

ВВЕДЕНИЕ

Цель дипломного проекта – разработать лабораторный практикум для изучения высоконагруженных реляционных баз данных.

В век информационных технологий актуальным становится использование электронных учебников, пособий, практикумов по дисциплинам, так как их легко копировать и тиражировать.

Электронный учебник - совокупность веб-страниц, связанных воедино в один комплекс, в которых отражено основное научное содержание учебной дисциплины.

Электронный учебник удобно использовать при проведении дисциплин, так как материал легко усваивается, доступен для понимания, а также каждый студент может работать со своей скоростью.

Лабораторный практикум может быть использован в учебном процессе при проведении лабораторно-практических занятий, для самостоятельной работы студентов при подготовке к занятиям, а также в качестве дополнительного материала.

Рассмотрим, что из себя представляет СУБД.

СУБД (система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД – комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

В данном лабораторном практикуме используется свободная объектно-реляционная система управления базами данных – PostgreSQL и MySQL.

В данном лабораторном практикуме будут рассмотрены основные моменты для изучения СУБД: третья нормальная форма, триггеры, индексация, права доступа, транзакции, установка и первичная настройка кластера Percona XtraDB.

1 КОНЦЕПЦИЯ ПРОГРАММНОЙ СИСТЕМЫ

1.1 Обзор PostgreSQL и MySQL

По мере роста популярности каждое приложение рано или поздно сталкивается с проблемами производительности своих подсистем. Эти проблемы имеют разные причины и могут быть связаны как с архитектурой самого приложения, так и с внешними ограничениями. Одной из наиболее часто встречающихся проблем оказывается решение вопросов производительности используемых баз данных. Особенно это актуально для приложений: в ряде случаев повышение производительности при масштабировании проекта может быть достигнуто за счет применения NoSQL-решений; однако во многих ситуациях отказ от традиционной реляционной СУБД нежелателен по причинам, связанным с теми же самыми архитектурой и/или внешними ограничениями.

В качестве основы для практикума были выбраны две свободные системы управления базами данных: PostgreSQL и MySQL.

Способность обеих СУБД справляться со значительными объемами данных наглядно иллюстрирует приведенная таблица. Заметим, что если PostgreSQL изначально разрабатывалась с прицелом на высокую масштабируемость и нагрузки в диапазоне от однопользовательских приложений до крупномасштабных сетевых серверов, то движение MySQL в сторону средних и крупных приложений носило эволюционный характер. Сравнение двух СУБД представлено в таблице 1.1.

Таблица 1.1 – Сравнение PostgreSQL и MySQL

Характеристики	PostgreSQL	MySQL
Макс. размер БД	нет ограничения	нет ограничения
Макс. размер таблицы	32 ТБайт	256 ТБайт (MyISAM storage limits)
Максимальный размер записи (строки в таблице)	1.6 ТБайт	64 КБайт

Продолжение таблицы 1.1

Характеристики	PostgreSQL	MySQL
Максимальный размер поля в записи	1 ГБайт (text, bytea) или 4 ТБайт (в pg_largeobject)	4 ГБайт (longtext, longblob)
Максимальное количество полей (колонок) в таблице	250 — 1600 в зависимости от типа данных в колонке	4096

Сильными сторонами PostgreSQL считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликаций;
 - расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, а также имеется поддержка загрузки C-совместимых модулей;
 - наследование;
 - лёгкая расширяемость.
- PostgreSQL поддерживает большой набор встроенных типов данных:
- численные типы;
 - целые;
 - с фиксированной точкой;
 - с плавающей точкой;
 - денежный тип (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой с двумя знаками после запятой);
 - символьные типы произвольной длины;
 - двоичные типы (включая BLOB);
 - типы «дата/время» (полностью поддерживающие различные форматы, точность, форматы вывода, включая последние изменения в часовых поясах).
 - булев тип;
 - перечисление;
 - геометрические примитивы;

- Сетевые типы.
- IP и IPv6-адреса.
- CIDR-формат.
- MAC-адрес.
- UUID-идентификатор.
- XML-данные.
- Массивы.
- JSON.
- Идентификаторы объектов БД.
- Псевдотипы.

Более того, пользователь может самостоятельно создавать новые требуемые ему типы и программировать для них механизмы индексирования с помощью GiST(Generalized Search Tree)

GiST представляет собой сбалансированное (по высоте) дерево, концевые узлы (листья) которого содержат пары (key, rid), где key – ключ, а rid – указатель на соответствующую запись на странице данных. Внутренние узлы содержат пары (p, ptr), где p – это некий предикат (используется как поисковый ключ), выполняющийся для всех наследных узлов, а ptr – указатель на другой узел в дереве. Для этого дерева определены базовые методы SEARCH, INSERT, DELETE, и интерфейс для написания пользовательских методов, которыми можно управлять работой этих (базовых) методов.

Метод SEARCH управляется функцией Consistent, возвращающая «true», если узел удовлетворяет предикату, метод INSERT – функциями penalty, picksplit и union, которые позволяют оценить сложность операции вставки в узел, разделить узел при переполнении и перестроить дерево при необходимости, метод DELETE находит лист дерева, содержащий ключ, удаляет пару (key, rid) и, если нужно, с помощью функции union перестраивает родительские узлы.

GiST является прямым индексом, используемым полнотекстовым поиском СУБД PostgreSQL. Это означает, что для каждого вектора tsvector, описывающего все лексемы документа, создаётся сигнатура, описывающая, какие из лексем входят в данный tsvector [1].

PostgreSQL может быть расширен пользователем для собственных нужд практически в любом аспекте. Есть возможность добавлять собственные:

- Преобразования типов.
- Типы данных.

– домены (пользовательские типы с изначально наложенными ограничениями);

– функции (включая агрегатные);

– индексы;

– операторы (включая переопределение уже существующих);

– процедурные языки;

Прочие возможности:

– соблюдение принципов ACID;

– соответствие стандартам ANSI SQL-92, SQL-99, SQL:2003, SQL:2011;

– Поддержка запросов с OUTER JOIN, UNION, UNION ALL, EXCEPT, INTERSECT и подзапросов;

– Последовательности;

– Контроль целостности;

– Репликация;

– общие табличные выражения и рекурсивные запросы;

– аналитические функции;

– поддержка Юникода (UTF-8);

– поддержка регулярных выражений в стиле Perl;

– встроенная поддержка SSL, SELinux и Kerberos;

– протокол разделяемых блокировок;

– подгружаемые расширения, поддерживающие SHA1, MD5, XML;

– расширения для написания сложных выборок, отчётов и т. д. (API открыт);

– средства для генерации совместимого с другими системами SQL-кода и импорта из других систем;

– автономные блоки на доступных языках, а не только SQL [1];

Рассмотрим особенности базы данных MySQL.

MySQL - это система управления реляционными базами данных.

В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

Внутренние характеристики и переносимость:

- Написан на C и C++. Протестирован на множестве различных компиляторов.
- Работает на различных платформах.
- Полностью многопоточный с использованием потоков ядра.
- Хеш-таблицы в памяти, используемые как временные таблицы.
- SQL-функции реализованы при помощи хорошо оптимизированной библиотеки классов, поэтому они выполняются настолько быстро, насколько это возможно.

- Все столбцы имеют значения по умолчанию. С помощью INSERT можно вставить подмножество столбцов таблицы; столбцы, для которых явно не заданы значения, устанавливаются в значения по умолчанию.

- Полная поддержка операторов и функций в SELECT- и WHERE- частях запросов.

- Полная поддержка для операторов SQL GROUP BY и ORDER BY с выражениями SQL.

- Управляет очень большими базами данных. Компания MySQL использует MySQL для работы с несколькими базами данных, которые содержат 50 миллионов записей.

- Для каждой таблицы разрешается иметь до 32 индексов. Каждый индекс может содержать от 1 до 16 столбцов или частей столбцов. Максимальная ширина индекса 500 бит (это значение может быть изменено при компиляции MySQL). Для индекса может использоваться префикс поля CHAR или VARCHAR.

Результатом является различия в компонентный подходе обеих СУБД: в случае PostgreSQL он нацелен на создание модулей расширения, в то время как к числу особенностей MySQL относится гибкость, достигаемая выбором подсистем хранения (database engines) под конкретные задачи (вплоть до узкоспециализированных и экзотических вариантов). Аналогичная ситуация наблюдается с расширением стандартного языка SQL и реляционной модели: в случае PostgreSQL это заметная концептуальная особенность проекта, в то время как в MySQL элементы, noSQL постепенно появляются также в виде дополнений (подсистема хранения myRocks и др.), сродни постепенному появлению поддержки ряда расширенных возможностей SQL в данной СУБД [2].

1.2 Разработка концепции системы

Концепция системы – это описание системы в первом приближении.

Основной целью дипломного проекта ставится создание методических пособий по важным аспектам СУБД.

Методическое пособие – это издание, предназначенное в помощь педагогам для практического применения на практике, в котором основной упор делается на методику преподавания. В основе любого пособия лежат конкретные примеры и рекомендации. Методическое пособие отличается от методических рекомендаций тем, что содержит, наряду с практическими рекомендациями, еще и теоретические положения, раскрывающие существующие точки зрения на излагаемый вопрос в педагогической науке. В методических рекомендациях теория вопроса дается минимально.

Методические пособия в данном дипломном проекте полностью раскрывают тему и дают необходимые теоретические и практические знания по работе с СУБД.

Каждое методическое пособие структурировано для последовательного обучения. Основные разделы методических пособий:

- цель работы;
- содержание;
- теоретические сведения;
- вывод;
- контрольные вопросы;
- задания для выполнения.

Цели работ методических пособий:

- разработка структуры БД в третьей нормальной форме в соответствии с заданной предметной областью;
- реализация запросов и представлений по заданным параметрам;
- проверка скорости работы запросов и их оптимизация с помощью внешней составной искусственной индексации;
- использование триггеров и создание пользовательских процедур;
- разделение прав доступа и проведения транзакций в соответствии с уровнем пользователей;
- изучение пост реляционных особенностей на примере наследования таблиц и многомерных полей;
- построение и первичная настройка кластера.

При чётком и последовательном следовании и выполнении всех пунктов данного лабораторного практикума у учащегося будут сформированы основные навыки с работой в системах управления реляционных баз данных.

Так же будут получены знания и умения разработки структуры баз данных в третьей нормальной форме в первой лабораторной работе данного практикума.

Во второй лабораторной работе будут получены практические умения для реализации запросов и представлений по заданным параметрам на языке запросов SQL.

Проверка скорости работы запросов и их оптимизация являются необходимыми знаниями при работе с большими объёмами информации в базе данных.

Разделение прав доступа необходимо для недопущения изменения или удаления данных в таблице, к которым у данного пользователя нет доступа.

В отличие от плоских, реляционные базы данных состоят из нескольких таблиц, связь между которыми устанавливается с помощью совпадающих значений одноименных полей.

Здесь следует отметить, что использование реляционной модели баз данных не является единственно возможным способом представления информации. В настоящее время существует несколько различных моделей представления данных, которые, однако, пока не получили такого широкого распространения среди разработчиков и пользователей, как реляционная модель. При разработке систем управления базами данных реляционная модель практически является стандартом.

Знания и умения, которые будут получены в последней лабораторной работе практикума пригодятся в дальнейшей рабочей деятельности по специальности в различных отраслях.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 ТЕХНОЛОГИИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

В данном лабораторном практикуме используется язык запросов к базам данных – SQL.

Данный язык используется практически во всех лабораторных работах в рамках данного лабораторного практикума, кроме той, где предусмотрено создание и первичная настройка кластера.

Язык структурированных запросов SQL – это декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Является прежде всего информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL считается языком программирования, в общем случае (без ряда современных расширений) не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений. Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

- создание в базе данных новой таблицы;
- добавление в таблицу новых записей;
- изменение записей;
- выборка записей из одной или нескольких таблиц;
- изменение структур таблиц.

Со временем SQL усложнился – обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Рассмотрим стандартизацию языка SQL.

В 1983 году Международная организация по стандартизации (ISO) и Американский национальный институт стандартов (ANSI) приступили к разработке стандарта языка SQL. По прошествии множества консультаций и отклонения нескольких предварительных вариантов, в 1986 году ANSI представил свою первую версию стандарта, описанную в документе ANSI X3.135-1986 под названием «Database Language SQL» (Язык баз данных SQL). Неофициально этот стандарт SQL-86 получил название SQL1.

Новая версия стандарта была принята в 1992 году, заменив стандарт SQL89. Новый стандарт, озаглавленный как SQL92, представлял собой по сути расширение стандарта SQL1, включив в себя множество дополнений, имевшихся в предыдущих версиях инструкций.

После принятия стандарта SQL92 к нему были добавлены ещё несколько документов, расширявших функциональность языка. Так, в 1995 году был принят стандарт SQL/CLI (Call Level Interface, интерфейс уровня вызовов), впоследствии переименованный в CLI95. На следующий год был принят стандарт SQL/PSM (Persistent Stored Modules, постоянно хранимые модули), получивший название PSM-96.

