкторский раздел

В данном разделе на основе диаграммы сущность-связь в нотации Чена проводится формализация сущностей проектируемой базы данных, описывается ролевая модель. Также в разделе создается схема триггера, используемого в системе.

Таблица Group

Содержит информацию об академических группах и включает следующие поля

1) id — идентификатор группы, являющийся первичным ключом;
2) group_num — номер группы (уникальный для потока), целочисленный тип;
3) faculty — название факультета, символьный тип;
4) qualification — квалификация (бакалавр, специалист и т.д.), символьный тип;
5) creation — год создания группы, целочисленный тип.

Таблица Student

Содержит информацию о студентах и включает следующие поля:

1) id — идентификатор студента, являющийся первичным ключом;
2) group — группа студента, внешний ключ;
3) FIO — ФИО студента, символьный тип;
4) book_num — номер зачетной книжки студента (уникальный для года поступления), целочисленный тип;
5) birth — дата рождения, тип-дата;
6) enrollment — год поступления, целочисленный тип.

Таблица Theme

Содержит информацию о темах курсовых проектов и включает следующие поля:

1) id — идентификатор темы, являющийся первичным ключом;
2) name — название темы, символьный тип;
3) complexity — сложность темы по 10-бальной шкале, целочисленный тип;
4) first_time — год первой реализации темы, целочисленный тип;

Таблица Source

Содержит информацию о литературных источников курсовых проектов и включает следующие поля:

1) id — идентификатор источника, являющийся первичным ключом;
2) name — название источника, символьный тип;
3) type — тип источника (учебник, статья и т.д.), символьный тип;
4) author — список авторов, символьный тип;
5) creation — год создания источника, целочисленный тип;

Таблица Project

Содержит информацию о курсовых проектах и включает следующие поля:

1) id — идентификатор проекта, являющийся первичным ключом;
2) theme_id — тема проекта, внешний ключ;
3) author_id — автор проекта, внешний ключ;
4) mark — оценка проекта, целочисленный тип;
5) passed — дата сдачи проекта, тип-дата;

2.1 Ролевая модель

Ролевая модель в разрабатываемом программном обеспечении необходима для того, чтобы обеспечить правильную организацию работы пользователей в системе. Ее задача заключается в предоставлении каждому пользователю необходимое ему множество операций в системе.

В разрабатываемом программной продукте выделены следующие роли:

1) StUser — студент. Имеет доступ SELECT над таблицей Theme и доступ SELECT над таблицей Group.

2) TeUser — преподаватель. Имеет доступ SELECT к таблицам Theme, Group, Student, Source, Project.

3) AdmUser — администратор. Имеет все права над Theme, Group, Student, Source, Project.

2.2 Разработка триггера

В разрабатываемой системе представлен INSERT/UPDATE триггер, который при добавлении в базу нового курсового проекта или изменения уже существующего, проверяет, что год его сдачи больше или равен году первой реализации соответствующей темы. В противном случае триггер меняет год первой реализации данной темы на год сдачи проекта.

Данный триггер разработан с целью поддержания целостности данных, избавления от возможных противоречий в таблицах.

