**Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.04.04 – Программная инженерия | |
| **Профиль** | Разработка распределенных программных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Кринкин К.В. |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА магистра

Тема: ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ БАЗЫ ДАННЫХ TRANSRELATIONAL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студентка |  |  |  | Полушина Ю.И. |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | Фомичёва Т.Г. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты |  |  |  | Иванов И.И. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2020**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Кринкин К.В. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студентка | Полушина Ю.И. | | | |  | Группа | 4303 |
| Тема работы: Программная реализация модели базы данных TransRelational | | | | | | | |
| Место выполнения ВКР: СПбГЭТУ "ЛЭТИ", кафедра МО ЭВМ | | | | | | | |
| Исходные данные (технические требования): Поставлена задача разработки приложения, позволяющего осуществить преобразование стандартной реляционной модели в модель TR. | | | | | | | |
| Содержание ВКР:  «Введение», «Исследование предметной области», «Описание разработки», «Результаты разработки», «Оценка и защита результатов интеллектуальной деятельности», «Заключение». | | | | | | | |
| Перечень отчетных материалов: текст ВКР, иллюстративный материал. | | | | | | | |
| Дополнительные разделы: Оценка и защита результатов интеллектуальной деятельности | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | | |
|  | | |  | | | | |
| Студентка | |  | | Полушина Ю.И. | | | |
| Руководитель к.т.н., доцент | |  | | Фомичёва Т.Г. | | | |
| *(Уч. степень, уч. звание)* | |  | |  | | | |
| Консультант | |  | | Иванов И.И. | | | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Кринкин К.В. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студентка | Полушина Ю.И. |  | Группа | 4303 |
| Тема работы: Программная реализация модели базы данных TransRelational | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 05.02 – 55.03 |
| 2 | Проектирование программы | 06.03 – 12.03 |
| 3 | Разработка программы | 13.03 – 25.04 |
| 4 | Тестирование и отладка программы | 26.04 – 29.04 |
| 5 | Оформление пояснительной записки | 30.04 – 13.05 |
| 6 | Оформление иллюстративного материала | 14.05 – 18.05 |

|  |  |  |
| --- | --- | --- |
| Студентка |  | Полушина Ю.И. |
| Руководитель |  | Фомичёва Т.Г. |

**РЕФЕРАТ**

Пояснительная записка 00 стр., 00 рис., 00 табл., 00 ист., 00 прил.

БД, субд, Реляционная модель, TransRelational

Объектом исследования и разработки являются является модель TransRelational.

Цель работы – кратко (в 2-3 строки) указать цель работы.

Кратко (в 10-12) строк описать основное содержание работы, методы исследования (разработки), полученные результаты.

**ABSTRACT**

Briefly (10-15 lines) the content of graduating work is specified

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 8 |
| 1 | Современное состояние вопроса | 9 |
| 1.1. | История вопроса | 10 |
| 1.2. | Современные проблемы | 15 |
| 1.3. | Пути решения проблем | 20 |
| 2 | Описание исследования (разработки) | 25 |
| 2.1. | Принципы, материалы и методы | 30 |
| 2.2. | Теоретические исследования и расчеты | 35 |
| 2.3. | Модель, чертежи, электрические схемы | 40 |
| 3 | Результаты исследования (разработки) | 45 |
| 3.1. | Методика эксперимента | 50 |
| 3.2. | Результаты эксперимента | 55 |
| 3.3. | Сравнение и оценка результатов | 60 |
| 4 | Название дополнительного раздела | 65 |
| 4.1. | Название 1-го подраздела дополнительного раздела | 66 |
| 4.2. | Название 2-го подраздела дополнительного раздела | 66 |
| 4.3. | Название 3-го подраздела дополнительного раздела | 67 |
|  | Заключение | 68 |
|  | Список использованных источников | 69 |
|  | Приложение А. Название приложения | 72 |

*Наименования разделов приведены в качестве примера.*

**определения, обозначения и сокращения**

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

АЦП – аналого-цифровой преобразователь

ВКР – выпускная квалификационная работа

ПЗС – прибор с зарядовой связью

РДП – режим «работа дизеля под водой»

СССР – Союз Советских Социалистических Республик

Тротил – 2,4,6-тринитрометилбензол

*Определения и сокращения приводятся в алфавитном порядке. Варианты определений и сокращений даны в качестве примера.*

**введение**

Современный мир трудно представить без использования баз данных. Большинство приложений активно используют базы данных, а размеры и количество хранимых данных с каждым годом растет. На данный момент выделяют основные модели баз данных: иерархическая, сетевая, реляционная, объектно-ориентированная. Такие модели как иерархическая, сетевая, объектно-ориентированная относят к не реляционным базам данных (NoSQL).

Реляционная модель предлагает математический способ структуризации, хранения и использования данных. Отношения позволяют группировать данные как связанные наборы, представленные в виде таблиц, содержащих упорядоченную информацию и соотносящих значения и атрибуты.

Базы данных NoSQL хорошо подходят для многих современных приложений, например мобильных, игровых, интернет‑приложений, когда требуются гибкие масштабируемые базы данных с высокой производительностью и широкими функциональными возможностями, способные обеспечивать максимальное удобство использования. Но также существуют недостатки NoSQL баз данных, например, проблемы с транзакциями, что является очень важным в банковских системах.

Таким образом реляционные базы данных до сих пор используются повсеместно и развиваются. Благодаря десятилетиям исследований и разработки реляционные СУБД работают производительно и надёжно. В сочетании с большим опытом использования администраторами реляционные базы данных стали выбором, гарантирующим защиту информации от потерь.

Модель TransRelational(TR) - это подход к реализации реляционной модели. Модель TR разработана Стивеном Тареном на основе его запатентованной технологии [1], получившей название преобразования Тарена. Предполагается, что модель TR предлагает улучшенную физическую независимость данных и производительность по сравнению с различными популярными стратегиями для реализации реляционных моделей.

В настоящее время не существует СУБД, основанной на данной модели.

**Цель** данной работы состоит в реализации программы, позволяющей осуществить преобразование стандартной реляционной модели в модель TR, а также предоставляющей возможность получения и обновления данных из таблиц описанной модели.

Для достижения цели необходимо решить следующие **задачи**:

* Провести анализ модели TransRelational.
* Выявить и сформулировать требования к разрабатываемой программе.
* Разработать архитектуру программы и сценарии использования.
* Реализовать алгоритм преобразования из реляционной модели в TR модель.
* Реализовать операции реляционной алгебры в TR модели, необходимые для выборки данных из таблиц.
* Реализовать алгоритмы доступа к БД.
* Реализовать программу, удовлетворяющую всем сформулированным критериям.
* Провести отладку и тестирование разработанной программы.

**Объектом исследования** является модель TransRelational.

**Предметом исследования** являются алгоритмы преобразования реляционной модели в модель TransRelational, алгоритмы доступа к БД.

**1. Исследование предметной области**

**1.1. Обзор информационных источников**

Цель исследования предметной области － изучение и точное описание подходов к решению поставленных задач и выявление требований к будущей программе.

В рамках данного исследования был сделан упор на изучение литературы. Так как реляционная модель достаточно хорошо изучена и используется давно, существует множество источников на русском и иностранных языках.

В отличии от реляционной модели модель TransRelational практически не описана. На русском языке краткое описание модели TR можно найти в книге Кристофера Дейта «Введение в системы баз данных, 8-е издание» [2]. Информацию о реализации модели TR можно найти только в англоязычных источниках. Основным источником для изучения подходов к реализации TR модели является книга Кристофера Дейта «Go Faster! The TransRelational Approach to DBMS Implementation» [3].

В данной книге рассмотрены следующие темы:

* Технология TR и реляционная модель.
* Проблемы непосредственного отображения данных на физический уровень.
* Уровни абстракции TR модели.
* Описание ключевой идеи и преимущества TR модели.
* Таблица значений полей.
* Таблица реконструкции записей.
* Описание подходов к реализации операций обновления.
* Предпочтительные таблицы реконструкции записей.
* Cжатые столбцы.
* Слившиеся столбцы.
* Описание подходов к реализации реляционных операций.
* Описание подходов к реализации СУБД с записью на диск.