Следующим стандартом стал SQL:1999 (SQL3). В настоящее время действует стандарт, принятый в 2003 году (SQL:2003) с небольшими модификациями, внесёнными позже (SQL:2008).

Для дипломного проекта была выбрана свободная СУБД PostgreSQL – это свободная объектно-реляционная система управления базами данных (СУБД).

Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD- системы, HP- UX, IRIX, Linux, macOS, Solaris/Open Solaris, Tru64, QNX, а также для Microsoft Windows.

PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году, имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres, на тот момент уже приобретённого компанией Computer Associates. Название расшифровывалось как «Post Ingres», и при создании Postgres были применены многие уже ранее сделанные наработки.

Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. Новая СУБД получила имя, под которым она известна и развивается в текущий момент – PostgreSQL.

						БрГТУ.150089 – 06 81 00	Лист 14
Изм	Код	Лист	№ док	Подпись	Дата		

В данном методическом практикуме рекомендуется использовать Postgres Pro в качестве основной СУБД [3].

Postgres Pro – это российская коммерческая СУБД, разработанная компанией Postgres Professional с использованием свободно-распространяемой СУБД PostgreSQL, значительно переработанная для соответствия требованиям корпоративных заказчиков.

PostgreSQL поддерживает одновременную модификацию БД несколькими пользователями с помощью механизма Multiversion Concurrency Control (MVCC). Благодаря этому соблюдаются требования ACID и практически отпадает нужда в блокировках чтения.

Каждая версия Postgres Pro Standard содержит все функциональные возможности PostgreSQL с дополнительными патчами ядра, которые скоро будут приняты сообществом, а также расширениями и патчами, разработанными Postgres Professional [4].

В качестве кластера для данного лабораторного практикума используется Percona XtraDB Cluster.

Percona XtraDB Cluster – это полностью доступное решение с высокой доступностью для MySQL. Он интегрирует Percona Server и Percona XtraBackup с библиотекой Galera для обеспечения синхронной репликации с несколькими мастерами.

Рассмотрим преимущества Percona XtraDB Cluster:

- Синхронная репликация. Транзакция проходит либо на всех нодах, либо ни на каком.
- Балансировщик нагрузки ProxySQL.
- Полная совместимость с базами данных MySQL.
- Режим multi-master. Писать данные можно в любой нод.

Кластер состоит из узлов, где каждый узел содержит одинаковый набор данных, синхронизированных по узлам. Рекомендуемая конфигурация должна иметь как минимум 3 узла, но вы также можете иметь 2 узла. Каждый узел является обычным экземпляром MySQL Server (например, Percona Server). Можно преобразовать существующий экземпляр MySQL Server в узел и запустить кластер, используя этот узел в качестве базы. Вы также можете отсоединить любой узел от кластера и использовать его как обычный экземпляр MySQL Server. [5]

3 РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ

3.1 ИСПОЛЬЗОВАНИЕ ЯЗЫКА ЗАПРОСОВ SQL

Данный лабораторный практикум создавался для получения необходимых знаний и умений при работе с СУБД.

Рассмотрим реализацию запросов для баз данных с использованием языка запросов SQL.

Запрос к базе данных – это средство выбора необходимой информации из базы данных.

Все запросы к базе данным начинаются с ключевых слов: `SELECT`, `WHERE`, `FROM`, `GROUP BY`, `ORDER BY`. Пример запроса показан на рисунке 3.1.

```
SELECT* FROM nametable WHERE условие_ограничения
```

Рисунок 3.1 – Простой запрос к БД

Где, `условие_ограничения` – это любое выражение значения, выдающее результат типа `boolean`, `nametable` – указывается имя таблицы с которой выбирать информацию.

После обработки списка выборки в результирующей таблице можно дополнительно исключить дублирующиеся строки. Для этого сразу после `SELECT` добавляется ключевое слово `DISTINCT`.

Пример запроса с ключевым словом `DISTINCT` показан на рисунке 3.2.

```
SELECT DISTINCT список_выборки
```

Рисунок 3.2 – запрос с ключевым словом `DISTINCT`

Две строки считаются разными, если они содержат различные значения минимум в одном столбце. При этом значения `NULL` полагаются равными.

Представление – это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления.

Для создания представления используется команда CREATE VIEW. Она состоит из слов CREATE VIEW (создать представление), имени представления которое нужно создать, слова AS (как). Пример показан на рисунке 3.3.

```
CREATE VIEW nameview
AS SELECT *
FROM Salespeople
WHERE city = 'London';
```

Рисунок 3.3 – Пример создания представления

Когда следует SQL выбрать (SELECT) все строки (*) из представления, он выполняет запрос содержащий в определении - Loncfonstaff, и возвращает все из его вывода.

Представления – превосходный способ дать публичный доступ к некоторой, но не всей информации в таблице.

Групповые представления – это представления, который содержит предложение GROUP BY, или который основывается на других групповых представлениях. Групповые представления могут стать превосходным способом обрабатывать полученную информацию непрерывно.

Предложение GROUP BY группирует строки таблицы, объединяя их в одну группу при совпадении значений во всех перечисленных столбцах. Порядок, в котором указаны столбцы, не имеет значения. В результате наборы строк с одинаковыми значениями преобразуются в отдельные строки, представляющие все строки группы. Это может быть полезно для устранения избыточности выходных данных и/или для вычисления агрегатных функций, применённых к этим группам.

INNER JOIN, FULL JOIN, LEFT JOIN, RIGHT JOIN – это наиболее часто используемое в языке запросов SQL соединение, оно возвращает пересечение двух множеств, которые отвечают указанному критерию.

Агрегатное выражение представляет собой применение агрегатной функции к строкам, выбранным запросом. Агрегатная функция сводит множество входных значений к одному выходному, как например, сумма или среднее. Агрегатное выражение показано на рисунке 3.4:

```
агрегатная_функция (выражение [ , ... ] [ предложение_order_by ] )
[ FILTER ( WHERE условие_фильтра ) ]
агрегатная_функция (ALL выражение [ , ... ] [ предложение_order_by ] )
[ FILTER ( WHERE условие_фильтра ) ]
```

Рисунок 3.4 – Агрегатное выражение

Здесь агрегатная функция — имя ранее определённой агрегатной функции (возможно, дополненное именем схемы), выражение — любое выражение значения, не содержащее в себе агрегатного выражения или вызова оконной функции.

PostgreSQL реализует наследование таблиц, что может быть полезно для проектировщиков баз данных.

Пример наследования показан на рисунке 3.5

```
CREATE TABLE cities (
  name          text,
  population     float,
  altitude       int    -- в футах
);

CREATE TABLE capitals (
  state          char(2)
) INHERITS (cities);
```

Рисунок 3.5 – Пример наследования

В этом примере таблица capitals наследует все столбцы своей родительской таблицы, cities. Столицы штатов также имеют дополнительный столбец state, в котором будет указан штат.

Когда в базе данных создаётся объект, ему назначается владелец. Владелец обычно становится роль, с которой был выполнен оператор создания. Для большинства типов объектов в исходном состоянии только владелец (или суперпользователь) может делать с объектом всё, что угодно. Чтобы разрешить использовать его другим ролям, нужно дать им права.

Существует несколько типов прав: SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE и USAGE. Набор прав, применимых к определённому объекту, зависит от типа объекта (таблица, функция и т. д.)

Неотъемлемое право изменять или удалять объект имеет только владелец объекта.

Для назначения прав применяется команда GRANT. Например, если в базе данных есть роль joe и таблица accounts, право на изменение таблицы можно дать этой роли так как показано на рисунке 3.6.

```
GRANT UPDATE ON accounts TO joe;
```

Рисунок 3.6 – Назначение прав доступа

Если вместо конкретного права написать ALL, роль получит все права, применимые для объекта этого типа.

Для назначения права всем ролям в системе можно использовать специальное имя «роли»: PUBLIC.

Чтобы лишить пользователей прав, используется команда REVOKE:

Пример лишения прав доступа показан на рисунке 3.7.

```
REVOKE ALL ON accounts FROM PUBLIC;
```

Рисунок 3.7 – Лишение прав доступа

Обычно распоряжаться правами может только владелец объекта (или суперпользователь). Однако возможно дать право доступа к объекту «с правом передачи», что позволит получившему такое право назначать его другим. Если такое право передачи впоследствии будет отозвано, то все, кто получил данное право доступа (непосредственно или по цепочке передачи), потеряют его.

Перечислим все возможные права:

- Ключевое слово `SELECT` - позволяет выполнять выборку для любого столбца или перечисленных столбцов в заданной таблице, представлении или последовательности. Также позволяет выполнять `COPY TO`. Помимо того, это право требуется для обращения к существующим значениям столбцов в `UPDATE` или `DELETE`.

- Ключевое слово `INSERT` позволяет вставлять строки в заданную таблицу. Если право ограничивается несколькими столбцами, только их значение можно будет задать в команде `INSERT` (другие столбцы получают значения по умолчанию). Также позволяет выполнять `COPY FROM`.

- Ключевое слово `UPDATE` позволяет изменять данные во всех, либо только перечисленных, столбцах в заданной таблице.

- Ключевое слово `DELETE` позволяет удалять строки из заданной таблицы.

- Ключевое слово `TRUNCATE` позволяет опустошить заданную таблицу.

- Ключевое слово `REFERENCES` позволяет создавать ограничение внешнего ключа, ссылающееся на определённую таблицу либо на определённые столбцы таблицы.

- Ключевое слово `TRIGGER` позволяет создавать триггеры в заданной таблице.

- Ключевое слово `CREATE` для баз данных это право позволяет создавать схемы и публикации в заданной базе.

- Ключевое слово `CONNECT` позволяет пользователю подключаться к указанной базе данных. Это право проверяется при установлении соединения.

- Ключевое слово `TEMP` позволяет создавать временные таблицы в заданной базе данных.

- Ключевое слово `EXECUTE` позволяет выполнять заданную функцию и применять любые определённые поверх неё операторы. Это единственный тип прав, применимый к функциям.

– Ключевое слово ALL PRIVILEGES даёт целевой роли все права сразу. Ключевое слово PRIVILEGES является необязательным в PostgreSQL, хотя в строгом SQL оно требуется.

Транзакции – фундаментальное понятие во всех СУБД. Суть транзакции в том, что она объединяет последовательность действий в одну операцию "всё или ничего". Промежуточные состояния внутри последовательности не видны другим транзакциям, и, если что-то помешает успешно завершить транзакцию, ни один из результатов этих действий не сохранится в базе данных.

Предположим, что надо перевести 100 долларов со счёта Алисы на счёт Боба. Пример данной реализации показан на рисунке 3.8.

```
UPDATE accounts SET balance = balance - 100.00
WHERE name = 'Alice';
UPDATE branches SET balance = balance - 100.00
WHERE name = (SELECT branch_name FROM accounts WHERE name = 'Alice');
UPDATE accounts SET balance = balance + 100.00
WHERE name = 'Bob';
UPDATE branches SET balance = balance + 100.00
WHERE name = (SELECT branch_name FROM accounts WHERE name = 'Bob');
```

Рисунок 3.8 – Пример реализации транзакции

Нам нужна гарантия, что, если что-то помешает выполнить операцию до конца, ни одно из действий не оставит следа в базе данных. И получаем эту гарантию, объединяя действия в одну транзакцию. Говорят, что транзакция атомарна: с точки зрения других транзакций она либо выполняется и фиксируется полностью, либо не фиксируется совсем.

Нам также нужна гарантия, что после завершения и подтверждения транзакции системой баз данных, её результаты в самом деле сохраняются и не будут потеряны, даже если вскоре произойдёт авария. Например, если сумма списана со счёта и выдана Бобу, следует исключить возможность того, что сумма на его счёте восстановится, как только он выйдет за двери банка. Транзакционная база данных гарантирует, что все изменения записываются в постоянное хранилище (например, на диск) до того, как транзакция будет считаться завершённой.

Другая важная характеристика транзакционных баз данных тесно связана с атомарностью изменений: когда одновременно выполняется множество транзакций, каждая из них не видит незавершённые изменения, произведённые другими. Например, если одна транзакция подсчитывает баланс по отделениям, будет неправильно, если она посчитает расход в отделении Алисы, но не учтёт приход в отделении Боба, или наоборот. Поэтому свойство транзакций "всё или ничего" должно определять не только, как изменения сохраняются в базе данных, но и как они видны в процессе работы. Изменения, производимые открытой транзакцией, невидимы для других транзакций, пока она не будет завершена, а затем они становятся видны все сразу.

В PostgreSQL транзакция определяется набором SQL-команд, окружённым командами BEGIN и COMMIT. Таким образом, наша банковская транзакция должна была бы выглядеть как показано на рисунке 3.9.