Схема триггера представлена на рисунке 3

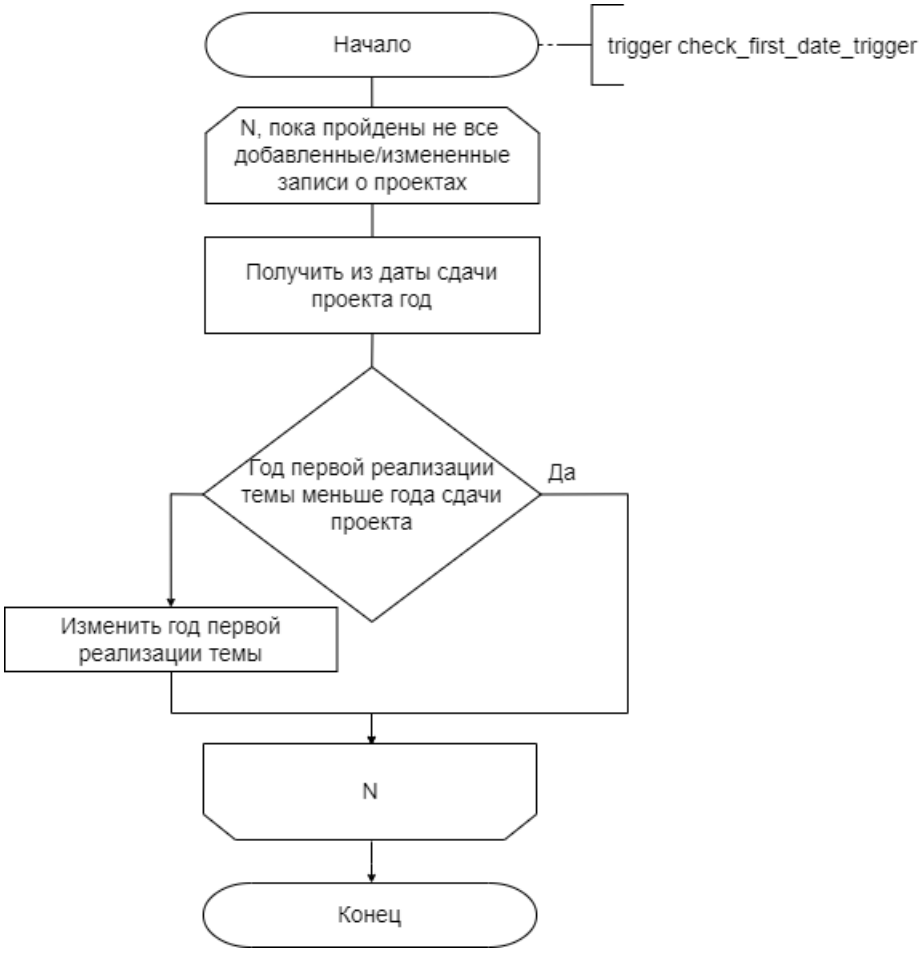

Рисунок 3 – Схема триггера

2.3 Вывод

В данном разделе на основе диаграммы сущность-связь в нотации Чена произведена формализация сущностей проектируемой базы данных, описаны все атрибуты, необходимые для представления сущностей предметной области. Также дано полное описание выделенных в системе ролей пользователей и их прав доступа. Была создана схема разрабатываемого INSERT/UPDATE триггера базы данных, выполняющего проверку целостности данных.

3 Технологический раздел

В данном разделе рассматриваются средства реализации программного продукта. Проводится анализ наиболее популярных реляционных СУБД, среди них выбирается наиболее подходящая для разработки программного продукта система. Также в разделе приводится обоснование выбора языка программирования и соответствующей библиотеки создания пользовательского интерфейса.

На основе спроектированной в предыдущих разделах ролевой модели создается ее реализация. Предоставляется код для создания ролей и выдачи им прав, а также создания триггера. Приводится пример работы разработанного приложения.

3.1 Выбор СУБД

В качестве наиболее популярных [5] реляционных СУБД чаще всего выделяют SQLite [6], MySQL [7] и PostgreSQL [8]. Рассмотрим их сильные и слабые стороны.

1) SQLite. Данная СУБД является файловой, то есть все данные хранятся в одном файле, что облегчает перемещение и использование в небольших приложениях. Но в связи с этим данная СУБД не оптимизирована для работы с многопользовательскими приложениями и большими объемами данных.

2) MySQL. Среди преимуществ данной СУБД часто выделяют ее простоту, масштабируемость и сравнительно высокую скорость работы. К минусам данной СУБД относят неполную SQL-совместимость, так как MySQL реализует не весь функционал SQL, и сравнительно невысокую оптимизацию процессов параллельного чтения.

3) PostgreSQL. В качестве главных преимуществ данной СУБД приводят полную SQL-совместимость, расширяемость и поддержку многими сторонними инструментами, связанными с СУБД. Среди недостатков выделяют операции чтения, в которых PostgreSQL может проигрывать конкурентам.

Для реализации программного продукта была выбрана СУБД PostgreSQL, так как она обладает наилучшим потенциалом для расширения и поддержкой форматов csv и json.

3.2 Средства реализации

Для реализации программного продукта был выбран язык программирования Python [9]. Простота разработки и малый объем кода, обилие библиотек и гибкость данного языка позволяют реализовать графический интерфейс и необходимый для работы с БД объем функций.