На основании изученных материалов можно сделать вывод, что для реализации TransRelational модели необходимо решить следующие задачи:

* Реализовать процесс формирования таблицы значений полей реляционной таблицы.
* Реализовать процесс формирования таблицы реконструкции записей из реляционной таблицы.
* Реализовать основные операции реляционной алгебры: селекция, проекция, Декартово произведение, объединение, пересечение, разность, соединение, деление. (Не факт, что получится реализовать все, нужно потом отредактировать)

В следующих разделах будет рассмотрено теоретическое описание реляционных баз данных, подходов к созданию БД и модель TransRelational. Вся эта информация необходима для достижения поставленных задач.

**1.2. Реляционные базы данных**

Система баз данных — это компьютеризированная система, основным назначением которой является хранение информации, предоставление пользователям средства ее извлечения и модификации.

Между физической базой данных и пользователями системы располагается уровень программного обеспечения - система управления базами данных, СУБД.

Система баз данных может быть основана на нескольких различных подходах. В данной работе рассматривается только реляционная модель. Реляционная модель основана на определенных математических и логических принципах.

Реляционная база данных – это база данных, которая рассматривается пользователями как множество переменных, значениями которых являются отношения (таблицы).

Реляционная система — это система, основанная на описанных ниже принципах:

* Данные рассматриваются пользователем как таблицы.
* Пользователю предоставляются операторы, позволяющие генерировать новые таблицы на основании уже существующих. Должны присутствовать такие операторы как селекция и проекция. Эти и подобные им операции называются операциями реляционной алгебры.

Реляционная модель определяет не только принципы хранения данных, но и способы их обработки. [5]

Реляционная алгебра – формальная система манипулирования отношениями в реляционной модели данных.

Отношение – это множество, поэтому средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми операциями, специфичными для баз данных.

Оригинальная алгебра, определенная Кодом, включает в себя восемь операций, которые делятся на два класса:

* Традиционные операции с множествами: объединение, пересечение, разность, декартово произведение;
* Специальные реляционные операции: сокращение, проекция, соединение, деление.

В состав реляционной алгебры включаются также операции присваивания и переименования атрибутов.

На рис. 1.1 приведена неформальная иллюстрация к описанию восьми первоначальных реляционных операций.

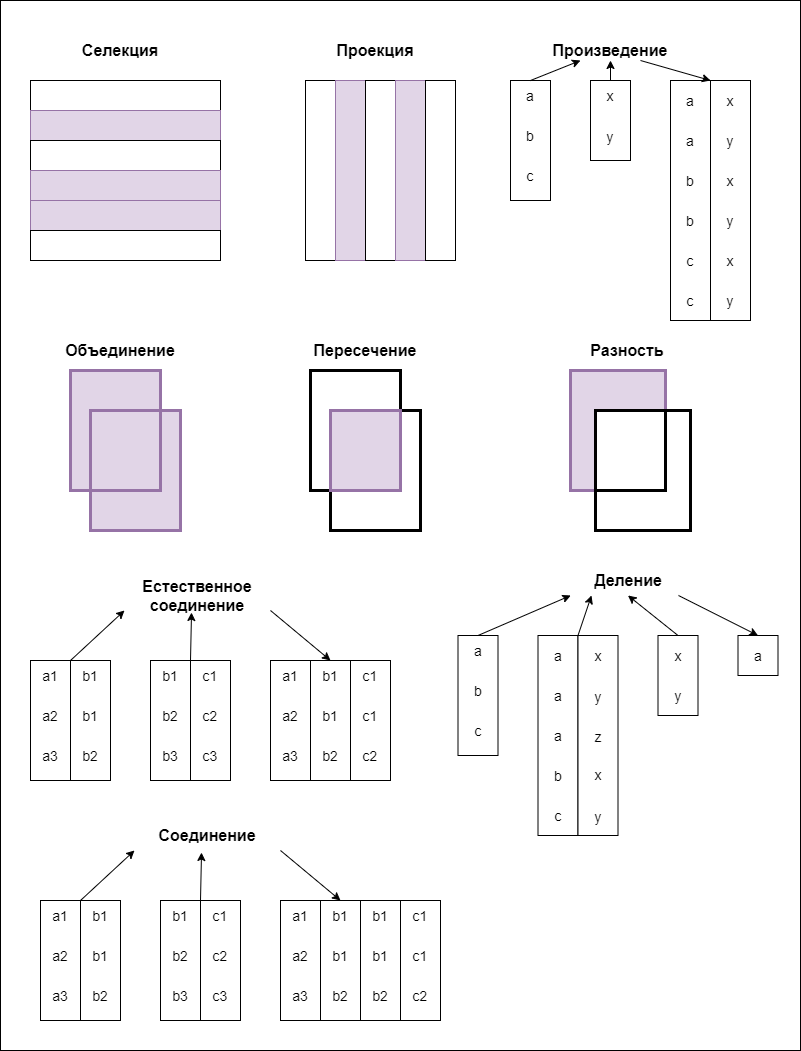
****

Рисунок 1.1 – Операции реляционной алгебры

**1.3.** **Уровни представления данных**

Современные подходы к созданию БД предполагают три уровня архитектуры: внутренний, концептуальный и внешний.

На рисунке 1.2 представлена архитектура в общих чертах.

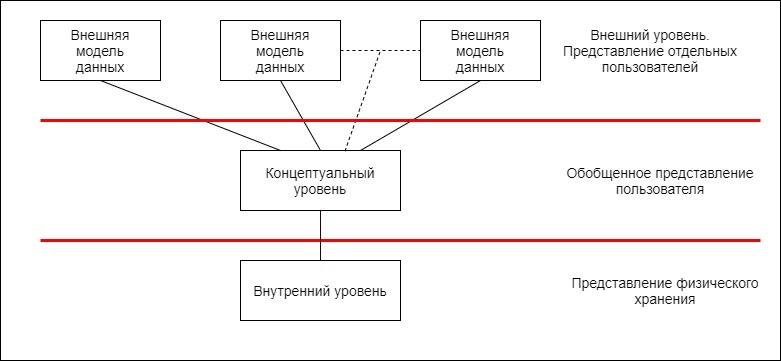


Рисунок 1.2 - Три уровня архитектуры БД

Внешний уровень (называемый также пользовательским логическим) — это индивидуальный уровень пользователя. Этот уровень определяет точку зрения на БД отдельных пользователей приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно ему.

Концептуальный уровень (называемый также логическим) — это представление всей информации БД в несколько более абстрактной форме (как и в случае внешнего представления) по сравнению с описанием физического способа хранения данных. Однако концептуальное представление существенно отличается от представления данных какого-либо отдельного пользователя. При разработке концептуальной модели усилия направлены на структуризацию данных и выявление взаимосвязей, без рассмотрения особенностей реализации и эффективности разработки.

Внутренний уровень (называемый также физическим) — это низкоуровневое представление БД, данные расположены на внешних носителях информации. Внутренняя модель определяет размещение данных, методы доступа, технику индексирования.

Трёхуровневая организация БД позволяет обеспечить логическую и физическую независимость при работе с данными. Логическая независимость предполагает возможность изменения одного приложения, без корректировки других приложений, работающих с этой же БД. Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, использующих эту БД.

Различные системы предоставляют физическую независимость данных в разной степени. Современные системы, которые используют непосредственное отображение данных на физический уровень, обеспечивают гораздо меньшую независимость данных, чем реляционные системы теоретически способны. Эту проблему может решить модель TransRelational.

**1.4.** **Модель TransRelational**

Модель TransRelational (модель TR) представляет собой иной способ реализации системы управления базами данных. В действительности, модель TR представляет собой пример применения более общей технологии, называемой преобразованием Тарена, для реализации реляционных систем. Модель TR и реляционная модель являются абстрактными моделями данных, но модель TR находится на более низком уровне абстракции, ближе к структурам физической памяти.

Метод преобразования Тарена, на который распространяется патент США [1], предназначен для использования в качестве технологии реализации для систем хранения и выборки данных многих типов, включая, например, системы хранилищ данных, инструментальные средства разработки данных, системы SQL, машины поиска Web, системы, основанные на использовании языка XML и т.д.