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
WHERE name = 'Alice';
-- ...
COMMIT;
```

Рисунок 3.9 – Определение транзакции в PostgreSQL

Если в процессе выполнения транзакции не обязательно фиксировать её изменения (например, потому что оказалось, что баланс Алисы стал отрицательным), следует выполнить команду ROLLBACK вместо COMMIT, и все внесённые изменения будут отменены.

PostgreSQL на самом деле обрабатывает каждый SQL-оператор как транзакцию. Если вы не вставите команду BEGIN, то каждый отдельный оператор будет неявно окружён командами BEGIN и COMMIT (в случае успешного завершения). Группу операторов, окружённых командами BEGIN и COMMIT иногда называют блоком транзакции.

Операторами в транзакции можно также управлять на более детальном уровне, используя точки сохранения. Точки сохранения позволяют выборочно отменять некоторые части транзакции и фиксировать все остальные. Определив точку сохранения с помощью SAVEPOINT, при необходимости вы можете вернуться к ней с помощью команды ROLLBACK TO. Все изменения в базе данных,

произошедшие после точки сохранения и до момента отката, отменяются, но изменения, произведённые ранее, сохраняются.

Всё это происходит в блоке транзакции, так что в других сеансах работы с базой данных этого не видно. Совершённые действия становятся видны для других сеансов все сразу, только когда вы фиксируете транзакцию, а отменённые действия не видны вообще никогда.

Вернувшись к банковской базе данных, предположим, что списываем 100 долларов со счёта Алисы, добавляем их на счёт Боба, и вдруг оказывается, что деньги нужно было перевести Уолли. В данном случае следует применять точки сохранения, как показано на рисунке 3.10.

```
BEGIN;  
UPDATE accounts SET balance = balance - 100.00  
WHERE name = 'Alice';  
SAVEPOINT my_savepoint;  
UPDATE accounts SET balance = balance + 100.00  
WHERE name = 'Bob';  
-- ошибочное действие... забыть его и использовать счёт Уолли  
ROLLBACK TO my_savepoint;  
UPDATE accounts SET balance = balance + 100.00  
WHERE name = 'Wally';  
COMMIT;
```

Рисунок 3.10 – Точка сохранения в транзакции

Ключевое слово ROLLBACK TO – это единственный способ вернуть контроль над блоком транзакций, оказавшимся в прерванном состоянии из-за ошибки системы, не считая возможности полностью отменить её и начать снова.

Пользовательские процедуры представляет собой объект базы данных, подобный функции. Отличие состоит в том, что процедура не возвращает значение, и поэтому для неё не определяется возвращаемый тип. Тогда как функция вызывается в составе запроса или команды DML, процедура вызывается явно, оператором CALL.

Синтаксис создания процедуры показан на рисунке 3.11.

```

CREATE [ OR REPLACE ] PROCEDURE
имя ( [ [ режим_аргумента ] [ имя_аргумента ] тип_аргумента
[ { DEFAULT | = } выражение_по_умолчанию ] [, ...] ] )
{ LANGUAGE имя_языка
| TRANSFORM { FOR TYPE имя_типа } [, ... ]
| [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
| SET параметр_конфигурации { TO значение | = значение | FROM CURRENT }
| AS 'определение'
| AS 'объектный_файл', 'объектный_символ'
}

```

Рисунок 3.11 – Пример создание процедуры

Команда CREATE PROCEDURE определяет новую процедуру. CREATE OR REPLACE PROCEDURE создаёт новую процедуру либо заменяет определение уже существующей. Чтобы определить процедуру, необходимо иметь право USAGE для соответствующего языка.

Команда CREATE OR REPLACE PROCEDURE предназначена для изменения текущего определения существующей процедуры. С её помощью нельзя изменить имя или типы аргументов (если попытаться сделать это, будет создана новая отдельная процедура).

Владельцем процедуры становится создавший её пользователь.

Чтобы создать процедуру, необходимо иметь право USAGE для типов её аргументов.

Параметры:

- Имя создаваемой процедуры.
- Режим аргумента: IN, INOUT или VARIADIC. По умолчанию подразумевается IN.
- Имя аргумента.
- Тип аргумента процедуры (возможно, дополненный схемой), при наличии аргументов. Тип аргументов может быть базовым, составным или доменным, либо это может быть ссылка на столбец таблицы.
- Выражение по умолчанию – это выражение, используемое для вычисления значения по умолчанию, если параметр не задан явно. Результат выражения должен сводиться к типу соответствующего параметра. Для всех входных параметров, следующих за параметром с определённым значением по умолчанию, также должны быть определены значения по умолчанию.

– Имя языка – это на котором реализована функция. Это может быть SQL, c, internal, либо имя процедурного языка, определённого пользователем, например, plpgsql.

– Определение – это строковая константа, определяющая реализацию процедуры; её значение зависит от языка. Это может быть имя внутренней процедуры, путь к объектному файлу, команда SQL или код на процедурном языке.

– Объектный файл, объектный символ – это форма предложения AS применяется для динамически загружаемых процедур на языке C, когда имя процедуры в коде C не совпадает с именем процедуры в SQL. Строка объектный файл задаёт имя файла, содержащего скомпилированную процедуру на C (данная команда воспринимает эту строку так же, как и LOAD).

Стандартный SQL используется в PostgreSQL и других реляционных БД как основной язык для создания запросов. Он переносим и прост, как для изучения, так и для использования. Однако слабое его место – в том, что каждая конструкция языка выполняется сервером отдельно. Это значит, что клиентское приложение должно отправлять каждый запрос серверу, получить его результат, определенным образом согласно логике приложения, обработать его, посылать следующий запрос и т. д. В случае, если клиент и сервер БД расположены на разных машинах, это может привести к нежелательному увеличению задержек и объема пересылаемых от клиента серверу и наоборот данных.

При использовании PL/pgSQL все становится проще. Появляется возможность сгруппировать запросы и вычислительные блоки в единую конструкцию, которая будет размещаться и выполняться на сервере, а клиент будет отправлять запрос на её выполнение и получать результат, минуя все промежуточные пересылки данных назад – вперед, что в большинстве случаев очень позитивно сказывается на производительности [6].

4 СЦЕНАРИЙ ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ С СИСТЕМОЙ

Postgres Pro реализован в архитектуре клиент-сервер. Рабочий сеанс Postgres Pro включает следующие взаимодействующие процессы:

- Главный серверный процесс, управляющий файлами баз данных, принимающий подключения клиентских приложений и выполняющий различные запросы клиентов к базам данных. Эта программа сервера БД называется postgres.

- Клиентское приложение пользователя, желающее выполнять операции в базе данных. Клиентские приложения могут быть очень разнообразными: это может быть текстовая утилита, графическое приложение, веб-сервер, использующий базу данных для отображения веб-страниц, или специализированный инструмент для обслуживания БД. Некоторые клиентские приложения поставляются в составе дистрибутива Postgres Pro, однако большинство создают сторонние разработчики.

Как и в других типичных клиент-серверных приложениях, клиент и сервер могут располагаться на разных компьютерах. В этом случае они взаимодействуют по сети TCP/IP. Важно не забывать это и понимать, что файлы, доступные на клиентском компьютере, могут быть недоступны (или доступны только под другим именем) на компьютере-сервере.

Сервер Postgres Pro может обслуживать одновременно несколько подключений клиентов. Для этого он запускает («порождает») отдельный процесс для каждого подключения. Можно сказать, что клиент и серверный процесс общаются, не затрагивая главный процесс postgres. Таким образом, главный серверный процесс всегда работает и ожидает подключения клиентов, принимая которые, он организует взаимодействие клиента и отдельного серверного процесса.

Так как подразумевается взаимодействие пользователя с оборудованием и средой, эта система называется «человек-техника-среда» (СЧТС).

Эта система включает в себя:

- человека (главное звено);
- технику (приборы, оборудование, компьютеры, средства связи), посредством которой человек осуществляет деятельность;
- среду, в которой осуществляется операционный процесс.

При этом все звенья системы рассматриваются в едином функциональном поле, где ведущее звено принадлежит человеку.

Поэтому система должна соответствовать эргономическим требованиям.

Эргономические требования – это требования, которые предъявляются к системе СЧТС в целях оптимизации деятельности человека с учётом его

социально-психологических, психофизических, психологических, антропометрических, физиологических и других объективных характеристик, и возможностей.

Одним из главных факторов успешного завершения обучающего процесса является удобство и простота использования системы – юзабилити.

Юзабилити – это способность продукта быть понимаемым, изучаемым, используемым и привлекательным для пользователя в заданных условиях; свойство системы, при наличии которого конкретный пользователь может эксплуатировать систему в определенных условиях для достижения установленных целей с необходимой результативностью, эффективностью и удовлетворённостью.

Удобство (пригодность) использования системы не сводится только к тому, насколько её легко эксплуатировать. эту характеристику следует понимать более широко, учитывая личные цели пользователя, его эмоции и ощущения, связанные с восприятием системы, а также удовлетворённость работой. Свойства, необходимые для обеспечения пригодности использования, зависят также от задачи и окружающей среды. Пригодность использования — не абсолютное понятие, оно может различным образом проявляться в определённых условиях эксплуатации.

Существует два основных способа оценки удобства использования продукта:

- прямая оценка на основе анализа результативности, эффективности и удовлетворённости, достигнутых в результате эксплуатации продукта в реальных условиях: если в указанных условиях одна система более эргономична, чем другая, то оценка должна это выявлять;

- косвенная оценка на основе анализа отдельных подхарактеристик, отражающих определённые свойства системы в установленных условиях эксплуатации;

Известный дизайнер Якоб Нильсен, предложил принципы проектирования взаимодействия:

- видимость статуса системы – пользователь должен всегда знать, что происходит, получая подходящую обратную связь в приемлемое время;

- соответствие между системой и реальным миром – система должна «говорить на языке пользователя», используя понятную ему терминологию и концепции, а не «системно-ориентированный» язык;

- управляемость и свобода для пользователя – пользователь часто выбирает системные функции по ошибке и должен иметь ясно видимый «аварийный выход» из не желаемого состояния системы, не требующий сложных диалогов. Следует поддерживать функции отмены и повтора;

– согласованность и стандарты – пользователи не должны гадать, значат ли одно и то же разные слова, ситуации или операции. Также нужно следовать соглашениям, принятым для данной платформы;

– предотвращение ошибок – продуманный дизайн, который не позволяет какой-то проблеме даже возникнуть, лучше, чем самые хорошие сообщения об ошибках. Следует устранять сами условия возникновения ошибок, либо выявлять их и предупреждать пользователя о предстоящей проблеме;

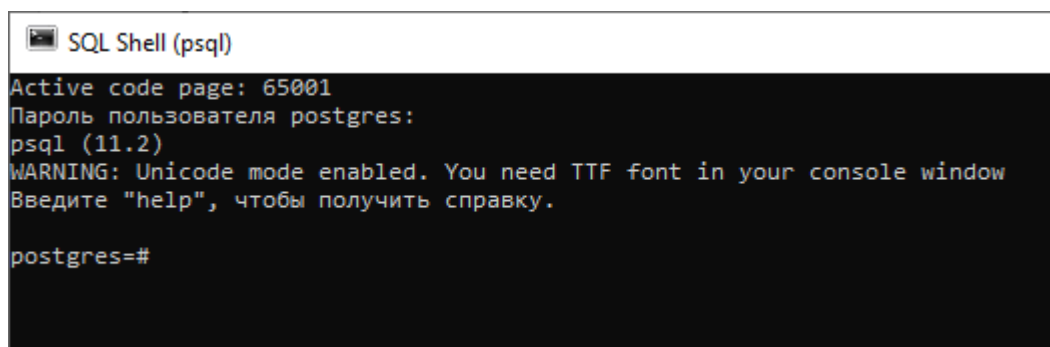
– распознавать лучше, чем вспоминать – минимизируйте нагрузку на память пользователя, явно показывая ему объекты, действия и варианты выбора. Пользователь не должен в одной части диалога запоминать информацию, которая потребуется ему в другой. Инструкции по использованию системы должны быть видимы или легко получаемы везде, где возможно;

– эстетический и минималистический дизайн – в интерфейсе не должно быть информации, которая не нужна пользователю или которая может понадобиться ему в редких случаях. Каждый избыточный элемент диалога отнимает внимание от нужных элементов;

– помочь пользователю понять и исправить ошибку – сообщения об ошибках следует писать простым языком, без кодов, чётко формулируя проблему и предлагая конструктивное решение;

– справка и документация – необходимо предоставлять справку и документацию. Информация должна быть простой в поиске, соответствовать задаче пользователя, описывать конкретную последовательность действий, и не должна быть слишком большой.

При запуске консоли Postgres Pro необходимо ввести пароль пользователя, после его ввода вы увидите версию и командную строку для ввода запросов, как показано на рисунке 4.1.



```
SQL Shell (psql)
Active code page: 65001
Пароль пользователя postgres:
psql (11.2)
WARNING: Unicode mode enabled. You need TTF font in your console window
Введите "help", чтобы получить справку.
postgres=#
```

Рисунок 4.1 – Вход в консоль

Создадим базу данных под названием work1, выполнив следующую команду, которая показана на рисунке 4.2.