Для связи Python с СУБД PostgreSQL выбрана библиотека SQLAlchemy [10]. Данная библиотека, как и любая другая библиотека python может быть установлена при помощи встроенного пакетного менеджера pip.

3.3 Создание триггера

Создание INSERT/UPDATE триггера над таблицей Project в соответствии с результатами проектирования представлено в листинге **??**.

Листинг 3.1 – Код функции проверки года первой реализации темы

```
1 create TRIGGER check_first_date_trigger
2 AFTER INSERT or UPDATE  ON "project"
3 FOR EACH ROW
4 EXECUTE FUNCTION check_first_date();
```

Для работы триггера также была написана функция *check_first_date*, выполняющей проверку года первой сдачи темы курсовой работы в соответствии с результатами проектирования. Ее код приведен в листинге 3.3

Листинг 3.2 – Код функции проверки года первой реализации темы

```
1 CREATE FUNCTION check_first_date() RETURNS TRIGGER AS $$
2 declare first_t int;
3 begin
4   select  first_time
5         from theme
6         where id = new.theme_id
7         into first_t;
```

Листинг 3.3 – Код функции проверки года первой реализации темы (продолжение)

```
 1   IF first_t > extract('Year' from new.passed) then
 2     RAISE NOTICE 'Previous first_time: %', first_t;
 3     RAISE NOTICE 'New first_time: %', extract('year' from new.passed);
 4     RAISE NOTICE 'Updating table theme with new value: %', extract('year'
      from new.passed);
 5     update theme set first_time = extract('year' from new.passed) where id
       = new.theme_id;
 6     Return NEW;
 7   ELSE
 8     RETURN NULL;
 9   END IF;
10 END;
11 $$ LANGUAGE plpgsql;
```

3.4 Создание ролевой модели

В соответствии с разработанной в конструкторском разделе ролевой моделью был написан SQL-сценарий, выполняющий создание ролей в базе данных и выделение им прав. Код сценария представлен в листинге 3.4.

Листинг 3.4 – Код сценария создания ролей базы данных и выделения им прав

```
1 CREATE ROLE StUser LOGIN PASSWORD 'postgres';
2 GRANT SELECT ON TABLE "theme", "grouptab"  TO StUser;
3
4 CREATE ROLE TeUser LOGIN PASSWORD 'postgres';
5 GRANT SELECT ON TABLE "theme", "grouptab", "student", "source", "project",
      "source_project"  TO TeUser;
6
7 CREATE ROLE AdmUser LOGIN PASSWORD 'postgres';
8 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO AdmUser;
```

3.5 Пользовательский интерфейс

Для работы с базой данных был реализован графический интерфейс пользователя. Для создания интерфейса была использована библиотека pyqt5 [11].

Для каждой сущности базы данных были реализованы операции чтения, добавления, обновления и удаления.

Для операций чтения и удаления необходимо указать идентификатор записи в соответствующей таблицы. Для операций обновления и добавления необходимо указать значения полей, соответствующих атрибутам целевой таблицы.

Для операций массового чтения также необходимо указать количество элементов на возвращаемой странице и количество страниц, которые необходимо пропустить при выводе.

Пример работы программы приведен на рисунке 4.

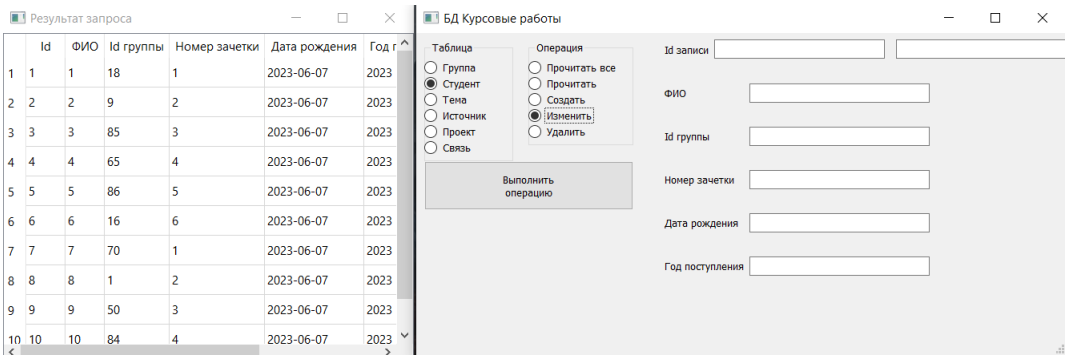

Рисунок 4 – Пример работы программы

Вывод

В данном разделе рассмотрены наиболее популярные СУБД, среди которых выбрана система,наилучшим образом подходящая для решения поставленной задачи. Приведено обоснование выбор средств реализаций, представлен код создания триггера и ролевой модели. Описан пользовательский интерфейс и приведен пример работы приложения.