Модель TransRelational удобно рассматривать с точки зрения ее использования для решения задачи — обеспечения независимости от физических данных. Обеспечение независимости от данных означает проведение четкого различия между логическим и физическим уровнями системы, а проведение такого четкого различия, в свою очередь, требует наличия средств преобразования между этими двумя уровнями, с помощью которых логический уровень отображается на физический, и наоборот. Но в большинстве современных СУБД применяется такой способ преобразования, который может рассматриваться почти как взаимно-однозначный. В подобной системе все, что хранится на физическом уровне, можно рассматривать в качестве непосредственного отображения того, что пользователь видит на логическом уровне. Одним из следствий из указанного факта является то, что подобные системы в действительности не обеспечивают настолько уж значительную независимость от данных. Еще одним следствием является то, что обязательно приходится располагать данные в памяти только в одной физической последовательности, а в результате этого возникает необходимость в использовании индексов и других избыточных структур для поддержки доступа к данным, расположенным в той последовательности, которая отличается от требуемой. Кроме того, возникает и такое следствие, что для достижения приемлемой производительности требуется сложная оптимизация. Наконец, важным следствием является то, что задача администрирования базы данных становится намного более сложной, чем могла бы быть.

***Основная идея модели TransRelational***

Допустим, что r — запись некоторого файла на файловом уровне. В таком случае справедливо утверждение: хранимая форма записи r состоит из двух логических различимых частей – множества значений полей и множества «связующих» данных, позволяющих связать друг с другом эти значения полей, причем для физического хранения каждой из этих частей можно воспользоваться широким спектром возможностей [2].

В системах с непосредственным отображением эти две части хранятся вместе. В таких системах связующая информация задается по принципу физической близости. В отличие от этого, в модели TR эти две части хранятся отдельно — значения полей находятся в таблице значений полей, а связующая информация — в таблице реконструкции записей. А именно такое разделение различает модель TR от предыдущих подходов реализации реляционной модели и является основным источником многочисленных преимуществ, которые способна предоставить модель TR.

Преимущества модели TransRelational:

* Модель TR обеспечивает гораздо большую независимость от данных по сравнению с той, которая достигнута или может быть достигнута в системах с непосредственным отображением.
* В модели TR данные, по существу, хранятся во многих разных физических последовательностях одновременно, поэтому исключается необходимость в использовании индексов и тому подобных структур.
* Производительность модели TR на несколько порядков выше по сравнению с системами непосредственного отображения. В частности, производительность операции соединения является линейной.
* Администрирование системы в значительной степени упрощается, поскольку гораздо реже приходится принимать субъективные решения.
* Запросы, требующие поиска конкретного значения, могут быть реализованы с помощью эффективного бинарного поиска.

***Уровни абстракции модели TransRelational***

Реляционная система, реализованная с использованием модели TR, может рассматриваться как охватывающая три уровня абстракции: реляционный уровень, файловый уровень и уровень модели TR.

На реляционном уровне данные представлены в виде отношений, которые обычным образом составлены из кортежей и атрибутов.

На уровне TR данные представлены с помощью различных внутренних структур модели TR, называемых таблицами, а сами эти таблицы состоят из строк и столбцов. Указанные таблицы, строки и столбцы не являются реляционными таблицами с теми же именами, а также непосредственно не соответствуют отношениям, кортежам или атрибутам на пользовательском уровне. Этот уровень наиболее близок к физическому уровню в классической архитектуре БД, но все еще является абстрактным.

Файловый уровень представляет собой уровень перенаправления между другими двумя уровнями — отношения верхнего уровня отображаются на файлы среднего уровня, а затем эти файлы отображаются на таблицы низкого уровня. Кроме того, указанные файлы состоят из записей и полей; записи соответствуют кортежам, а поля — атрибутам верхнего уровня.

В табл. 1.1. приведен пример файла с данными о студентах. Пример содержит информацию о номере зачетной книжки (№ЗК), фамилии, номере группы и даты защиты студента. В таком файле записи имеют упорядочение сверху вниз, а поля — упорядочение слева направо, как показано с помощью номеров записей и номеров полей в таблице. Рассматриваемое упорядочение выбрано произвольным образом, поэтому, например, данное отношение можно было бы отобразить на любой из 2880 (5! x 4!) различных файлов, при этом, не теряя какой-либо информации и связи между данными.

Таблица 1.1 - Исходный файл с данными

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Последовательность полей:*** | | ***1*** | ***2*** | ***3*** | ***4*** |
| ***Последовательность записей:*** |  | №ЗК | Фамилия | Номер группы | Дата защиты |
| 1 | N448 | Иванов | 4303 | 06.06.2020 |
| 2 | N554 | Петров | 4302 | 05.06.2020 |
| 3 | N221 | Сидоров | 4304 | 03.06.2020 |
| 4 | N123 | Попов | 4301 | 01.06.2020 |
| 5 | N300 | Кузнецов | 4305 | 05.06.2020 |

Файл, подобный приведенному в табл. 1.1, может быть представлен с помощью таблиц на уровне TR и реконструирован из этих таблиц. При этом следует учитывать тот факт, что все возможные различные версии одного и того же файла могут быть реконструированы из одних и тех же таблиц TR одинаково легко. Это означает, что в разных версиях файла упорядочение записей и полей может отличаться, а содержание остается одним и тем же. В этих таблицах TR строки имеют упорядочение сверху вниз, а столбцы — упорядочены слева направо. В частности, строки таблиц TR не имеют какого-либо взаимно однозначного соответствия записям на файловом уровне, а в силу этого и не имеют какого-либо взаимно однозначного соответствия кортежам на реляционном уровне.

***Таблица значений полей***

В табл. 1.2. приведена таблица TR, называемая таблицей значений полей, которая соответствует файлу в табл. 1.1.

Стоит обратить внимание на то, что ее строки не соответствуют каким-либо очевидным образом тем записям, которые показаны в табл. 1.1.

Таблица 1.2 - Таблица значения полей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Последовательность столбцов:*** | | ***1*** | ***2*** | ***3*** | ***4*** |
| ***Последовательность строк:*** |  | №ЗК | Фамилия | Номер группы | Дата защиты |
| 1 | N123 | Иванов | 4301 | 01.06.2020 |
| 2 | N221 | Кузнецов | 4302 | 03.06.2020 |
| 3 | N300 | Петров | 4303 | 05.06.2020 |
| 4 | N448 | Попов | 4304 | 05.06.2020 |
| 5 | N554 | Сидоров | 4305 | 06.06.2020 |

Таблица значений полей — это единственная таблица TR, которая содержит пользовательские данные как таковые. Все остальные таблицы содержат внутреннюю информацию, которая имеет смысл только для модели TR, но непосредственно не касается пользователя. Формируется таблица значений полей с помощью сортировки: каждый столбец таблицы содержит значения из соответствующего поля файла, отсортированные в порядке возрастания. Стоит отметить, что независимо от первоначального расположения записей в файле всегда формируется одна и та же таблица значений полей.

Использование такой таблиц предоставляет следующие преимущества:

* Т.к. каждый атрибут отсортирован сразу в обоих направлениях - возрастающем и убывающем, таблица значений предоставляет множество различных вариантов сортировки одновременно. Пользовательские запросы с конструкцией ORDER BY будут выполняться быстрее, для этого не требуется ни сортировка на этапе выполнения, ни индекс.
* Запросы, требующие поиска конкретного значения, могут быть реализованы с помощью эффективного бинарного поиска. Бинарный поиск имеет логарифмическую сложность O(logN) против последовательного поиска со сложностью O(n).

***Таблица реконструкции записей***

Для того чтобы иметь возможность реконструировать приведенный в табл. 1.1 файл из таблицы значений полей в табл. 1.2, требуется еще одна таблица — таблица реконструкции записей, приведенная в табл. 1.3.

Таблица 1.3 - Таблица реконструкции записей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Последовательность столбцов:*** | | ***1*** | ***2*** | ***3*** | ***4*** |
| ***Последовательность строк:*** |  | №ЗК | Фамилия | Номер группы | Дата защиты |
| 1 | 4 | 3 | 1 | 1 |
| 2 | 5 | 5 | 3 | 2 |
| 3 | 2 | 2 | 5 | 5 |
| 4 | 1 | 1 | 2 | 3 |
| 5 | 3 | 4 | 4 | 4 |

Таблицы значений полей и реконструкции записей имеют такое же количество строк и столбцов, сколько записей и полей, соответственно, имеется в файле в табл. 1.1. Данные в таблице реконструкции записей представляют собой номера строк, а они могут рассматриваться как указатели на строки таблицы значений полей.