```
postgres=# createdb work1
postgres=#
```

Рисунок 4.2 – Создание базы данных

Если нет никаких сообщений, значит операция была выполнена успешно и дополнительная проверка установки не требуется.

Далее подключимся к нашей базе данных как показано на рисунке 4.3.

```
postgres=# psql work1
postgres=#
```

Рисунок 4.3 – Подключение к базе данных

Заполним нашу базу данных создав таблицу погоды, для этого необходимо указать её имя, и перечислить все имена столбцов и их типы как показано на рисунке 4.4.

```
postgres=# CREATE TABLE weather (
postgres(#   city          varchar(80),
postgres(#   temp_lo      int,           -- минимальная температура дня
postgres(#   temp_hi      int,           -- максимальная температура дня
postgres(#   prcp         real,          -- уровень осадков
postgres(#   date         date
postgres(# );
```

Рисунок 4.4 – Создание таблицы с параметрами

Создадим ещё одну таблицу, где укажем название города и его географическое положение как показано на рисунке 4.5.

```
postgres=# CREATE TABLE cities (
postgres(#   name          varchar(80),
postgres(#   location      point
postgres(# );
```

Рисунок 4.5 – Создание таблицы с городами

Добавим записи в таблицу погоды как показано на рисунке 4.6.

```
postgres=# INSERT INTO weather VALUES ('San Francisco', 22, 40, 0.35, '2018-12-27');
INSERT 0 1
```

Рисунок 4.6 – Добавление записи в таблицу погоды

Выведем на экран таблицу погоды со всеми столбцами как показано на рисунке 4.7.

```
postgres=# SELECT * FROM weather;
  city      | temp_lo | temp_hi | prcp |    date
-----+-----+-----+-----+-----
San Francisco |      46 |      50 |  0.25 | 1994-11-27
San Francisco |      43 |      57 |    0 | 1994-11-29
San Francisco |      46 |      50 |  0.25 | 2018-11-27
San Francisco |      46 |      50 |  0.25 | 2018-11-27
San Francisco |      22 |      40 |  0.35 | 2018-12-27
(5 строк)
```

Рисунок 4.7 – Вывод таблицы со всеми столбцами

Сделаем выборку с условием: название города, средняя температура (минимальная и максимальная температура, делённая на 2) и дату. Результат выборки показан на рисунке 4.8.

```
postgres=# SELECT city, (temp_hi+temp_lo)/2 AS temp_avg, date FROM weather;
  city      | temp_avg |    date
-----+-----+-----
San Francisco |      48 | 1994-11-27
San Francisco |      50 | 1994-11-29
San Francisco |      48 | 2018-11-27
San Francisco |      48 | 2018-11-27
San Francisco |      31 | 2018-12-27
Hayward      |      45 | 1994-11-29
(6 строк)
```

Рисунок 4.8 – Результат выборки

Сделаем ещё одну выборку с условием, когда уровень осадков был выше нуля. Результат выборки показан на рисунке 4.9.

```

postgres=# SELECT * FROM weather
postgres=# WHERE city = 'San Francisco' AND prcp > 0.0;
  city      | temp_lo | temp_hi | prcp | date
-----+-----+-----+-----+-----
San Francisco | 46      | 50      | 0.25 | 1994-11-27
San Francisco | 46      | 50      | 0.25 | 2018-11-27
San Francisco | 46      | 50      | 0.25 | 2018-11-27
San Francisco | 22      | 40      | 0.35 | 2018-12-27
(4 строки)

```

Рисунок 4.9 – Результат выборки

Сделаем запрос и отсортируем его (по умолчанию сортируется в алфавитном порядке). Результат показан на рисунке 4.10.

```

postgres=# SELECT * FROM weather
postgres=# ORDER BY city;
  city      | temp_lo | temp_hi | prcp | date
-----+-----+-----+-----+-----
Hayward     | 37      | 54      |      | 1994-11-29
Hayward     | 37      | 54      |      | 1994-11-29
Hayward     | 37      | 54      |      | 1994-11-29
Hayward     | 37      | 54      |      | 1994-11-29
San Francisco | 46      | 50      | 0.25 | 1994-11-27
San Francisco | 22      | 40      | 0.35 | 2018-12-27
San Francisco | 43      | 57      | 0     | 1994-11-29
San Francisco | 46      | 50      | 0.25 | 2018-11-27
San Francisco | 46      | 50      | 0.25 | 2018-11-27
(9 строк)

```

Рисунок 4.10 – Отсортированный запрос

С помощью следующего запроса с использованием ключевого слово DISTINCT уберём повторяющиеся строки как показано на рисунке 4.11.

```

postgres=# SELECT DISTINCT city
postgres=# FROM weather
postgres=# ORDER BY city;
  city
-----
Hayward
San Francisco
(2 строки)

```

Рисунок 4.11 – Запрос без повторяющихся строк

Выберем самую высокую из всех минимальных дневных температур как показано на рисунке 4.12.

```
postgres=# SELECT max(temp_lo) FROM weather;
max
-----
 46
(1 строка)
```

Рисунок 4.12 – Выбор максимальной дневной температуры из таблицы

Выберем максимум минимальной дневной температуры в разрезе городов как показано на рисунке 4.13.

```
postgres=# SELECT city, max(temp_lo)
postgres-#      FROM weather
postgres-#      GROUP BY city;
 city      | max
-----+-----
 Hayward   |  37
 San Francisco |  46
(2 строки)
```

Рисунок 4.13 – Выборка в разрезе городов минимальной температуры

Если же, хотим удалить полностью всю таблицу, то для этого необходимо использовать ключевое слово DROP TABLE, как показано на рисунке 4.14.