4 Исследовательский раздел

В данном разделе поставлена задача исследования зависимости времени выполнения запросов, выполняющих подсчет количества записей в таблице базы данных, от наличия в базе данных индекса для соответствующего атрибуты таблицы.

Также в разделе приведены технические характеристики устройства, на котором проводилось тестирование, приведены результаты замеров времени в табличном и графическом виде. Сделан вывод о рациональности использования индекса в зависимости от синтаксиса запроса.

4.1 Технические характеристики

Замеры времени одних и тех же операций могут сильно различаться в зависимости от технических характеристик устройства, на котором проводится тестирование. Поэтому для качественного проведения исследования временных свойств системы необходимо знать основные параметры тестирующего устройства.

Ниже приведены технические характеристики устройства, на котором было проведено измерение времени работы ПО:

— операционная система Windows 10 Домашняя Версия 21H1 x86_64;

— оперативная память 8 Гбайт 2133 МГц;

— процессор Intel Core i5-8300H с тактовой частотой 2.30 ГГц, 4 физических ядра, 8 логических ядер.

4.2 Постановка задачи исследования

Индекс [12] в базе данных — любая структура данных, позволяющая сократить время нахождения записи в БД по одному или нескольким значениям полей отношения. Сокращение времени достигается за счет дополнительных затрат памяти на хранение структур данных. Также повышается время вставки, изменения и удаления, так как при выполнении данных операций системе

необходимо вносить изменения в индексы для поддержания их в актуальном состоянии.

Целью исследования является изучение зависимости времени выполнения запросов подсчета количества записей в таблице базы данных от наличия соответствующего индекса базы данных.

Для исследования были выбраны следующие запросы:

1) Запрос, выполняющий подсчет общего числа студентов, обучающихся в группе со значением id = 50. Код данного запроса представлен в листинге 4.1.

2) Запрос, выполняющий подсчет общего числа студентов в соответствующей таблице. Код данного запроса представлен в листинге 4.2.

Листинг 4.1 – Литинг запроса выполняющего поиск числа студентов обучающихся в группе со значением id

```
1 select count(group_id) from student s where group_id = 50
```

Листинг 4.2 – Литинг запроса выполняющего поиск общего числа студентов в соответствующей таблице

```
1 select count(group_id) from student s where group_id = 50
```

Также для исследования был использован индекс для атрибута *group_id* таблицы студентов.

Исследование проводилось для состояний таблицы студентов, содержащих $10^5, 2 \times 10^5, 3 \times 10^5, 4 \times 10^5, 5 \times 10^5, 6 \times 10^5, 7 \times 10^5, 8 \times 10^5, 9 \times 10^5, 10^6$ студентов.

Для каждого размера таблицы было проведено 2 измерения: время выполнения запросов без использования индекса и с его использованием.

4.3 Время выполнения запросов

В таблицах 2 – 3 продемонстрировано пользовательское время выполнения запросов с использованием индекса и без него.

Таблица 2 – Время выполнения запроса 1

Количество студентов	Время без индекса, мс	Время с индексом, мс
10^5	12.47	7.41
2×10^5	19.62	8.31
3×10^5	26.78	9.18
4×10^5	31.43	10.08
5×10^5	33.60	10.84
6×10^5	36.05	11.75
7×10^5	38.85	12.63
8×10^5	44.15	13.24
9×10^5	47.17	14.08
10^6	49.50	15.53

Таблица 3 – Время выполнения запроса 2

Количество студентов	Время без индекса, мс	Время с индексом, мс
10^5	12.44	13.27
2×10^5	18.68	20.83
3×10^5	26.67	27.63
4×10^5	30.88	30.23
5×10^5	33.45	34.84
6×10^5	35.55	36.14
7×10^5	38.44	39.77
8×10^5	42.70	43.35
9×10^5	46.89	46.91
10^6	49.21	49.15

На рисунках 5 – 6 представлена графическая интерпретация полученных результатов.