***Реконструирование записей по таблицам***

Допустим, необходимо получить информацию о студенте с номером зачетной книжки “N300”. Для понимания, как используется таблица реконструкции записей, рассмотрим последовательность действий, описанную ниже:

1. Перейти в ячейку [3, 1] таблицы значений полей и прочитать значение, а именно “N300”. Данное значение является первым значением поля в записи с данными о студенте в исходном файле.
2. Перейти в ячейку [3, 1] таблицы реконструкции записей и прочитать значение, а именно номер строки “2”. Этот номер строки означает значение следующего поля “Фамилия” в записи студента. Значение поля можно найти в позиции “Фамилия” второй строки таблицы значений полей, иными словами, в ячейке [2, 2] таблицы значений полей. Таким образом, можно получить фамилию студента “Кузнецов”.
3. Перейти в ячейку [2, 2] таблицы реконструкции записей и прочитать номер строки “5”. Значение следующего поля “Номер группы” в реконструируемой записи студента находится в ячейке [5, 3]. Таким образом, можно получить номер группы “4305”.
4. Перейти в ячейку [5, 3] таблицы реконструкции записей и прочитать значение “4”. Значение следующего поля “Дата защиты” находится в ячейке [4, 4]. Таким образом, можно получить дату защиты “05.06.2020”.
5. Перейти в ячейку [4, 4] таблицы реконструкции записей и прочитать значение “3”. Может показаться, что значением следующего поля должно быть значение 5, но так как записи студентов имеют только четыре поля, “пятое” поле замыкает цикл и становится первым. Таким образом, значение следующего поля “Дата защиты” находится в ячейке [5, 1]. Так как с этого значения начиналось считывание записей, поэтому весь процесс останавливается.

Приведенная выше последовательность операций приводит к реконструкции одной конкретной записи из файла студентов под номером 5 в табл. 1.1.

Указатели номеров строк, по которым осуществляется переход в приведенном выше примере, образуют кольцо, а фактически образуют два изоморфных кольца: в таблице значений полей и в таблице реконструкции записей. Такие кольца часто называют зигзагами, а алгоритм реконструкции - алгоритмом зигзага.

***Формирование таблицы реконструкции записей***

Для формирования таблицы реконструкции записей в первую очередь необходимо упорядочить с помощью сортировки исходный файл. В качестве примера рассмотрим исходный файл в табл. 1.1. Для каждого столбца выполним перестановку, соответствующей упорядочению с возрастанием атрибута. В результате получим данные по этим перестановкам, которые приведены в таблице перестановок в табл. 1.4. перестановок, в которой ячейка [i, j] содержит номер в файле студентов той записи, которая присутствует в i-й позиции, если файл отсортирован по возрастанию значений в j-м поле.

Таблица 1.4 - Таблица перестановок

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | №ЗК | Фамилия | Номер группы | Дата защиты |
| 1 | 4 | 1 | 4 | 4 |
| 2 | 3 | 5 | 2 | 3 |
| 3 | 5 | 2 | 1 | 2 |
| 4 | 1 | 4 | 3 | 5 |
| 5 | 2 | 3 | 5 | 1 |

Далее для каждого столбца необходимо выполнить обратную перестановку. Обратной перестановкой a−1 к перестановке a называется такая перестановка, что: (a−1a)i = (aa−1)i = i. Применим это правило к каждой из перестановок и получим следующую таблицу, приведенную в табл. 1.5.

Таблица 1.5 - Таблица обратных перестановок

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | №ЗК | Фамилия | Номер группы | Дата защиты |
| 1 | 4 | 1 | 3 | 5 |
| 2 | 5 | 3 | 2 | 3 |
| 3 | 2 | 5 | 4 | 2 |
| 4 | 1 | 4 | 1 | 1 |
| 5 | 3 | 2 | 5 | 4 |

Теперь можно приступить к формированию таблицы реконструкций записей. Рассмотрим последовательность действий, описанную ниже:

1. Перейти в ячейку [1,1] таблицы обратных перестановок и считать значение r = “4”. Перейти в ячейку справа [1, 2] и прочитать значение “1”. Перейти в r-ю строку таблицы реконструкции записей, т.е. в 4 строку и поместить туда значение “1” в ячейку [4,1].
2. Перейти в ячейку [2,1] таблицы обратных перестановок и считать значение r = “5”. Перейти в ячейку справа [2, 2] и прочитать значение “3”. Перейти в r-ю строку таблицы реконструкции записей, т.е. в 5 строку и поместить туда значение “3” в ячейку [5,1].
3. Перейти в ячейку [3,1] таблицы обратных перестановок и считать значение r = “2”. Перейти в ячейку справа [3, 2] и прочитать значение “5”. Перейти в r-ю строку таблицы реконструкции записей, т.е. в 2 строку и поместить туда значение “5” в ячейку [2,1].
4. Перейти в ячейку [4,1] таблицы обратных перестановок и считать значение r = “1”. Перейти в ячейку справа [4, 2] и прочитать значение “4”. Перейти в r-ю строку таблицы реконструкции записей, т.е. в 1 строку и поместить туда значение “4” в ячейку [1,1].
5. Перейти в ячейку [5,1] таблицы обратных перестановок и считать значение r = “3”. Перейти в ячейку справа [5, 2] и прочитать значение “2”. Перейти в r-ю строку таблицы реконструкции записей, т.е. в 3 строку и поместить туда значение “2” в ячейку [3,1].

Выполнение всех этих шагов для каждого столбца позволит получить таблицу реконструкции записей, которая приведена выше в табл. 1.3.

***Уникальные таблицы***

Следует отметить, что приведенная в табл. 1.2. таблица значений полей содержит избыточную информации, например, дата защиты «05.06.2020» встречается два раза. Так как существует как минимум два варианта перестановки для данного столбца, то таблица реконструкции записей также не является уникальной. Применительно к каждому конкретному отношению пользовательского уровня всегда имеются определенные таблицы реконструкции записей, являющиеся «предпочтительными», которые обладают дополнительными полезными свойствами перед другими таблицами реконструкции записей. Составление таких «предпочтительных» таблиц решает проблему не уникальности таблиц реконструкции записей. Для устранения избыточности таблицы значений полей существует метод называемый «сжатые столбцы».

***Сжатые столбцы***

Идея способа заключается в том, чтобы в таблице значений полей не хранить дублирующие значения в столбце. Для дальнейшего реконструирования первоначальной таблицы необходимо хранить некоторую дополнительную информацию. Один из способов состоит в том, чтобы вместе с каждым значением поля в каждом сжатом столбце таблицы значений полей хранить данные о диапазоне номеров строк в несжатой версии этой таблицы. Таблица реконструкции записей не изменяется. В табл. 1.6. приведена сжатая таблица значений полей с указанием диапазонов строк для вышерассмотренного примера.

Таблица 1.6 – Сжатая таблица значений полей с указанием диапазонов строк

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | ***1*** | ***2*** | ***3*** | ***4*** |
|  |  | №ЗК | Фамилия | Номер группы | Дата защиты |
| 1 | N123 | Иванов | 4301 | 01.06.2020 **[1:1]** |
| 2 | N221 | Кузнецов | 4302 | 03.06.2020 **[2:2]** |
| 3 | N300 | Петров | 4303 | 05.06.2020 **[3:4]** |
| 4 | N448 | Попов | 4304 | 06.06.2020 **[5:5]** |
| 5 | N554 | Сидоров | 4305 |  |

Можно заметить, что при сжатии таблицы значений полей информация о взаимно однозначном соответствии между ячейками этой таблицы и ячейками таблицы реконструкции записей уничтожается. Поэтому при реконструировании чтение записи будет происходить не из конкретной ячейки таблицы значений полей, соответствующей ячейке из таблицы реконструкции записей, а из ячейки, которая содержит диапазон строк, включающий искомую строку.