```
postgres=# DROP table weather
postgres-#
```

Рисунок 4.14 – Пример удаления таблицы

Язык запросов к базам данных SQL, позволяет пользователю выполнять различные действия с базой данных следуя требованиям в данном методическом пособии [7].

5 ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

5.1 ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СИСТЕМЫ ПРИ ПОМОЩИ ВЕРТИКАЛЬНОГО МАСШТАБИРОВАНИЯ

Тестирование производительности – это комплекс типов тестирования, целью которого является определение работоспособности, стабильности, потребления ресурсов и других атрибутов качества приложения в условиях различных сценариев использования и нагрузок. Тестирование производительности позволяет находить возможные уязвимости и недостатки в системе с целью предотвратить их пагубное влияние на работу программы в условиях использования. Необходимые параметры работы системы в определенной среде можно тестировать с помощью:

- определения рабочего количества пользователей приложения;
- измерение времени выполнения различных операций системы;
- определения производительности приложения при различных степенях нагрузки;
- определения допустимых границ производительности программы при разных уровнях нагрузки.

В зависимости от характеристик, которые нам нужно протестировать, тестирование производительности делится на типы:

- нагрузочное тестирование (Loadtesting) – тестирование времени отклика приложения на запросы различных типов, с целью удостовериться, что приложение работает в соответствии с требованиями при обычной пользовательской нагрузке;
- стресс-тестирование (Stresstesting) – тестирование работоспособности приложения при нагрузках, превышающих пользовательские в несколько раз. При стресс-тестировании (зачастую, только при нем) мы можем получить реальные данные границ производительности приложения, исследовать способность программы обрабатывать исключения, ее стабильность и устойчивость. Именно в значительно увеличенной нагрузке на приложение и заключается разница между тестированием производительности и стресс тестированием;
- тестирование стабильности или наработка на отказ (Stability/Reliabilitytesting) исследует работоспособность приложения при длительной работе во времени, при нормальной для программы нагрузке;
- объемное тестирование (VolumeTesting) – тестирование проводится с увеличением не нагрузки и времени работы, а количества используемых данных, которые хранятся и используются в приложении.

Тестирование производительности – это проверка таких нефункциональных требований, как производительность и работоспособность приложения при различных нагрузках и условиях.

Существует два вида масштабирования:

- Вертикальное масштабирование
- Горизонтальное масштабирование

Вертикальное масштабирование – увеличение производительности каждого компонента системы с целью повышения общей производительности. Масштабируемость в этом контексте означает возможность заменять в существующей вычислительной системе компоненты более мощными и быстрыми по мере роста требований и развития технологий. Это самый простой способ масштабирования, так как не требует никаких изменений в прикладных программах, работающих на таких системах.

Горизонтальное масштабирование – разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам), и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию. Масштабируемость в этом контексте означает возможность добавлять к системе новые узлы, серверы, процессоры для увеличения общей производительности. Этот способ масштабирования может требовать внесения изменений в программы, чтобы программы могли в полной мере пользоваться возросшим количеством ресурсов.

В данном дипломном проекте рассматривается вертикальное масштабирование.

Сравнение производительности PostgreSQL и MySQL показано на рисунке 5.1.

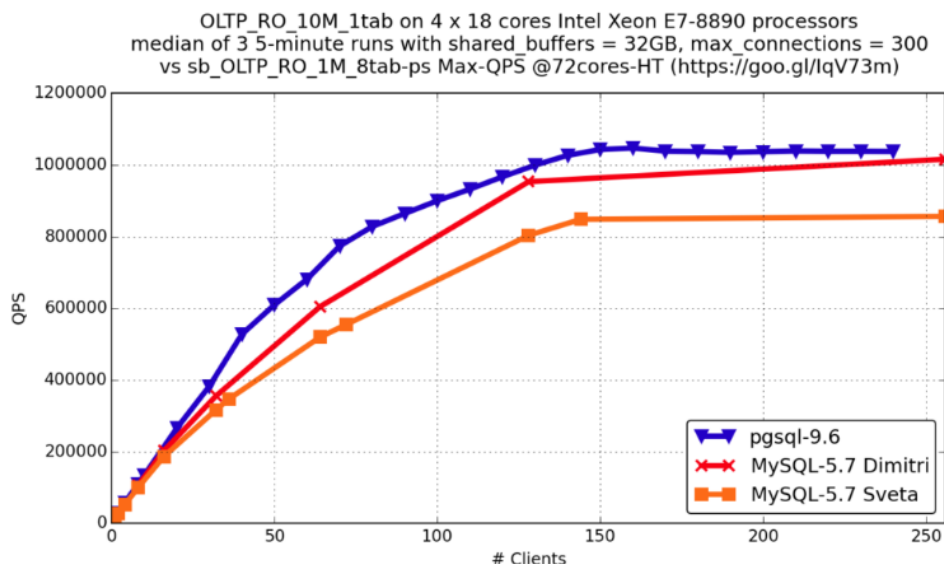


Рисунок 5.1 – Диаграмма обработки транзакций в режиме RO(только чтение) в зависимости от количества клиентов и запросов в секунду(QPS)

На рисунке 5.2 указана диаграмма выборки в зависимости от количества клиентов и запросов в секунду(QPS).

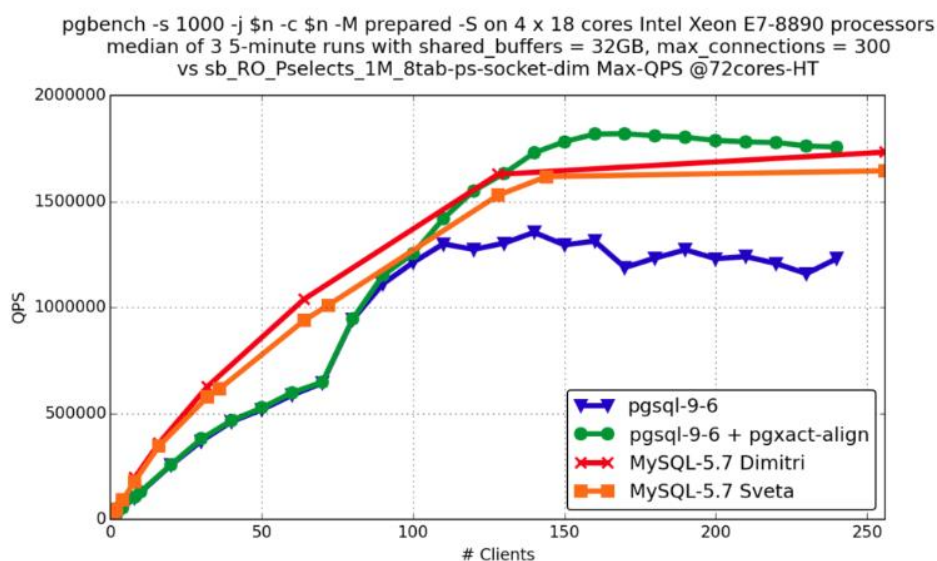


Рисунок 5.2 – Диаграмма выборки (SELECT) в зависимости от количества клиентов и запросов в секунду

Следующая диаграмма обработки транзакций в режиме реального времени (OLTP – Online Transactional Processing) в режиме работы RW(Чтение-Запись) в зависимости от количества подключений, показана на рисунке 5.3

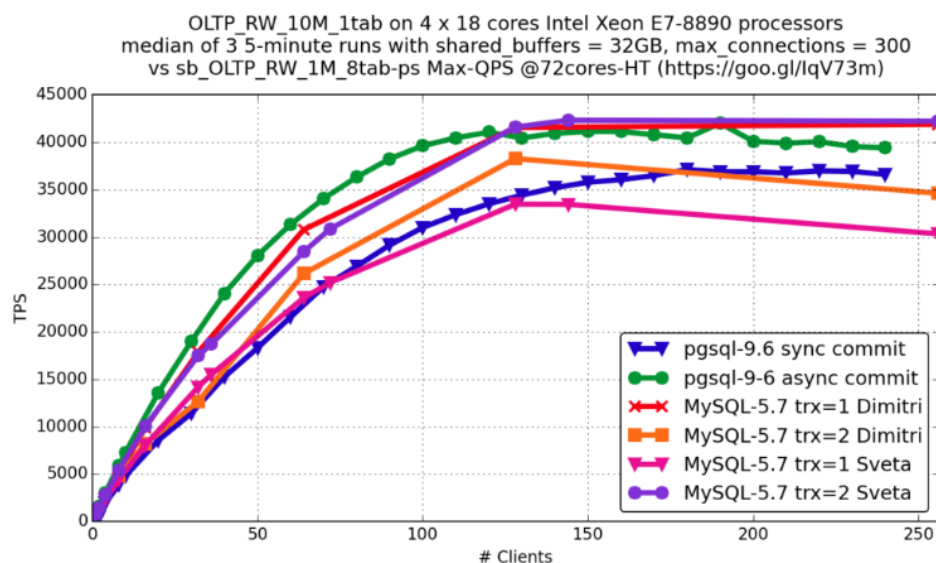


Рисунок 5.3 – Диаграмма обработки транзакций в реальном времени в зависимости от количества подключений.

Из данных графиков видно, что:

- PostgreSQL обеспечивает большую скорость обработки транзакций в режиме RO(только чтение);
- PostgreSQL работает на уровне конкурентов в режиме выборки в зависимости от количества клиентов и запросов в секунду и при обработке транзакций в реальном времени при режиме работы RW(чтение-запись) в зависимости от количества подключений.

Стоит отметить то, что изначально PostgreSQL создавался именно для баз данных с высокой нагрузкой и рассчитан на высокую производительность, которую можно добиться используя современные жёсткие диски – SSD-диск.

5.2 ТЕСТЫ ПРОИЗВОДИТЕЛЬНОСТИ PERCONA XTRADB CLUSTER В СРАВНЕНИИ С MYSQL GROUP REPLICATION ПРИ РАЗНЫХ ПАРАМЕТРАХ.

Теперь рассмотрим тесты кластеров баз данных при разных параметрах. Для MySQL 5.7.17 Group Replication рассмотрим два случая «долговечный» с `sync_binlog=1` и «расслабленный срок службы с `sync_binlog = 0`. Так же

рассмотрим с разными настройками параметров `flow_control = 25000` (по умолчанию). Это обеспечивает лучшую производительность, но с одним недостатком , при использовании параметра с такими настройками мы можем прочесть очень устаревшие данные, поэтому неосновные узлы могут отставать. Во втором случае будем использовать значение `flow_control = 1000`, чтобы минимизировать устаревшие данные на неосновных узлах.

Так же изменим параметр `flow_control = 100` для Percona XtraDB Cluster.

Результаты тестов при параметре `sync_binlog=1`(для MySQL Group Replication) показаны на рисунке 5.4.

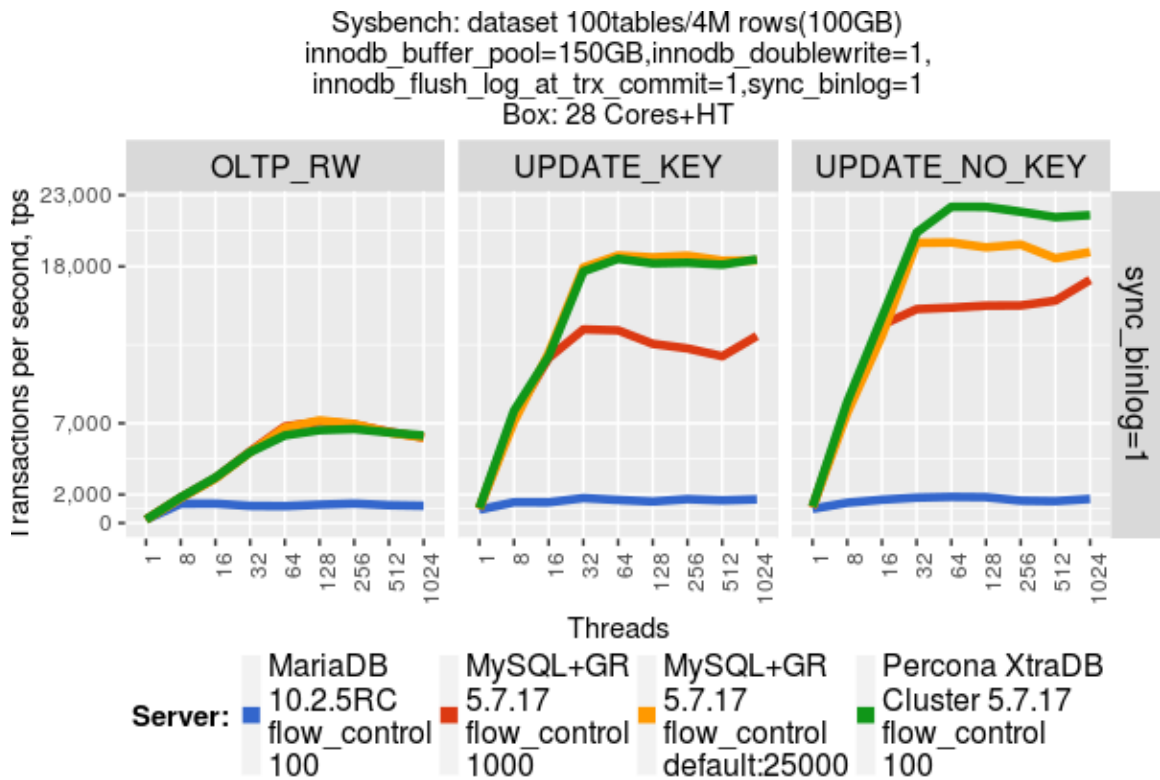


Рисунок 5.4 – Тест с параметром `sync_binlog=1`

Результаты тестов при параметре `sync_binlog=0` (для MySQL Group Replication) показаны на рисунке 5.5.

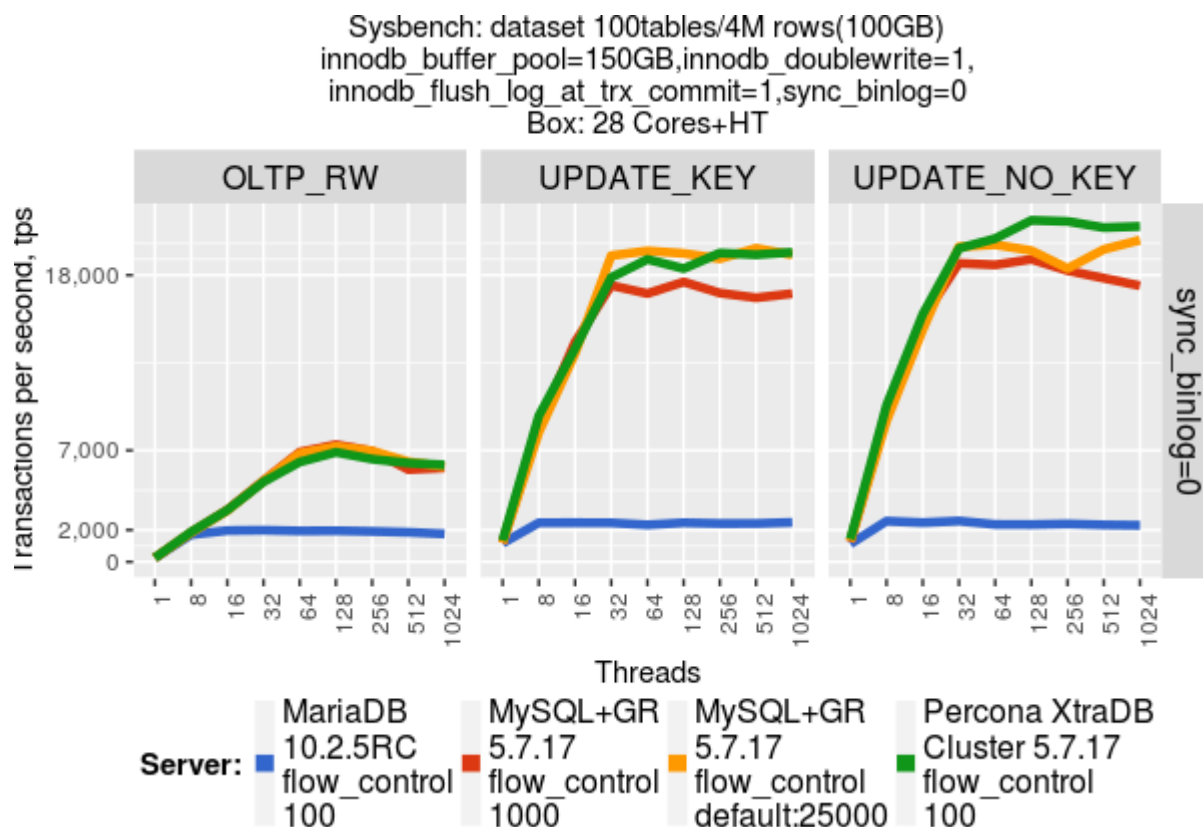


Рисунок 5.5 – Тест с параметром sync_binlog=0

Таким образом из графиков видно, что:

- Percona XtraDB Cluster работает на одном уровне с MySQL Group Replication.
- Параметр flow_control для MySQL Group Replication действительно влияет на производительность, и по умолчанию flow_control = 25000 лучше (с риском большого количества устаревших данных на неосновных узлах).

Можно заключить, что Percona XtraDB Cluster является более надёжным решением для репликации базы данных.

5.3 Тестирование репликации в Percona XtraDB Cluster

Чтобы проверить репликацию, создадим новую базу данных на втором узле, создадим таблицу для этой базы данных на третьем узле и добавим несколько записей в таблицу на первом узле. После этого получим все строки из данной таблицы на втором узле.

Код создания базы данных на втором узле представлен на рисунке 5.6.

```
mysql@pxc2> CREATE DATABASE percona;  
Query OK, 1 row affected (0.01 sec)
```

Рисунок 5.6 – Создание базы данных на втором узле

Код создания таблицы на третьем узле представлен на рисунке 5.7.

```
mysql@pxc3> USE percona;  
Database changed  
  
mysql@pxc3> CREATE TABLE example (node_id INT PRIMARY KEY,  
node_name VARCHAR(30));  
Query OK, 0 rows affected (0.05 sec)
```

Рисунок 5.7 – Создание таблицы на третьем узле.

Код добавления записи в первый узел представлен на рисунке 5.8.

```
mysql@pxc1> INSERT INTO percona.example VALUES (1, 'percona1');  
Query OK, 1 row affected (0.02 sec)
```

Рисунок 5.8 – Добавление записи в первый узел

Код получения всех строк из таблицы на втором узле представлен на рисунке 5.9

