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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ НА ТЕМУ:

«Методы блокировки объектов в реляционных базах данных»

Студент ИУ7-56Б Пискунов П.

Студент ИУ7-55Б Юсупов Ф.К.

Руководитель Кострицкий А.С.

Москва — 2024 г.

РЕФЕРАТ

Научно – иследовательская работа 19 с., 3 рис., 1 табл., 7 ист.

БЛОКИРОВКИ В БАЗЕ ДАННЫХ, ТРАНЗАКЦИИ, УРОВНИ ИЗОЛЯЦИИ, МНОГОПОЛЬЗОВАТЕЛЬСКАЯ СРЕДА, СУБД, ПРОИЗВОДИТЕЛЬНОСТЬ.

Объект исследования – методы блокировки объектов в реляционных базах данных.

Целю работы является предоставить практические решения для обеспечения бесперебойной работы баз данных в условиях многопользовательской среды, где конфликты и взаимные блокировки могут оказать серьезное воздействие на результативность и надежность системы.

В данной исследовательской работе были рассмотрены методы управления блокировками в реляционных базах данных, охватывающие уровни строки, столбцов и страниц, а также включающие в себя блокировки намерений, механизмы обнаружения и разрешения взаимных блокировок. Также проведен анализ воздействия различных уровней изоляции на производительность и согласованность данных.

Результат данной работы показал, что каждый метод имеет ряд преимуществ и недостаткам и применим в зависимости от требований и ограничений к задаче.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

Блокировка в реляционных базах данных это установка метки на запись, что запись заблокирована для изменений [1].

Многопользовательская среда – среда, в которой база данных одновременно используется несколькими пользователями, что создает потенциальные конфликты и взаимные блокировки.

Уровни изоляции – механизмы, которые присваиваются нужной транзакции и выдают при ее работе соответствующие блокировки [2].

Транзакция – совокупность операций, выполняемых прикладной программой, которые переводят согласованное состояние базы данных в согласованное [3].

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В текущей расчетно-пояснительной записке применяется следующие сокращения и обозначения.

СУБД – система управления базами данных.

PostgreSQL – это реляционная база данных.

ВВЕДЕНИЕ

В рамках нашего исследования глубоко проанализируем проблемы взаимных блокировок, возникающих в многопользовательских средах. Кроме того, предполагается рассмотрение различных методов и стратегий для их обнаружения и разрешения, включая специализированные методы блокировки объектов и изучение воздействия различных уровней изоляции транзакций на эффективность и согласованность данных в реляционных базах данных.

Целью работы является предоставить практические решения для обеспечения бесперебойной работы баз данных в условиях многопользовательской среды, где конфликты и взаимные блокировки могут оказать серьезное воздействие на результативность и надежность системы.

Для достижения поставленной цели необходимо решить следующие задачи:

1) проанализировать существующие методы управления блокировками, механизмы обеспечения целостности данных и методы уровней изоляции в реляционных базах данных;

2) установить требования и ограничения для обеспечения бесперебойной работы баз данных в многопользовательской среде, учитывая особенности конкретной системы;

3) разработать стратегию управления блокировками, включая определение оптимальных уровней изоляции и применение соответствующих методов обнаружения и разрешения взаимных блокировок;

4) подготовить отчет по данной работе.

1 Анализ предметной области

1.1 Введение в современные реляционные базы данных

В мире современной информационной технологии реляционные базы данных играют решающую роль в хранении и управлении информацией, обеспечивая быстрый и структурированный доступ к данным. Однако, в многозадачных средах, где множество пользователей одновременно обращается к базам данных, могут возникнуть ситуации, которые представляют серьезные вызовы для обеспечения целостности, производительности и надежности данных. Одной из таких ситуаций является взаимная блокировка, или deadlock.

1.2 Проблема взаимных блокировок

В многозадачной среде, где множество пользователей одновременно обращаются к базе данных, возникают ситуации, когда один пользователь ожидает освобождение ресурса (строку или таблицу), необходимый для завершения своей работы, в то время как другой пользователь, в свою очередь ожидает освобождения другого ресурса [4]. Это явление, называемое взаимной блокировкой (deadlock), создает конфликты и приводит к замедлению работы системы. Разрешение взаимных блокировок и управление уровнями изоляции становятся ключевыми аспектами обеспечения целостности, производительности и надежности баз данных в таких многопользовательских средах.

На рисунке 1.1 приведен пример взаимной блокировки (deadlock) [5].