Данный метод может быть применен в качестве способа совместного использования значений полей в нескольких записях, но все рассматриваемые записи находятся в одном файле. В случае, когда все рассматриваемые записи могут находиться в одном или в разных файлах, применяется метод «слившиеся столбцы».

***Слившиеся столбцы***

Основная идея этого способа состоит в том, что разные поля на файловом уровне могут соответствовать одному и тому же столбцу таблицы значений полей на уровне TR, при условии, что все рассматриваемые поля имеют один и тот же тип данных. Данные метод активно используется при реализации таких реляционных операций, как: объединение, пересечение, разность.

Рассмотрим пример таблицы оценок студентов, где в двух столбцах содержится много дублирующей информации. Пример приведен в табл. 1.7.

Таблица 1.7 – Таблица оценок студентов

|  |  |  |
| --- | --- | --- |
| Фамилия студента | Лабораторные работы | Экзамен |
| Петров | 5 | 5 |
| Кузнецов | 5 | 4 |
| Сидоров | 3 | 4 |
| Иванов | 5 | 5 |
| Попов | 4 | 4 |

Можно заметить, что столбцы «Лабораторные работы» и «Экзамен» имеют один типа данных и дублирующую информацию, после выполнения слияние столбцов получена таблица значений полей, приведенная в табл. 1.8.

Таблица 1.8 – Таблица значения полей после слияния второго и третьего столбца

|  |  |
| --- | --- |
| Фамилия студента | Лабораторные работы + Экзамен |
| Иванов | 3 **[1 : 1] [ : ]** |
| Кузнецов | 4 **[2 : 2] [1 : 3]** |
| Петров | 5 **[3 : 5] [4 : 5]** |
| Попов |  |
| Сидоров |

Первый диапазон строк показывает, в каких строках несжатой таблицы значений полей соответствующая оценка является оценкой за лабораторные работы, а второй диапазон показывает, в каких строках несжатой таблицы значений полей соответствующее значение является оценкой за экзамен.

Таблица реконструкции записей для этого способа не меняется. Разница в процессе реконструирования заключается лишь в том, что для конкретного столбца в таблице реконструкции нужно будет искать значение внутри первого или второго диапазона.

**1.5.** **Выводы**

В процессе исследования предметной области рассмотрены основные понятия в области реляционных баз данных, а также изучена модель TransRelational, выявлены преимущества этой модели и подходы к ее программной реализации. Сформулированы задачи, которые необходимо решить для реализации модели TransRelational.

**2. Описание разработки**

**2.1. Алгоритмы преобразования в модель TR**

В предыдущем разделе формирование таблиц значений полей и реконструкций записи было рассмотрено на примере. В данном разделе приведено детальное описание алгоритмов, применение которых необходимо для преобразования реляционной модели в модель TransRelational.

***Формирование таблицы значений полей***

Для каждого столбца значения упорядочиваются по возрастанию.

***Формирование таблицы реконструкций записей***

1. Каждый столбец сортируется по возрастанию. Формируется таблица перестановок;
2. Для каждого столбца выполняется в таблице перестановок обратная перестановка. В результате появляется таблица обратных перестановок;
3. Для каждого столбца выполняется шаг 4;
4. Для каждой строки выполняется шаг 5;
5. Перейти в ячейку [i, j] таблицы обратных перестановок. Пусть эта ячейка содержит значение r. Кроме того, допустим, что следующая ячейка справа [i, j + 1] содержит значение r1. Перейти в r-ю строку таблицы реконструкций записей и поместить значение r1 в ячейку [r, j].

На рис 2.1 приведена блок-схема данного алгоритма:

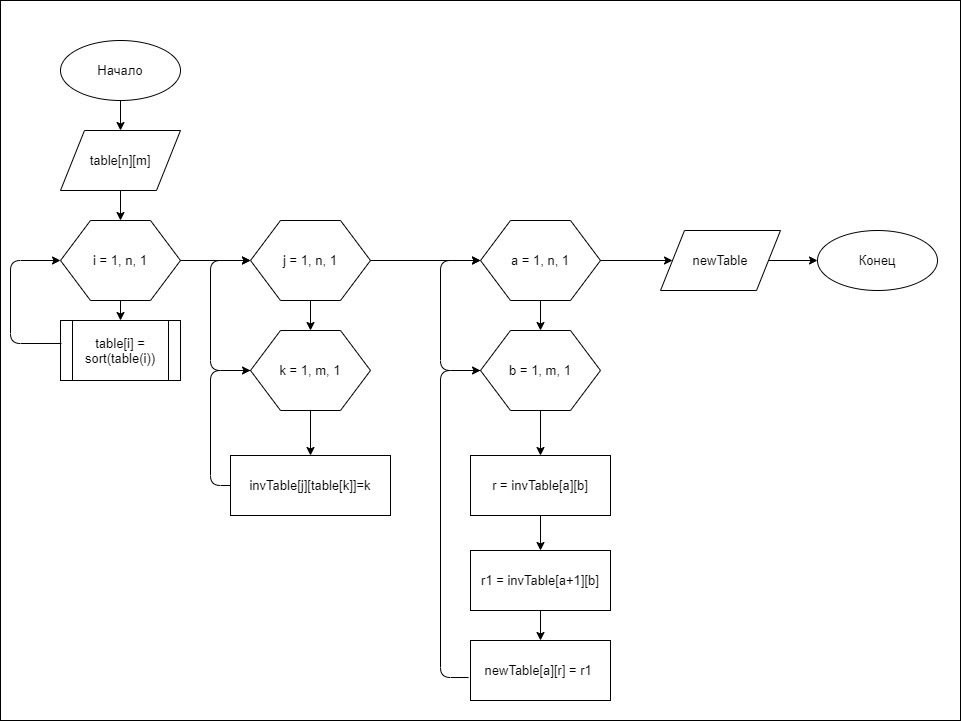


Рисунок 2.1 - блок схема алгоритма формирования таблицы реконструкции

***Реконструкция записи (алгоритм зигзага):***

1. Допустим, что j – начальный столбец, а [i, j] – стартовая ячейка;
2. Для каждого столбца выполнить шаг 3;
3. Перейти в ячейку [i, j] таблицы реконструкции записей и прочитать номер строки r. Прочитать в таблице значений полей в ячейке [r, j + 1] искомое значение.

**2.2. Реализация реляционных операторов**

В данном разделе описаны алгоритмы TR для реализации основных реляционных операторов.

***Селекция***

Поиск конкретного значения с помощью оператора “ =”.

Пример запроса: SELECT \* FROM STUDENTS WHERE STUDENTS.GROUP\_NUMBER = 4303;

В этом случае необходимо выполнить следующую последовательность действий:

1. В таблице значений полей методом бинарного поиска для заданного столбца j определяется стартовая ячейка [i, j];
2. Для каждого столбца выполнить шаг 3;
3. Перейти в ячейку [i, j] таблицы реконструкции записей и прочитать номер строки r. Прочитать в таблице значений полей в ячейке [r, j + 1] искомое значение.

Поиск с помощью оператора “<”.

Пример запроса: SELECT \* FROM STUDENTS WHERE STUDENTS.GROUP\_NUMBER <4303;

В этом случае необходимо выполнить следующую последовательность действий:

1. В таблице значений полей выполнить линейный поиск для заданного столбца.
2. Реконструировать все соответствующие записи, обнаруженный в процессе этого поиска (см. алгоритм зигзага).
3. Остановиться, как только будет достигнута ячейка, содержащая условие.

Для оператора “>” алгоритм будет аналогичен с разницей лишь в том, что линейный поиск будет выполняться в обратном порядке, начиная с последнего значения в заданном столбце.

Поиск с помощью оператора “AND”.

Пример запроса: SELECT \* FROM STUDENTS WHERE STUDENTS.GROUP\_NUMBER = 4303 AND STUDENTS.LAST\_NAME = “Сидоров”;

В этом случае необходимо выполнить следующую последовательность действий:

1. Найти с помощью бинарного поиска количество записей для каждого столбца и выбрать тот, который содержит наименьшее количество записей.
2. Для выбранного столбца реконструировать алгоритмом зигзага найденные записи, проверяю в процессе соответствие условия для второго столбца. Если условие не выполняется, то реконструкция записи останавливается.