```
mysql@pxc2> SELECT * FROM percona.example;
+-----+-----+
| node_id | node_name |
+-----+-----+
|      1 | percona1  |
+-----+-----+
1 row in set (0.00 sec)
```

Рисунок 5.9 – Получение всех строк из таблицы на втором узле

Если процедура сработала это служит гарантией того, что все узлы в кластере синхронизированы и работают как и было задумано.

Кластеризация баз данных немного неоднозначна, некоторые считают, что кластер с двумя или более серверами использует одно и то же хранилище, другие называют кластер набором реплицированных серверов.

Репликация определяет метод, с помощью которого набор серверов остается синхронизированным без необходимости совместного использования хранилища, который может быть географически рассеян, есть два основных способа:

- Репликация master-master (или multi-master): любой сервер может обновлять базу данных. Обычно он заботится о другом модуле в базе данных (или в некоторых случаях используется другое программное обеспечение, работающее поверх них).

- Репликация master-slave: существует только одна копия авторитетных данных, которая выводится на подчиненные серверы.

Балансировка нагрузки – это другая концепция, она заключается в распределении запросов, отправленных на эти серверы, поэтому загрузка распределяется как можно равномерно. Обычно это делается на уровне приложения (или с пулом соединений). Единственная прямая связь между репликацией и балансировкой нагрузки заключается в том, что вам нужна некоторая репликация, чтобы иметь возможность загрузить баланс, иначе будет один сервер [8].

6. РАСЧЕТ ЭКОНОМИЧЕСКИХ ПОКАЗАТЕЛЕЙ

6.1 Исходные данные для проведения расчёта

Задачей данного дипломного проекта является создание лабораторного практикума для высоконагруженных реляционных СУБД.

Разработка лабораторного практикума по высоконагруженным реляционным базам данных, предусматривает проведение практически всех стадий проектирования и относится ко второй группе сложности.

Расчёты будут произведены в следующем порядке:

- расчёт объема функций;
- расчёт полной себестоимости лабораторного практикума.

6.2 Расчёт объёма функций программного модуля

Общий объём ПО определяется исходя из объёма функций, реализуемых программой:

$$V_0 = \sum_{i=1}^n V_i, \quad (6.1)$$

где V_0 – общий объём ПО;
 V_i – объём функций ПО;
 n – общее число функций.

В том случае, когда на стадии технико-экономического обоснования проекта невозможно рассчитать точный объём функций, то данный объём может быть получен на основании прогнозируемой оценки имеющихся фактических данных по аналогичным проектам, выполненным ранее, или применением нормативов по каталогу функций.

По каталогу функций на основании функций разрабатываемого ПО определяется общий объём ПО. Также на основе зависимостей от организационных и технологических условий, был скорректирован объём на основе экспертных оценок.

Уточнённый объём ПО (V_y) определяется по формуле:

$$V_y = \sum_{i=1}^n V_{yi}, \quad (6.2)$$

где V_{yi} – уточненный объем отдельной функции в строках исходного кода.
Перечень и объём функций ПО приведён в таблице 6.1.

Таблица 6.1 – Перечень и объём функции программного обеспечения

№ функции	Наименование (содержание) функции	Объём функции строк исходного кода	
		По каталогу V_i	Уточнённый V_{yi}
1	2	3	4
2	Организация ввода информации	90	72
3	Формирование баз данных	460	85
4	Организация ввода/вывода информации	1980	203
5	Обработка наборов и записей базы данных	2370	447
6	Организация поиска и поиск в базе данных	5200	620
7	Обеспечение интерфейса между компонентами	1321	510
Итого	-	11331	1937

С учётом информации, указанной в таблице 6.1, уточнённый объём ПО составил 1937 строк исходного кода вместо предполагаемого количества 11331 строк.

6.3 Расчёт полной себестоимости программного продукта

Стоимостная оценка программного средства у разработчика предполагает составление сметы затрат, которая включает следующие статьи расходов:

- заработную плату исполнителей (основную – $ЗП_{осн}$ и дополнительную – $ЗП_{доп}$);
- отчисления на социальные нужды ($P_{соц}$);
- материалы и комплектующие изделия (P_m);
- спецоборудование (P_c);
- машинное время ($P_{мв}$);
- расходы на научные командировки ($P_{нк}$);
- прочие прямые расходы ($P_{пр}$);
- накладные расходы ($P_{нр}$);
- затраты на освоение и сопровождение программного средства (P_o и $P_{со}$);

Полная себестоимость ($C_{п}$) разработки лабораторного практикума по высоконагруженным реляционным базам данных рассчитывается как сумма расходов по всем статьям с учётом рыночной стоимости аналогичных продуктов.

Основной статьёй расходов на создание лабораторного практикума по высоконагруженным реляционным базам данных является заработная плата проекта (основная и дополнительная) разработчиков (исполнителей) ($ЗП_{осн} + ЗП_{доп}$),

в число которых принято включать инженеров-программистов, руководителей проекта, системных архитекторов, дизайнеров, разработчиков баз данных, Web-мастеров и других специалистов, необходимых для решения специальных задач в команде.

Расчёт заработной платы разработчиков программного продукта начинается с определения:

- продолжительности времени разработки ($\Phi_{рв}$), которое устанавливается студентом экспертным путём с учётом сложности, новизны программного

обеспечения и фактически затраченного времени. В данном дипломном проекте $\Phi_{рв}=2$ месяца;

Разработчик лабораторного практикума по высоконагруженным реляционным базам данных.

Заработная плата разработчиков определяется как сумма основной и дополнительной заработной платы всех исполнителей.

Основная заработная плата каждого исполнителя определяется по формуле:

$$ЗП_{осн} = T_{ст.1р} \cdot \frac{T_k}{\Phi_{эфф.р.в.}} \cdot \Phi_{рв} \cdot K_{пр} , \quad (6.3)$$

где $T_{ст.1р}$ – месячная тарифная ставка 11 разряда рабочего (действующая на данном предприятии – 45 Br.);

T_k – тарифный коэффициент согласно разряду исполнителя;

$\Phi_{эфф.р.в.}$ – среднее количество рабочих дней;

$\Phi_{рв}$ – фонд рабочего времени исполнителя (продолжительность разработки программного продукта, дни);

$K_{пр}$ – коэффициент премии, $K_{пр} = 1,5$.

Рассчитаем основную заработную плату инженера-программиста и техника-программиста согласно формуле 6.3. Тарифный коэффициент согласно 1 разряду инженера-программиста $T_k = 2,65$. Продолжительность разработки лабораторного практикума по высоконагруженным реляционным базам данных составила – 60 дней.

Определение часовой тарифной ставки:

$$T_{час.ст.1р.} = \frac{T_{мес.ст.1р.} \cdot 12}{P_{рв}} , \quad (6.4)$$

где $T_{мес.ст.1р.}$ – месячная тарифная ставка рабочего 11 разряда (на 1.05.19 г. – 45 Br.);

$P_{рв}$ – расчётная норма рабочего времени (в часах) за год, определяется по производственному календарю текущего года (например, в 2019 г. Для 5-дневной недели при 40-часовой рабочей неделе – 2008 ч.).

Дополнительная заработная плата каждого исполнителя ($H_{\text{доп.зп}} - 20 \%$).

Рассчитывается от основной заработной платы по формуле:

$$ЗП_{\text{доп}} = ЗП_{\text{осн}} \cdot \frac{H_{\text{доп.зп}}}{100}, \quad (6.7)$$

Результаты вычислений внесём в таблицу 6.2

Таблица 6.2 – результаты вычислений

Категории работников	Разряд	Тарифный коэффициент (T_k)	$\Phi_{\text{эфф.р.в.}}$ (дн.)	Коэффициент премии ($K_{\text{пр}}$)	T_k (час.)	Зарплата, руб		
						Основная	Дополнительная	Всего
1	2	3	4	5	6	7	8	9
Инженер-программист	11	2,65	44	1,4	8	392	78,40	470,40
Итого	-	-	-	-	-	392	78,40	470,40

Таким образом, как видно из таблицы, заработная плата инженера-программиста составляет 470,4 Вг.

Отчисления на социальные нужды ($P_{\text{соц}}$) определяются в соответствии с действующим законодательством по нормативу (34 % – отчисления в ФСЗН + 0,6 % отчисления по обязательному страхованию):

$$P_{\text{соц}} = (3P_{\text{осн}} + 3P_{\text{доп}}) \cdot \frac{34,60}{100} \quad (6.6)$$

Расходы по статье «Спецоборудование» (P_c) включает затраты на приобретение технических и программных средств специального назначения, необходимых для разработки методического пособия, включая расходы на проектирование, изготовление, отладку и др.

В данном дипломном проекте для разработки лабораторного практикума по высоконагруженным реляционным базам данных приобретение какого-либо спецоборудования не предусматривалось. Так как спецоборудование не было приобретено, данная статья не рассчитывается.

По статье «Материалы и комплектующие изделия» (P_m) отражаются расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимы для разработки программного продукта. Норма расхода материалов в суммарном выражении определяется в расчете на 100 строк исходного кода по формуле:

$$P_m = H_m \cdot \frac{V_0}{100}, \quad (6.7)$$

где V_0 – уточнённый общий объём функций строк исходного кода. Согласно расчётам пункта 6.2 данное значение равно 1937 строк;

H_m – норма расхода материалов в расчете на 100 строк исходного кода программного продукта. Принимаем равной 1 Вт.

По статье «Машинное время» ($P_{\text{мв}}$) включают оплату машинного времени, необходимого для разработки и отладки программного продукта. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени.

$$P_{\text{мв}} = C_{\text{мвi}} \cdot \frac{V_0}{100} \cdot H_{\text{мв}}, \quad (6.8)$$

где $C_{\text{мвi}}$ – цена одного машино-часа (1,6 Вт.);

V_0 – уточнённый общий объём функций строк исходного кода;

$H_{\text{мв}}$ – норматив расхода машинного времени на отладку 100 строк кода, машино-часов. Принимается в размере 0,8.

Расходы по статье «Научные командировки» ($P_{\text{нк}}$) берутся либо по смете научных командировок, разрабатываемой на предприятии, либо в процентах от основной заработной платы исполнителей (10-15 %). Так как в данном проекте научные командировки не предусмотрены, данная статья не рассчитывается.

Расходы по статье «Прочие затраты» ($P_{\text{пр}}$) включают затраты на приобретение специальной научно-технической информации и специальной литературы. Так как специальная научно-техническая информация и специальная литература не приобреталась, то данная статья не рассчитывается.

Затраты по статье «Накладные расходы» ($P_{\text{нр}}$) связаны с содержанием вспомогательных хозяйств, а также с расходами на общехозяйственные нужды. Определяется по нормативу в процентах к основной заработной плате:

$$P_{\text{нр}} = \frac{H_{\text{нр}}}{100} \cdot 3П_{\text{осн}}, \quad (6.9)$$

где $H_{\text{нр}}$ – норматив накладных расходов, в данном дипломном проекте норматив накладных расходов равен 40 %.

Сумма вышеперечисленных расходов по статье на лабораторный практикум по высоконагруженным реляционным базам данных служит исходной базой для расчёта затрат на освоение и сопровождение программного продукта:

$$СЗ = 3П_{\text{осн}} + 3П_{\text{доп}} + P_{\text{соц}} + P_{\text{м}} + P_{\text{мв}} + P_{\text{с}} + P_{\text{нк}} + P_{\text{пр}} + P_{\text{нр}} \quad (6.10)$$

Затраты на освоение лабораторного практикума по высоконагруженным реляционным базам данных (P_o). Организация-разработчик участвует в освоении программного продукта и несёт соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Затраты на освоение определяются по установленному нормативу от суммы затрат:

$$P_o = СЗ \cdot \frac{H_o}{100}, \quad (6.11)$$

где $CЗ$ – сумма вышеперечисленных расходов по статьям на разработку лабораторного практикума по высоконагруженным реляционным базам данных;

H_o – установленный норматив затрат на освоение. Для данного дипломного проекта принимается равной 10 %.

Затраты на сопровождение лабораторного практикума по высоконагруженным реляционным базам данных (P_{co}). Организация-разработчик осуществляет сопровождение лабораторного практикума по высоконагруженным реляционным базам данных и несёт расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение.

$$P_{co} = CЗ \cdot \frac{H_{co}}{100}, \quad (6.12)$$

где $CЗ$ – сумма вышеперечисленных расходов по статьям на разработку программного продукта;

H_{co} – установленный норматив затрат на сопровождение программного продукта. Для данного дипломного проекта принимается равной 5 %.

Полная себестоимость (СП) разработки программного продукта рассчитывается как сумма расходов по всем статьям.

Определяется по формуле:

$$СП = CЗ + P_o + P_{co} \quad (6.13)$$

Результаты вычислений внесём в таблицу 6.3

Таблица 6.3 – результаты вычислений

№ расх.	Наименование статей затрат	Норматив, %	Сумма затрат Вг.
1	Заработная плата всего	-	470,00
1.1	Основная заработная плата	-	392,00
1.2	Дополнительная заработная плата	-	78,40

Продолжение таблицы 6.3

№ расх.	Наименование статей затрат	Норматив, %	Сумма затрат Вр.
2	Отчисления на специальные нужды	36,40 %	159,94
3	Спецоборудование	Не применялось	-
4	Материалы	Не применялись	-
5	Машинное время	-	129,55
6	Научные командировки	Не планировались	-
7	Прочие затраты	Не применялись	-
8	Накладные расходы	40 %	165,80
9	Сумма затрат	-	955,30
10	Затраты на освоение и сопровождение	9 %	85,90
12	Полная себестоимость	-	1136,80

Полная себестоимость программного продукта составляет 1136,8 Вр.

6.4 Расчёт цены и прибыли по лабораторному практикуму

Для определения цены лабораторного практикума по высоконагруженным реляционным базам данных необходимо рассчитать плановую прибыль, которая рассчитывается по формуле:

$$П = СП \cdot \frac{R}{100}, \quad (6.14)$$

где СП – полная себестоимость методических пособий, Вр.;

R – уровень рентабельности программного модуля. В данном дипломном проекте уровень рентабельности в размере 20%.

После расчета прибыли от реализации определяется прогнозируемая цена лабораторного практикума по высоконагруженным реляционным базам данных без налогов:

$$Ц_{п} = СП + П, \quad (6.15)$$