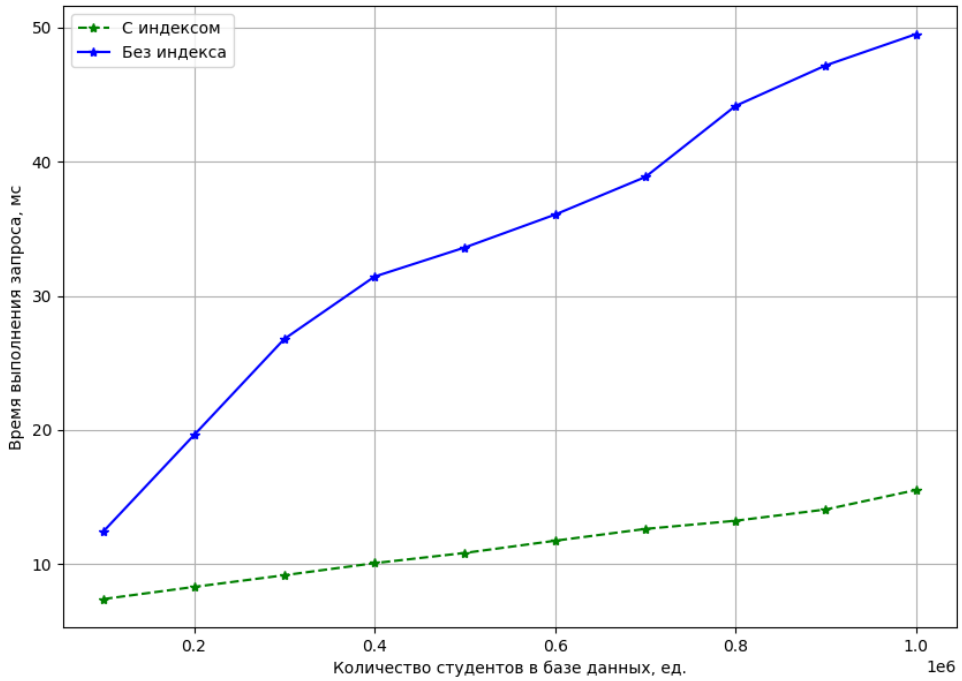

Рисунок 5 – Графическая интерпретация результатов исследования времени выполнения запроса 1