Поиск с помощью оператора “OR”.

Пример запроса: SELECT \* FROM STUDENTS WHERE STUDENTS.GROUP\_NUMBER = 4303 OR STUDENTS.LAST\_NAME = “Сидоров”;

В этом случае необходимо выполнить следующую последовательность действий:

1. Найти с помощью бинарного поиска значение для первого столбца.
2. Реконструировать алгоритмом зигзага найденные записи.
3. Найти с помощью бинарного поиска значение для остальных столбцов.
4. Реконструировать записи, проверяю, что они еще не были реконструированы.

***Проекция***

Пример запроса: SELECT STUDENTS.GROUP\_NUMBER, STUDENTS.LAST\_NAME FROM STUDENTS;

Для вычисления проекции выполняется следующее:

Для каждой строки выполняется процесс реконструкции, но чтение из таблицы значений полей выполняется только для заданных столбцов, остальные игнорируются.

***Декартово произведение***

Пример запроса: SELECT STUDENTS.\*, GRADE.\* FROM STUDENTS, RESULTS;

Для реализации Декартова произведения необходимо выполнить следующие действия:

1. Реконструировать все записи первой таблицы.
2. Реконструировать все записи второй таблицы.
3. Для каждой строки первой таблицы выполнить шаг 4.
4. Сцепить строку с каждой строкой из второй таблицы.

Построение Декартово произведения в модели TransRelational требует больше операций чем в реляционной модели и не является оптимальной операцией.

***Объединение***

Реляционные операторы объединения, пересечения и разности требуют, чтобы два входных отношения имели одинаковые атрибуты.

Пример запроса: SELECT STUDENTS.GROUP\_NUMBER FROM STUDENTS UNION SELECT STUDENTS\_1.GROUP\_NUMBER FROM STUDENTS\_1;

Для реализации объединения необходимо выполнить следующие действия:

1. Реконструировать алгоритмом зигзага все строки из первой таблицы.
2. Реконструировать алгоритмом зигзага все строки из второй таблицы, проверяя что они еще не были реконструированы.

В случае, если для формирования таблицы значений полей использовались методы «сжатые столбцы» и «слившиеся столбцы» необходимо будет только реконструировать все записи.

Если необходимо выполнить объединение и выбрать только один столбец, то реконструкцию записей можно не выполнять.

***Пересечение***

Пример запроса: SELECT STUDENTS.GROUP\_NUMBER FROM STUDENTS INTERSECT SELECT STUDENTS\_1.GROUP\_NUMBER FROM STUDENTS\_1;

Для реализации пересечения необходимо выполнить следующее:

1. Реконструировать все записи из первой таблицы.
2. Реконструировать все записи из второй таблицы, проверяя, были ли они уже реконструированы. Если да, то добавлять их в результат.

В случае, если для формирования таблицы значений полей использовались методы «сжатые столбцы» и «слившиеся столбцы» необходимо будет реконструировать записи, которые имеют непустой диапазон для двух таблиц одновременно.

Можно сделать вывод, что построение пересечения может быть неоптимальной операцией, в случае если к таблице значений полей не применены методы «сжатые столбцы» и «слившиеся столбцы», и когда нужно выбрать данные из нескольких столбцов.

***Разность***

Пример запроса: SELECT STUDENTS.GROUP\_NUMBER FROM STUDENTS EXCEPT SELECT STUDENTS\_1.GROUP\_NUMBER FROM STUDENTS\_1;

Для реализации разности необходимо выполнить следующее:

1. Реконструировать все записи таблицы, из которой нужно сделать вычитание.
2. Реконструировать все записи из второй таблицы, проверяя, были ли они уже реконструированы. Если да, то удалять их из результата.

Можно сделать вывод, что операция разности является неоптимальной, в случае если к таблице значений полей не применены методы «сжатые столбцы» и «слившиеся столбцы», и когда нужно выбрать данные из нескольких столбцов.

***Соединение***

В модели TR соединение выполняется методом сортировки-слияния (Merge Join), т.к. данные в таблице значений полей отсортированы ещё в моменте формирования таблицы. Из этого следует преимущество модели TR в том, что чем больше отношений необходимо соединить, тем больше будет выигрыш в скорости, если сравнивать с системами с непосредственным отображением данных.

Пример запроса: SELECT STUDENTS.NUMBER, STUDENTS.LAST\_NAME, RESULTS.FINAL\_GRADE FROM STUDENTS, RESULTS WHERE STUDENTS.NUMBER = GRADE.STUDENT\_NUMBER;

Для соединения необходимо совершить следующие действия:

1. Последующие шаги выполняются для каждой строки в первой таблице.
2. Для заданного столбца первой таблицы значений полей [i, j] находится значение в другой таблице [i1, j1].
3. Для каждой таблицы реконструируется найденная запись по алгоритму зигзага, начиная с ячеек [i, j] и [i1, j1].
4. Две реконструированные записи объединяются в одну.

**2.3. Реализация операций обновления данных**

В данном разделе рассматриваются реализация таких операторов языка SQL как INSERT, DELETE, UPDATE в модели TransRelational.

Так как таблица значений полей хранит отсортированные по возрастанию значения, то при вставке новой строки в конец таблицы необходимо будет перестроить таблицу значений полей, а также таблицу реконструкции записей. Этот способ не является оптимальным и ведет к проблемам с производительностью.

Существует один из возможных подходов к реализации вставки нового значения. В этом случае при вставке значения в таблицу значения полей для каждого столбца бинарным поиском находится позиция для нового значения, но при этом таблица реконструкции не перестраивается. Когда запись удаляется из пользовательского файла, физического удаления соответствующего элемента в таблицах значения полей и реконструкции записей не происходит. Эти ячейки помечаются как «удалены логически», их можно затем рассматривать как свободное место в каждой из двух таблиц. При последующей вставке новой записи эти ячейки будут использоваться, таким образом можно избежать полного перестроения таблицы реконструкции записей.

Для случая, когда необходимо вставить новую запись, но при этом ни одна из записей не была удалена до этого, в реализации TR модели при инициализации таблиц резервируются пустые ячейки для последующей вставки.

Перед тем как подробно рассматривать реализацию операции обновления, необходимо рассмотреть алгоритм, который позволяет управлять отмеченными ячейками.

***Алгоритм обмена (The swap algorithm)***

Основная идея реализации алгоритмов обновления данных в TR модели заключается в том, что ячейки таблиц значений полей и реконструкции могут быть повторно использованы. Т.е. при удалении ячейки помечаются как «свободные», в дальнейшем в них же будут вставляться новые значения.

Зачастую возникает необходимость поменять отмеченные ячейки местами. Алгоритмы обновления предназначены для работы таким образом, чтобы обеспечить постоянное перемещение объекта в настолько малой области, насколько это возможно, с общей целью сохранения как можно более статичных таблиц TR и, таким образом, минимизации накладных расходов.

Если в случае добавления или обновления элемента нет «отмеченной» ячейки необходимо выполнить следующую последовательность действия:

1. Пусть j – текущий номер столбца, fj – новое значение, i – номер строки.
2. Найти ближайшую «отмеченную» ячейку в столбце – [i1, j]. Если вставка в ячейку значения fj сохраняет сортировку, то перейти к шагу 3. Если нет, то выполнить шаг 4.
3. Вставить fj в отмеченную ячейку [i1, j], удалить маркер. Перестроить цепочку для записи в таблице реконструкции.
4. Выполнять шаг 5 до тех пор, пока положение «отмеченной» ячейки не будет подходить для вставки элемента, т.е. будет сохранять сортировку.
5. Если i1 > i, то необходимо найти ближайшее меньшее значение к i1. Если i1 < i, то необходимо найти ближайшее большее значение к i1. Если найденное значение ячейки [i2, j] является уникальным в столбце, то она помечается как «свободное», а для ячейки [i1, j] удаляется маркет и записывается значение ячейки [i2, j]. Перестраиваются цепочки для записей в таблице реконструкции. Если найденное значение ячейки [i2, j] не является уникальным, то в этом случае для обмена выбирается ячейка с таким же значением, но ближняя к ячейке [i, j].