где СП – полная себестоимость методических пособий, Вг.;

П – плановая прибыль от реализации программного модуля, Вг.

Отпускная цена (цена реализации) программного продукта включает налог на добавленную стоимость и рассчитывается по формуле:

$$Ц_о = СП + П + НДС_{пп}, \quad (6.16)$$

где СП – полная себестоимость методических пособий, Вг.;

П – плановая прибыль от реализации методических пособий, Вг.;

НДС_{пп} – налог на добавленную стоимость для данного программного продукта, рассчитывается по формуле (6,28).

$$НДС_{пп} = Ц_{п} \cdot \frac{НДС}{100}, \quad (6.17)$$

где Ц_п – прогнозируемая цена, Вг.;

НДС – налог на добавленную стоимость, в настоящее время составляет 20%.

Прибыль от реализации лабораторного практикума по высоконагруженным реляционным базам данных за вычетом налога на прибыль является чистой прибылью (ПЧ).

$$ПЧ = П \cdot \left(1 - \frac{H_{п}}{100}\right), \quad (6.18)$$

где П – плановая прибыль от реализации программного модуля, Вг.;

H_п – ставка налога на прибыль (в настоящее время H_п = 18 %).

Все расчёты цены и прибыли по лабораторному практикуму по высоконагруженным реляционным базам данных сведены в таблицу 6.4.

Таблица 6.4 – Расчёт цены и прибыли.

№ ст.	Наименование статей затрат	Норматив, %	Сумма затрат, Br.
1	Полная себестоимость	-	1136,80
2	Прибыль	30	340,20
3	Цена без НДС	-	1398,20
4	НДС	20	261,46
5	Отпускная цена	-	1568,00
6	Налог на прибыль	18	32,40
7	Чистая прибыль	-	214,40

В ходе произведенных расчетов определены основные экономические показатели: полная себестоимость – 1136,8 Br.; прогнозируемая цена – 1568 Br.; Чистая прибыль – 214,4 Br.

Разработанный лабораторный практикум имеет малое количество конкурентов с более высокими ценами на их услуги. Таким образом, рассчитанная отпускная цена на программный продукт, разрабатываемой в рамках данного дипломного проекта, является конкурентоспособной. При расчете цены учтены отчисления в фонд социальной защиты, а также налоги, необходимые к уплате. К конечному итогу получаем окончательную цену продукта, равную 1398,2 белорусских рубля.

7 ОХРАНА ТРУДА И ЭКОЛОГИЧЕСКАЯ БЕЗОПАСНОСТЬ

7.1 Реализация пространственно-антропометрической эргономической совместимости студента и ПК

Так как тема дипломного проекта – разработка лабораторного практикума по высоконагруженным реляционным базам данных. В ходе работы с лабораторным практикумом учащийся будет взаимодействовать с электронно-вычислительным устройством. Поэтому важно уделять внимание инструкций по охране труда.

Технологический прогресс и широкое внедрение в производство информационных технологий значительно изменяют содержание и условия труда, что является предпосылкой для облегчения труда человека, освобождения его от выполнения однообразных трудоемких ручных операций, и вместе с тем приводит к появлению новых факторов, негативно влияющих на организм работников, среди которых на первое место выходит повышенная напряженность труда, обусловленная высокими требованиями к уровню психической деятельности человека. По этой причине внедрение в производство новейших технологий может быть успешно реализовано и дать положительный эффект лишь при достаточно полном учете характера все усложняющихся связей между человеком и техническим окружением всестороннем учете возможностей человека (человеческого фактора), его физиологических, психологических, антропометрических, эстетических и других свойств

Эргономика – это научная дисциплина, комплексно изучающая человека в конкретных условиях его деятельности в современном производстве. Основной объект исследования эргономики - система «человек-машина».

Рациональная совместимость возможностей человека и характеристик технических средств, оптимальное распределение функций между элементами системы «человек-машина» существенно повышают ее надежность, эффективность и обуславливают оптимальное использование человеком технических средств в соответствии с их назначением.

Пространственно-антропометрическая совместимость предполагает необходимость учета размеров тела человека, его возможности обзора внешнего пространства, определения зоны досягаемости для конечностей.

Антропометрические характеристики человека подразделяются на статические и динамические. К статическим характеристикам относятся размеры тела и его отдельных частей – рук, ног, кистей, стоп, к динамическим – возможные

углы поворота отдельных частей тела, зоны досягаемости. Антропологические особенности человека напрямую влияют на ход трудового процесса.

Суть трудового процесса заключается в воздействии работников на предметы труда с целью изготовления продукции, выполнения работ, оказания услуг, сопровождаемых затратами физической и нервной энергии.

Трудовой процесс включает в себя элементы, т.е. операции, приемы, действия, движения и факторы – тяжесть труда и напряженность труда.

Трудовая операция – это законченная часть технологического процесса, выполняемая на одном рабочем месте одним работником или группой работающих и включающая все их действия по выполнению единицы заданной работы над одним предметом труда.

Трудовой прием – это законченное действие работника в процессе выполнения операции, имеющее четкое целевое назначение. Трудовые приемы в свою очередь могут быть расчленены на трудовые действия.

Трудовое действие – это совокупность трудовых движений, совершаемых без перерыва рабочими органами трудящегося для выполнения части приема.

Трудовое движение – это однократное перемещение рук, ног, пальцев и туловища работника при выполнении трудового действия.

Эффективность и качество трудового процесса во многом зависят от методов труда.

Метод труда – это способ выполнения производственного задания, характеризующийся совокупностью определенных трудовых приемов и последовательностью их выполнения.

Тяжесть труда – это характеристика трудового процесса, отражающая преимущественную нагрузку на опорно-двигательный аппарат и функциональные системы организма (сердечно-сосудистую, дыхательную и др.), обеспечивающие его деятельность.

Напряженность труда – это характеристика трудового процесса, отражающая нагрузку преимущественно на центральную нервную систему, органы чувств, эмоциональную сферу работника.

Трудовая функция (работа по одной или нескольким профессиям, специальностям, должностям с указанием квалификации в соответствии со штатным расписанием нанимателя, функциональными обязанностями, должностной инструкцией). Наименование профессий, должностей, специальностей должно соответствовать квалификационным справочникам, утверждаемым в порядке, определяемом законодательством Республики Беларусь;

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места работника справочно-информационной службы должны быть соблюдены следующие основные условия:

- Оптимальное размещение оборудования, входящего в состав рабочего места.
- Достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.
- Необходимо естественное и искусственное освещение для выполнения поставленных задач.
- Уровень акустического шума не должен превышать допустимого значения.
- Достаточная вентиляция рабочего места.

Эргономическими аспектами проектирования рабочих мест, в частности, являются: высота рабочей поверхности, размеры пространства для ног, требования к расположению документов на рабочем месте (наличие и размеры подставки для документов, возможность различного размещения документов, расстояние от глаз пользователя до экрана, документа, клавиатуры и т.д.), характеристики рабочего кресла, требования к поверхности рабочего стола, регулируемость рабочего места и его элементов.

Главными элементами рабочего места являются письменный стол и кресло. Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Максимальная зона досягаемости рук – это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

На рисунке 7.1 приведены зоны досягаемости рук в горизонтальной плоскости:

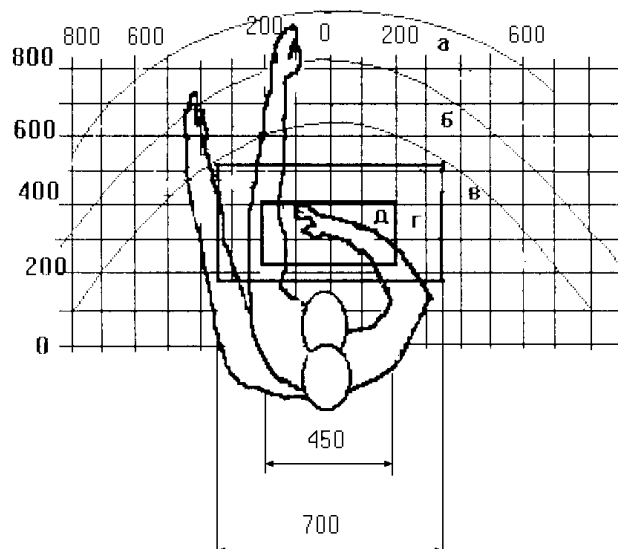


Рисунок 7.1 – Зоны досягаемости рабочего в горизонтальной плоскости:
а - зона максимальной досягаемости; б - зона досягаемости пальцев при вытянутой руке; в - зона легкой досягаемости ладони; г - оптимальное пространство для тонкой ручной работы; д - оптимальное пространство для тонкой ручной работы.

Основной частью рабочего места в большинстве случаев является письменный стол. Поэтому к нему предъявляются следующие требования:

- Высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники.
- Нижняя часть стола должна быть сконструирована так, чтобы специалист справочно-информационной группы мог удобно сидеть, не был вынужден поджимать ноги.
- Поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения специалиста справочно-информационной группы.
- Конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей, личных вещей).
- Высота рабочей поверхности рекомендуется в пределах от 680 до 760 мм. Высота рабочей поверхности, на которую устанавливается клавиатура, должна быть 650 мм. Высота края стола, обращенного к работающему с ПК, и высота пространства для ног должны соответствовать росту операторов в обуви.

Положение экрана монитора на рабочем месте должно быть в пределах данных параметров:

- Расстоянием считывания (0,60 + 0,10 м).
- Углом считывания, направлением взгляда на 20 ниже горизонтали к центру экрана, причем экран перпендикулярен этому направлению.
- Должна предусматриваться возможность регулирования экрана.
- По высоте плюс три сантиметра.
- По наклону от 10 до 20 см. Относительно вертикали.
- В левом и правом направлениях.

Зрительный комфорт подчиняется двум основным требованиям:

- Четкости на экране, клавиатуре и в документах.
- Освещенности и равномерности яркости между окружающими условиями и различными участками рабочего места.

Конструкция и размеры стола и кресла должны способствовать оптимальной позе оператора с определенными угловыми соотношениями между «шарнирными» частями тела. Это поможет сохранить здоровье и воспрепятствует возникновению симптомов синдромов компьютерного стресса и постоянных нагрузок. Рекомендуемые характеристики используемого рабочего места:

- высота рабочей поверхности стола 750 мм;
- высота сиденья над уровнем пола 450 мм;
- поверхность сиденья мягкая с закругленным передним краем;
- предусмотрена возможность размещения документов справа и слева;
- расстояние от глаза до экрана 700 мм;
- расстояние от глаза до клавиатуры 400 мм;
- расстояние от глаза до документов 500 мм;
- возможно регулирование экрана по высоте, по наклону.

Таким образом, можно заключить что при выполнении лабораторного практикума по высоконагруженным реляционным базам данных необходимо соблюдать вышеописанные эргономические принципы и режим труда. Рабочее место должно быть обустроено в соответствии с антропологическими характеристиками человека и рекомендуемым характеристикам рабочего места [9].

7.2 Обеспечение пожаробезопасности производства

Анализ причин возгораний и пожаров в учреждениях показывает, что основными причинами их являются:

- Неосторожное обращение с огнем и в первую очередь курение в цехах, на складах и других помещениях, где используются горючие материалы, ЛВЖ и ГЖ.

- Использование паяльных ламп и факелов для разогревания труб, несоблюдение правил пожарной безопасности при электро- и газосварочных работах.

- Неисправность электрооборудования, электросетей и электроаппаратуры (возгорание происходит в основном вследствие перегрузки электросети, коротких замыканий, загрязнения электрооборудования бумажной пылью и смазочными маслами, больших переходных сопротивлений).

- Нарушение технологического режима при работе на позолотных прессах, сушильных установках и т.п. (возгорание возникает чаще всего при повышении температуры выше рабочей из-за отсутствия или неисправности терморегулирующих устройств).

- Неосторожное обращение с огнём.

- Самовозгорание веществ и материалов.

- Грозовые разряды.

- Поджоги.

- Неправильное пользование газовым оборудованием.

- Солнечный луч, действующий через различные оптические системы.

- Умышленный поджог.

Основными причинами возникновения пожаров и взрывов на вышеперечисленных объектах являются случаи взрывов различных емкостей или отдельных участков трубопровода из-за технических неисправностей, увеличенного давления, повышенного внешнего температурного режима, а также различных механических аварий или повреждений. В качестве сторонних факторов могут выступать различные стихийные бедствия, другие техногенные аварии, военные действия и другие причины, которые вызваны резким изменением режима жизнедеятельности человека, социальных факторов или состояния окружающей природы.

При этом легковоспламеняющиеся жидкости, газы или вещества при горении могут нанести гораздо больший урон при отсутствии надежных и эффективных средств локализации и подавления таких аварий [10].

Противопожарная безопасность на взрывопожароопасных объектах разрабатывается как в виде профилактического комплекса мер и порядка, так и в виде правил активных мер по ликвидации пожаров или других аварийных ситуаций.

Меры предупреждения заключаются в создании условий и разработке мероприятий по предупреждению взрывов и пожаров, а сама профилактика достигается нижеперечисленными методами и способами:

- Разработка и утверждение индивидуальных для каждого промышленного объекта описываемой категории пожарных правил и норм. Кроме того, немаловажным условием является особый контроль за выполнением этих предупредительных мер.

- Осуществление проектирования и закладки конструкционных особенностей в ново созданные промышленные объекты с учетом требований противопожарной безопасности.