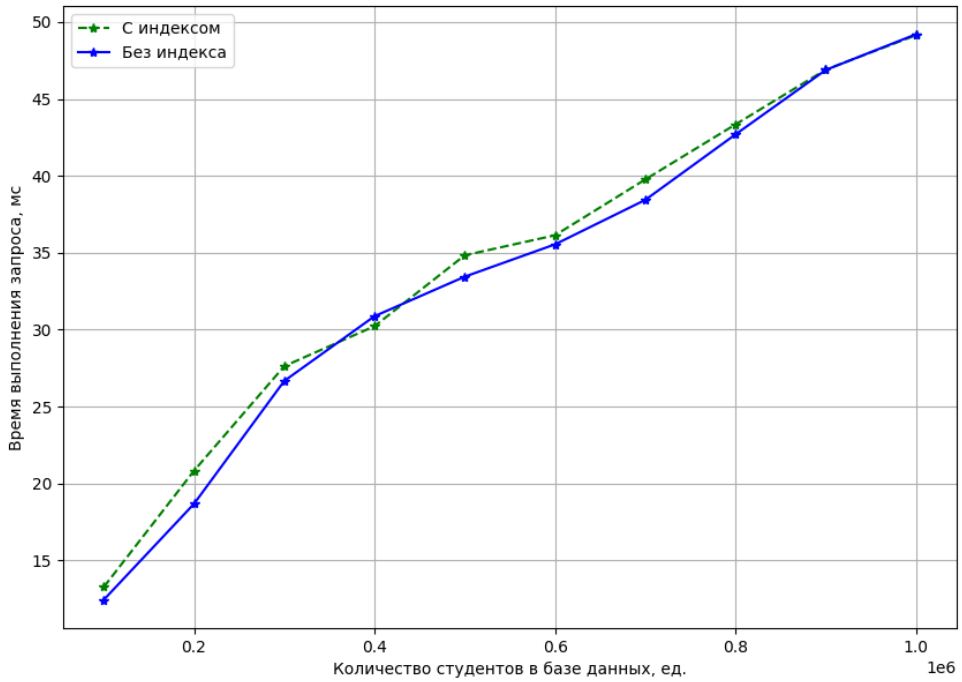

Рисунок 6 – Графическая интерпретация результатов исследования времени выполнения запроса 2

Из результатов исследования можно сделать вывод, что время выполнения запроса 1, обладающего явным указанием значения искомого поля, может быть значительно снижено с использованием индекса. Для таблицы из 10^6 записей время было сокращено более чем в 3 раза.

В случае запроса 2 различия значений времени выполнения с использованием индекса и без него отличаются слабо и, вероятно, объясняются случайной погрешностью. Также наличие индекса увеличивает объем занимаемого дискового пространства и его создание для выполнения запросов, подобных 2 не рекомендуется.

Вывод

В данном разделе было проведено исследование влияния индекса базы данных на производительность запросов вычисления количества записей в таблице.

Из результатов исследования можно сделать вывод, что в зависимости от структуры запроса использование индекса в базе данных может как уменьшать время выполнения запроса, так и не оказывать на него влияния. В обоих случаях наличие индекса повышает объем занимаемой памяти и увеличивает время работа операций вставки, изменения и удаления.

Рекомендуется применять индексы с осторожностью и проводить анализ оправданности их создания.

ЗАКЛЮЧЕНИЕ

Цель, поставленная в начале работы, была достигнута. Кроме того были выполнены все поставленные задачи:

1) были проанализированы существующие методы представления данных и определен подходящий для выполнения работы вариант;

2) был проведен анализ существующих систем управления базами данных и определен подходящий для выполнения работы вариант;

3) была спроектирована база данных, описана ее структуру сущностей и связей между ними;

4) была реализована описанная база данных и программное обеспечение для работы с ней;

5) были исследовано влияние использования индекса базы данных на эффективность запросов подсчета количества записей таблицы.

Список использованных источников

1. Курсовая работа // Большая советская энциклопедия / гл. ред. А. М. Прохоров. — 3-е изд. — М. : Советская энциклопедия, 1969—1978.

2. Database Systems The Complete Book Second Edition Hector Garcia-Molina Jeffrey D. Ullman.

3. Meier A., Kaufmann M. SQL and NoSQL Databases, Wiesbaden: Springer Fachmedien Wiesbaden GmbH. 2019. p. 176.

4. Meier A., Kaufmann M. SQL and NoSQL Databases, Wiesbaden: Springer Fachmedien Wiesbaden GmbH. 2019. p. 177.

5. The Most Popular Databases for 2022 [Электронный ресурс] // LearnSQL. URL: https://learnsql.com/blog/most-popular-databases-2022/ (Дата обращения: 18.04.2023).

6. What Is SQLite? [Электронный ресурс] // SQLite. URL: https://www.sqlite.org/index.html (Дата обращения: 18.04.2023).

7. The world's most popular open source database [Электронный ресурс] // MySQL. URL: https://www.mysql.com/ (Дата обращения: 18.04.2023).

8. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс] // PostgreSQL. URL: https://www.postgresql.org/ (Дата обращения: 18.04.2023).

9. Welcome to Python [Электронный ресурс] // URL: https://www.python.org (дата обращения: 18.04.2022).

10. The Python SQL Toolkit and Object Relational Mapper [Электронный ресурс] // SQLAlchemy. URL:https://www.sqlalchemy.org/ (Дата обращения: 18.04.2023).

11. PyQt5 5.15.9 [Электронный ресурс] // URL: https://pypi.org/project/PyQt5/ (дата обращения: 18.04.2022).

12. Garcia-Molina H., Ullman J. D., Widom J. Database Systems: The Complete Book, New Jersey: Prentice Hall. 2008.

ПРИЛОЖЕНИЕ А

Презентация к работе

Презентация содержит 11 слайдов.