***Вставка нового значения***

На основании алгоритма обмена можно описать реализацию операции INSERT следующим образом:

1. Пусть r – запись для вставки, а f1, f2, …, fn – поля внутри r.
2. Для каждого j (j = 1, 2, …, n) выполнить шаг 3.
3. Найти в столбце j «отмеченную» ячейку, при вставке в которую сохранится сортировка. Если такая ячейка есть, то элемент вставляется и осуществляется переход к следующему столбцу. Если нет, то выполняется шаг 4.
4. С помощью алгоритма обмена подбирается нужная ячейка для вставки. Вставить элемент и перейти к следующему столбцу.

***Удаление значения***

Как описано выше удаления записей осуществляется методом пометки ячеек как свободных, без физического удаления.

Таким образом для реализации DELETE необходимо выполнить следующую последовательность действия:

1. С помощью бинарного поиска для заданного столбца в таблице значений полей определяется стартовая ячейка.
2. Алгоритмом зигзага происходит обход и пометка ячеек как свободных в таблицах значений полей и реконструкции записей.

***Обновление записи***

Для реализации UPDATE необходимо выполнить следующую последовательность действия:

1. Пусть r – запись, а f1, f2, …, fn – поля внутри r, j- номер столбца.
2. По заданному условию определяется с помощью бинарного поиска начальная позиция [i, j].
3. Для каждого j выполнить оставшиеся шаги.
4. Перейти в ячейку [i, j] таблицы реконструкции записей и прочитать номер строки r. Прочитать в таблице значений полей в ячейке [r, j + 1] значение. Если полученное значение совпадает с fj, то перейти к следующему столбцу. Если ячейка была отмечена, то необходимо удалить маркер. Если fj не совпадает с полученным значением, то необходимо fj сравнить с ячейками таблицы значения полей [r - 1, j +1] и [r + 1, j +1]. Если [r - 1, j +1] <= fj <= [r + 1, j +1], перейти к шагу 5. Если нет, то к шагу 6.
5. Заменить значение ячейки [r, j + 1] таблицы значений полей на fj и перейти к следующему столбцу. Если ячейка была отмечена, то необходимо удалить маркер.
6. Для вставки нового значения необходимо выполнить алгоритм обмена, который был описан выше.

Стоит отметить, что операция UPDATE может быть логически представлена как операция DELETE, за которой следует INSERT.

**2.4. Сценарии использования**

Учитывая функциональные требования к программе, была разработана диаграмма сценариев использования, представленная рис.2.2.

Данная диаграмма описывает следующие варианты использования: ***Импорт данных в формате SQL:***

* Пользователь вводит команду для импорта «import file\_name;», где file\_name - путь к файлу формата SQL.
* Пользователь видит созданные и выведенные на экране таблицы TR, а именно: таблицу значения полей и таблицу реконструкции.

***Просмотр таблицы значений полей:***

* Пользователь вводит команду «show FVT table\_name;», где table\_name – название таблицы значений полей.
* Пользователь видит на экране выведенную таблицу значений полей.

***Просмотр таблицы реконструкции записей:***

* Пользователь вводит команду «show RRT table\_name;», где table\_name – название таблицы реконструкции записей.
* Пользователь видит на экране выведенную таблицу реконструкции записей.

***Операция INSERT:***

* Пользователь вводит команду «INSERT INTO table\_name VALUES (value1, value2, value3, ...);», где table\_name – название таблицы.
* Пользователь получает сообщение на экране об успешной записи.

***Операция DELETE:***

* Пользователь вводит команду «DELETE FROM table\_name WHERE condition;», где table\_name – название таблицы, а condition – условие с оператором «=».
* Пользователь получает сообщение на экране об успешном удалении записи.

***Операция UPDATE:***

* Пользователь вводит команду «UPDATE table\_name SET column1 = value1, column2 = value2, ...WHERE condition;», где table\_name – название таблицы, а condition – условие с оператором «=».
* Пользователь получает сообщение на экране об успешном изменении записи.

***Выборка данных с заданными столбцами:***

* Пользователь вводит команду «SELECT column1, column2 … FROM table\_name;», где table\_name – название таблицы, а column1, column2 – заданные столбцы для выборки.
* Пользователь видит на экране результат выборки в виде таблицы.

***Выборка данных с помощью оператора WHERE:***

* Пользователь вводит команду «SELECT \* FROM table\_name WHERE condition;», где table\_name – название таблицы, а condition – условие с операторами: «=», «<=», «>=», «<», «>».
* Пользователь видит на экране результат выборки в виде таблицы.

***Выборка данных с помощью оператора INNER JOIN:***

* Пользователь вводит команду «SELECT \* FROM table\_name1 INNER JOIN table\_name2 ON table1.column\_name = table2.column\_name;», где table\_name1 и table\_name2– названия таблиц, а column\_name – название столбца.
* Пользователь видит на экране результат выборки в виде таблицы.
* ***Со сценариями использования не поняла. Почему выборка данных - это селекция, проекция и соединение? Если рассматриваются операции реляционной алгебры, то части из них не хватает. Вообще-то выборка и селекция - синонимы. Почему селекция делится на 3 варианта? Если выше вы описываете сценарии вариантов использования, то почему есть только 1 вариант для селекции?***

Хотела сделать акцент на поддерживаемых операторах для условий. Например, не реализовывала возможность использования операторов: BETWEEN, LIKE, IN, но поддерживаются «=», «<=», «>=», «<», «>». И хотела выделить отдельно подобный синтаксис «SELECT column1, column2, ...

FROM table\_name;»

***По поводу консоли - это уже реализация, при проектировании интерфейса важны только операции пользователя и системы, а не способ их осуществления.*** *Заменила на «*на экране»

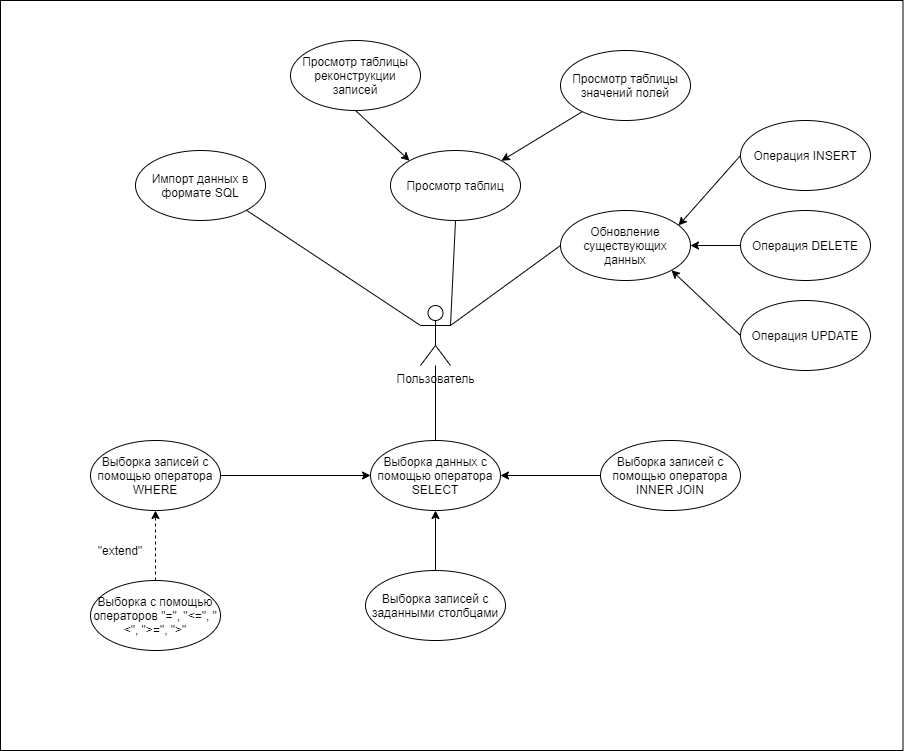


Рисунок 2.2 – Сценарии использования

**2.5. Архитектура решения**

Так как основные задачи данной работы заключаются в реализации алгоритмов преобразования реляционной модели в модель TransRelational, реляционных операторов и доступа к данным, было принято решение разработать консольное приложение, которое обрабатывает определенные команды пользователя. По той же причине решено хранить данные в оперативной памяти, без записи на диск. Хранение данных на диске является самостоятельной задачей, выходящей за рамки данной работы.