- Тщательный уход, своевременной обслуживание и периодическая проверка технического состояния противопожарных средств. В качестве необходимых и обязательных мероприятий в этом случае стоит выделить плановые периодические осмотры и обследования ответственными лицами с участием представителей государственных органов противопожарной безопасности.

Наиболее распространенными источниками возникновения чрезвычайных ситуаций техногенного характера являются пожары и взрывы, которые происходят:

- На промышленных объектах.
- На объектах добычи, хранения и переработки легковоспламеняющихся, горючих и взрывчатых веществ.
- На транспорте.
- В зданиях и сооружениях жилого, социально-бытового и культурного назначения.

Пожар – это вышедший из-под контроля процесс горения, уничтожающий материальные ценности и создающий угрозу жизни и здоровью людей. Основными причинами пожара являются: неисправности в электрических сетях, нарушение технологического режима и мер пожарной безопасности (курение, разведение открытого огня, применение неисправного оборудования и т.п.). Основными опасными факторами пожара являются тепловое излучение, высокая температура,

отравляющее действие дыма (продуктов сгорания: окиси углерода и др.) и снижение видимости при задымлении.

Взрыв – это быстропротекающий физический или физико-химический процесс, проходящий со значительным выделением энергии в небольшом объёме за короткий промежуток времени и приводящий к ударным, вибрационным и тепловым воздействиям на окружающую среду вследствие высокоскоростного расширения продуктов взрыва. Взрыв в твёрдой среде вызывает разрушение и дробление

При обнаружении возгорания реагируйте на пожар быстро, используя все доступные способы для тушения огня (песок, воду, огнетушители и т.д.). Если потушить огонь в кратчайшее время невозможно, вызовите пожарную охрану предприятия (при ее наличии) или города (по телефону 01). При эвакуации горящие помещения и задымленные места проходите быстро, задержав дыхание, защитив нос и рот влажной плотной тканью. В сильно задымленном помещении передвигайтесь ползком или пригнувшись – в прилегающем к полу пространстве чистый воздух сохраняется дольше. Не подходите к взрывоопасным предметам и не трогайте их. При угрозе взрыва ложитесь на живот, защищая голову руками, дальше от окон, застекленных дверей, проходов, лестниц. Если произошел взрыв, примите меры к недопущению пожара и паники, окажите первую медицинскую помощь пострадавшим.

Пожаротушение – процесс воздействия сил и средств, а также использование методов и приёмов для окончательного прекращения горения, а также на исключение возможности его повторного возникновения. Для возникновения и развития процесса горения необходимо одновременное присутствие горючего материала, окислителя и непрерывного потока тепла от огня пожара к горючему материалу (источника огня), то для прекращения горения достаточно отсутствие какого-нибудь из этих компонентов.

Таким образом, прекращение горения можно добиться снижением содержимого горючего компонента, уменьшением концентрации окислителя, уменьшением энергии активации реакции и, наконец, снижением температуры процесса.

В соответствии с вышесказанным существуют следующие основные способы пожаротушения:

- Охлаждение источника огня или горения ниже определённых температур.
- Изоляция источника горения от воздуха.

- Понижение концентрации кислорода воздуха путём разведения негорючими газами.
- Торможение (ингибирование) скорости реакции окисления.
- Механический срыв пламени сильной струей газа или воды, взрывом.
- Создание условий огнезаграждения, при которых огонь распространяется через узкие каналы, диаметр которых меньше диаметра гашения.

Для достижения этого применяют различные огнегасящие материалы и смеси (называемые далее веществами гашения или способами гашения).

Основными способами гашения пожара являются:

- Вода, которая может подаваться в огонь пожара цельными или распыленными струями.
- Пены (воздушно-механические и химические разной кратности), которые представляют собой коллоидные системы, состоящие из пузырьков воздуха (в случае воздушно-механической пены), окруженных пленкой воды.
- Инертные газовые разбавители (диоксид углерода, азот, аргон, водяной пар, дымовые газы).
- Гомогенные ингибиторы – галогенуглеводороды (хладоны) с низкой температурой кипения.
- Гетерогенные ингибиторы - порошки для гашения огня.
- Комбинированные смеси.

Выбор способа гашения и его подачи определяется классом пожара и условиями его развития.

Из вышеописанного заключим, что в ходе выполнения лабораторного практикума по высоконагруженным реляционным базам данных учащиеся и преподаватели должны знать правила противопожарной безопасности и уметь незамедлительно реагировать на возникновение чрезвычайной ситуации [11].

8 РЕСУРСО И ЭНЕРГОСБЕРЕЖЕНИЕ

С учётом постоянного роста применения информационных технологий энергетическая эффективность стала приоритетом для компьютерных систем общего и специального назначения. Это связано с одной стороны с необходимостью снижения уровня потребления энергии и снижению теплоотдачи поскольку данные факторы препятствуют увеличению производительности процессоров. Ещё одним фактором является сохранение экономико-энергетических ресурсов. Так как данный проект подразумевает работу с компьютером проблема энергосбережения для него актуальна.

Решение проблемы энергосбережения актуально для всего диапазона компьютерной техники. Для ПК характерно требование обеспечения пиковой производительности только в течении небольшого промежутка времени. КПД источников питания в маломощных ПК не превышает 90 %. Программное выключение ПК оставляет его в сети, и он продолжает потреблять примерно 3 Вт, с одной стороны это немного, но, если умножить эти цифры на сотни миллионов пользователей, получается значительный расход энергии.

В любой операционной системе существует несколько режимом энергосбережения. При их использовании сокращается потребление энергии и, как следствие экономия ресурсов [12].

У монитора есть два основах узла: блок вертикальной развёртки и блок горизонтальной развёртки. В зависимости от сочетания работающих и неработающих блоков существует четыре режима энергосбережения монитора:

– Нормальный режим – собственно, это не энергосберегающий режим, а основное состояние работающего монитора, когда оба блока работают. При работе в нормальном режиме монитор потребляет в среднем от 80 до 90 Вт.

– Ждущий режим – отключается блок горизонтальной развертки, а блок вертикальной развертки продолжает работать. Этот режим хорош, если вы ненадолго отошли от компьютера: монитор включается почти мгновенно, а экономия составляет порядка 10 Вт от общего энергопотребления.

– Режим приостановки – отключается блок вертикальной развертки, а блок горизонтально развертки продолжает работать. Выход из этого режима осуществляется дольше, но и экономия энергии значительнее: монитор потребляет в общей сумме около 15 Вт.

– Отключен – отключаются оба блока монитора. Для выхода из этого режима требуется примерно столько же времени, сколько необходимо монитору при включении питания, однако в этом режиме монитор потребляет только 5 Вт.

Ещё один режим экономии энергии где применяется отключение от жёсткого диска:

– Stand-by (Ждущий режим). Результат работы сохраняется в оперативной памяти компьютера, а затем компьютер переключается в энергосберегающий режим и отключает жесткий диск. Это быстрый и несложный способ уменьшить потребление электроэнергии.

– Hibernate (Режим гибернации). Текущее состояние системы сохраняется в специальном файле на жестком диске, после чего компьютер можно выключить. При последующем включении система вернется в сохраненное состояние.

В Windows Vista появился новый энергосберегающий режим - Hybrid Sleep (Гибридный спящий режим). В этом режиме результат работы сохраняется и в оперативную память, и на жесткий диск. В портативных компьютерах этот режим по умолчанию отключен.

Все операционные системы снабжены настройками энергосбережения. Например, в Windows XP это можно сделать, зайдя в Пуск -> Панель управления -> Электропитание. В Linux для этого существуют специальные команды, которые вводятся в консоли: `setterm`, `xset`. В MacOS в Системных настройках нужно выбрать вкладку «Энергосбережение».

Особенно энергосберегающие режимы актуальны для портативных компьютеров. При покупке рекомендуется выбирать модель с более долгим временем работы от аккумуляторов. Также полезно иметь в запасе дополнительные аккумуляторы и зарядное устройство и по возможности носить их с собой, сейчас многие публичные заведения (кафе, аэропорты, лекционные залы) снабжены розетками для владельцев ноутбуков. В ноутбуках, на которых установлена операционная система Windows, зачастую настройки энергосбережения вынесены в System tray (область пиктограмм панели задач в правом нижнем углу экрана).

То, рекомендуется настроить энергосберегающий режим работы, который не потребует от вас регулярного внимания, но будет постоянно способствовать экономии электроэнергии.

Используя природные ресурсы, стоит задуматься о том, что будет завтра.

Ресурсосбережение – уменьшение расхода сырьевых ресурсов, используемых в промышленности, при сохранении или увеличении количества конечной

продукции. Ресурсосбережение – один из важных элементов развития мирового сообщества в соответствии с моделью устойчивого развития [13].

Энергосбережение в любой сфере сводится по существу к снижению бесполезных потерь энергии. Анализ потерь в сфере производства, распределения и потребления электроэнергии показывает, что большая часть потерь – до 90 % – приходится на сферу энергопотребления, тогда как потери при передаче электроэнергии составляют лишь 10 %. Поэтому основные усилия по энергосбережению сконцентрированы именно в сфере потребления электроэнергии.

Энергосбережение – это научиться использовать энергию, находящуюся в нашем распоряжении, настолько эффективно и безопасно по отношению к окружающей среде, насколько это возможно.

Основная роль в увеличении эффективности использования энергии принадлежит современным энергосберегающим технологиям. Энергосберегающая технология – новый или усовершенствованный технологический процесс, характеризующийся более высоким коэффициентом полезного использования топливно-энергетических ресурсов (ТЭР).

В связи с переходом к интенсивному ресурсосберегающему типу экономического роста, основанного на использовании достижений НТР, снижении фондоемкости и материалоемкости продукции, повышения производительности труда, улучшении технико-экономических показателей и качества продукции возрастают возможности ресурсосбережения. Важное значение в решении проблемы ресурсосбережения имеет научно-технический прогресс.

Из вышеуказанного можно заключить, что важными факторами деятельности человека являются ресурсо- и энергосбережение. Учащиеся и преподаватели технического университета должны придерживаться данных принципов [14].

ЗАКЛЮЧЕНИЕ

Данный диплом был посвящен разработке лабораторного практикума по высоконагруженным реляционным СУБД.

В ходе выполнения дипломного проекта выполнена следующая работа:

- разработана концепция системы;
- описаны технологии при разработке;
- разработано информационное обеспечение;
- описан сценарий взаимодействия пользователя с системой;
- проведено тестирование системы;
- рассчитаны экономические показатели разработки практикума

и сделаны соответствующие выводы.

Разработанный практикум предназначен для студентов первой степени обучения в высших государственных учреждениях Республики Беларусь, с целью получения знаний по высоконагруженным реляционным СУБД.

Были разработаны следующие схемы:

- схема данных БД;
- схема алгоритма;
- схема ресурсов системы;
- схема взаимодействия программ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. О PostgreSQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/about/>. – Дата доступа: 17.05.2019.
2. О MySQL [Электронный ресурс]. – Режим доступа: <https://www.mysql.com/products/>. – Дата доступа: 17.05.2019.
3. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 3-е издание. – «Вильямс»: 2014. — 960 с.
4. О Postgres Pro [Электронный ресурс]. – Режим доступа: <https://postgrespro.ru/about>. – Дата доступа: 20.05.2019.
5. О Percona Xtrabld Cluster [Электронный ресурс]. – Режим доступа: <https://www.percona.com/doc/percona-xtradb-cluster/LATEST/intro.html>. – Дата доступа: 20.05.2019
6. Процедурное расширение языка SQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/9.6/plpgsql.html>. – Дата доступа: 22.05.2019.
7. Примеры языка запросов SQL [Электронный ресурс]. – Режим доступа: <https://sql-language.ru/query-select.html>. – Дата доступа: 20.05.2019
8. Настройка Percona Xtrabld Cluster [Электронный ресурс]. – Режим доступа: <https://www.percona.com/doc/percona-xtradb-cluster/5.7/configure.html>. – Дата доступа: 22.05.2019.
9. Трудовые процессы [Электронный ресурс]. – Режим доступа: <http://www.grandars.ru/shkola/bezopasnost-zhiznedeyatelnosti/trudovye-processy.html>. – Дата доступа: 16.05.2019.
10. Охрана труда: учебник. – 3-е изд., испр. и доп. – М.: ФОРУМ: ИНФРА-М, 2013. – 448 с.: ил
11. Обеспечение пожарной безопасности [Электронный ресурс]. – Режим доступа: <http://ki.by/catalog/obespechenie-ohrany-truda-i-pozharnoj-bezopasnosti>. – Дата доступа: 23.05.2019.
12. Ресурсосбережение и государственные требования к ресурсосбережению [Электронный ресурс]. – Режим доступа: <http://www.techvarious.ru/varwems-565-1.html>. – Дата доступа: 16.05.2019.
13. Володин, В. И. Энергосбережение: учеб. пособие / В. И. Володин. – Минск: БГУИР, БГТУ, 2001. – 182 с.
14. Ресурсосбережение [Электронный ресурс]. – Режим доступа: <https://studfiles.net/preview/4339377/>. – Дата доступа: 23.05.2019.