На рисунке 2.3 представлена спроектированная архитектура.

Рассмотрим более подробно слои и компоненты предложенной архитектуры.

***Слой клиента***

Данный уровень отвечает за взаимодействия пользователя с текстовым интерфейсом. Главная функция интерфейса заключается в представлении задач и результатов, понятных пользователю. На этом уровне происходит обработка команд пользователя и распределение их по компонентам следующего уровня.

***Слой компонентов***

На данном уровне происходят операции по импортированию данных в модель TR, управлению созданными таблицами TR, выборке данных в соответствии с запросом пользователя, а также добавление, изменение и удаление данных.

Компонент вывода таблиц TR отвечает за получение таблиц значения полей и реконструкции записей из хранилища и отображение пользователю.

Компонент импорта отвечает за преобразование поступивших данных в формате SQL в модель TR. После чтения и анализа файла происходит формирование таблиц значений полей и реконструкции записей, сохранение таблиц в хранилище данных и вывод созданных таблиц пользователю.

Компонент выборки данных отвечает за обработку оператора запроса SELECT в языке SQL. В данном компоненте анализируется команда и в соответствии с заданной командой происходит чтение из хранилища данных.

Компонент изменения данных отвечает за обработку следующих операторов SQL: INSERT, DELETE и UPDATE. Т.е. в этом компоненте происходит добавление, удаление или изменение записи. В соответствии с запросом происходит обновление в хранилище данных.

***Слой данных***

Данный уровень представляет из себя структуру данных, которая хранит таблицы TR: значений полей и реконструкции записей. В этих таблицах содержатся данные пользователя. Архитектура слоя данных может быть значительно расширена при выборе реализации с записью данных на диск.

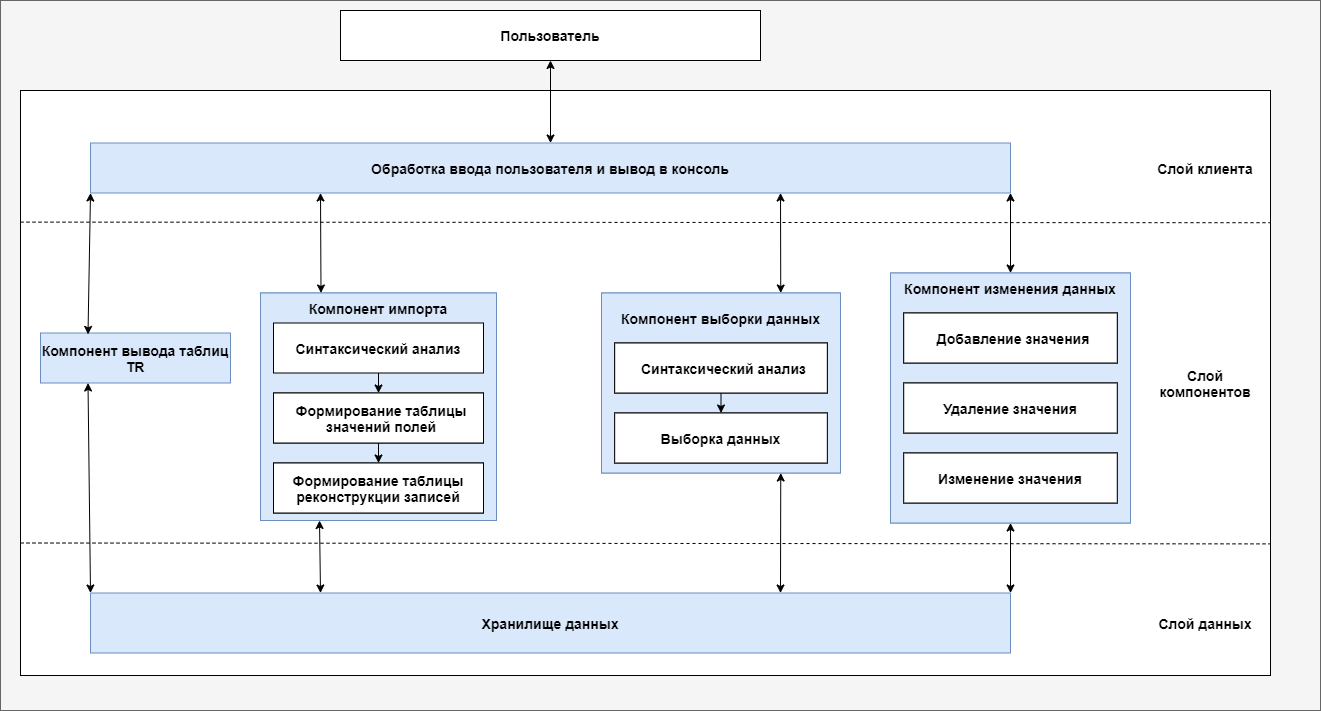


Рисунок 2.3 – Архитектура приложения

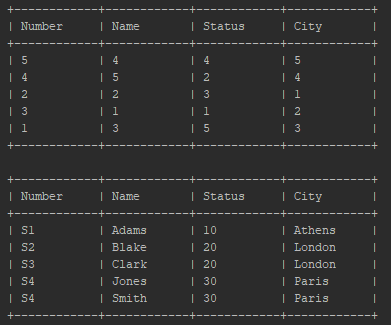
**2.6 Выводы**

В результате проектирования приложения были пошагово описаны алгоритмы преобразования в TR модель, реализации реляционных операторов, операций изменения данных. Выявлено, что следующие реляционные операторы: селекция, проекция, соединение являются наиболее оптимизированными в TR модели и показывают её преимущества. Также выявлено, что операции обновления и удаления значений могут быть эффективно реализовано в TR модели.

Разработана основные сценарии использования, архитектура приложения.

**3. Результаты разработки**

**3.1. Пользовательский интерфейс (надо ли если консольное приложение?)**



**3.2. Структура программной реализации.**

При разработке приложения предлагается использовать язык программирования Java, который хорошо приспособлен для работы с консолью, файлами, а также содержит встроенную реализацию некоторых алгоритмов поиска, например, реализацию бинарного поиска, который используется при выборке данных в TR модели.

**3.2. Результаты эксперимента (Performance)**

**3.3. Тестирование**

**4. дополнительный раздел**

**4.1. Первый подраздел дополнительного раздела**

**4.2. Второй подраздел дополнительного раздела**

**4.3. Третий подраздел дополнительного раздела**

Содержание дополнительного раздела определяется:

- «Методическими указаниями», разработанными кафедрой, отвечающей за содержание дополнительного раздела;

- указаниями консультанта, назначаемого студенту на начальном этапе выполнения выпускной квалификационной работы.

**заключение**

Кратко (на одну-две страницы) описать основные результаты работы, проанализировать их соответствие поставленной цели работы, показать рекомендации по конкретному использованию результатов исследования и перспективы дальнейшего развития работы.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. U.S. Patent and Trademark Office: Value-lnstance-Connectivity Computer-

Implemented Database. U.S. Patent No. 6,009,432. - December 28, 1999.

2. Дейт, К. Дж. Введение в системы баз данных, 8-е издание/ М.: Издательский дом "Вильяме", 2005. — 1328 с.

3. Date С. J. Go Faster! The TransRelational™ Approach to DBMS Implementation

4. Фомичева Т.Г. Базы данных. Проектирование приложений реляционной БД: конспект лекций. Ч.1 / Т.Г.Фомичева. – СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2008. 82 с.

5. Фомичева Т.Г. Методические указания к лабораторным работам по дисциплине «Базы данных»: учеб.-метод. пособие / сост.: Т.Г. Фомичева: СПб. 2016. 123 с.

6. Уровни представления данных – [Электронный ресурс]. URL: <http://economyreview.ru/teoriya-subd/urovni-modeley-dannyh> (Дата обращения: 10.12.2019)

<https://aws.amazon.com/ru/nosql/>

<https://habr.com/ru/company/mailru/blog/266811/>

<https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/>

<https://habr.com/ru/post/278087/>

<https://sites.google.com/site/raznyeurokipoinformatiki/home/bazy-dannyh/teoria-po-bazam-dannyh/varianty-hranenia-dannyh/osnovnye-modeli-baz-dannyh>