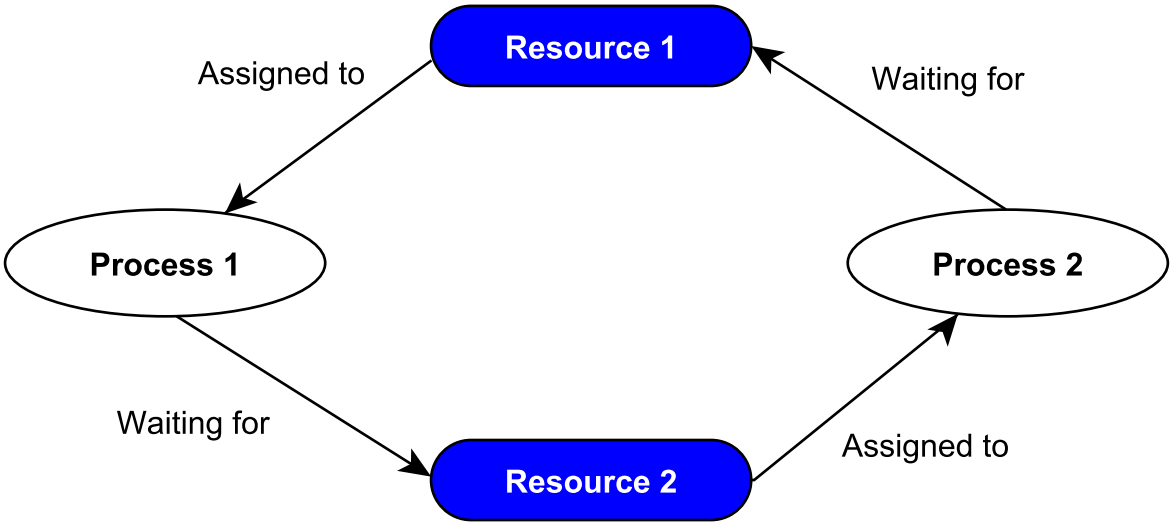

Рисунок 1.1 – Пример взаимной блокировки (deadlock).

Целостность баз данных означает, что данные остаются согласованными и корректными в течение всего их жизненного цикла, несмотря на разнообразные операции и изменения.

Производительность баз данных связана с способностью обеспечивать высокую скорость доступа к данным и эффективную обработку запросов пользователей.

Надежность баз данных означает, что система сохраняет работоспособность и доступность даже в случае сбоев или непредвиденных ситуаций.

1.3 Многопользовательские среды: вызовы и задачи

Многопользовательская среда представляет собой ситуацию, в которой множество пользователей одновременно взаимодействует с базой данных. Это может быть организация, предоставляющая доступ к своей базе данных через сеть, веб-приложение с множеством одновременных пользователей или любая другая среда, где данные используются и изменяются сразу несколькими пользователями. В таких средах обеспечение целостности, производительности и надежности баз данных становится особенно актуальным и сложным

заданием.

Однако, в контексте многопользовательских сред, возникают уникальные проблемы, связанные с параллельным доступом к данным и управлением транзакциями.

Транзакция – совокупность операций, выполняемых прикладной программой, которые переводят согласованное состояние базы данных в согласованное [3], если:

- отсутствуют помехи со стороны других приложений;
- транзакция выполнена полностью.

Недостаточная координация между пользователями и некорректное управление транзакциями может привести к взаимным блокировкам и конфликтам, которые влияют на целостность данных и производительность системы. В таких условиях возникает необходимость в разработке и применении методов обнаружения и разрешения взаимных блокировок.

1.4 Взаимоблокировка в многозадачных средах

Наиболее распространенная форма взаимоблокировки возникает, когда два или более потоков ожидают ресурса, который принадлежит другому потоку [6]. Это проиллюстрировано следующим образом:

Поток1	Поток2
Принимает блокировку A	Принимает блокировку B
Блокировка запросов B	Запросы блокируют A

Если обе последовательности выполняются одновременно, поток 1 никогда не получит блокировку B, так как она принадлежит потоку 2, а поток 2 никогда не получит блокировку A, так как принадлежит потоку 1. В лучшем случае это приводит к остановке участвующих потоков, а в худшем – к остановке системы. Реляционная база данных может следить за тем, какие транзакции блокируют какие ресурсы и какие запросы на разблокировку они выполняют.

2 Обзор существующих методов блокировок

2.1 Классификация блокировок

По области действия блокировки классифицируются на строчные, табличные и предикатные. По строгости блокировки разделяются на совместные (англ. *shared*) и исключительные (англ. *exclusive*). По логике реализации блокировки делятся на оптимистические и пессимистические.

В данной теме рассмотрены блокировки базы данных: строчная блокировка, страничная блокировка, блокировка намерений, и табличная блокировка [7].

2.1.1 Строчная блокировка

Строчная блокировка — действуют только на одну строку таблицы базы данных, не ограничивая манипуляции над другими строками таблицы. Снимаются такие блокировки в конце транзакции или при откате к точке сохранения.

Различные СУБД предоставляют разные уровни блокировок на уровне строки, обеспечивая различные уровни изоляции и поведения при параллельном доступе к данным. В PostgreSQL, как и во многих других СУБД, существуют различные уровни изоляции транзакций, которые определяют, какие типы блокировок применяются при доступе к строкам данных.

Полный перечень конфликтов блокировок на уровне строк приведён в

таблице 2.1.

Таблица 2.1 – Таблица с режимами блокировки

Запрашиваемый режим блокировки	**Текущий режим блокировки**			
	FOR KEY	FOR SHARE SHARE	FOR NO KEY UPDATE	FOR UPDATE
FOR KEY SHARE				X
FOR SHARE			X	X
FOR NO KEY UPDATE		X	X	X
FOR UPDATE	X	X	X	X

Можно заметить, что одна транзакция может владеть несколькими конфликтующими блокировками одной строки, даже в разных подтранзакциях; но две разных транзакции никогда не получат конфликтующие блокировки одной и той же строки.

FOR UPDATE

FOR UPDATE – в этом режиме строки, выданные оператором *SELECT*, блокируются как для изменения. При этом они защищаются от блокировки, изменения и удаления другими транзакциями до завершения текущей. То есть другие транзакции, пытающиеся выполнить *UPDATE*, *DELETE*, *SELECT FOR UPDATE*, *SELECT FOR NO KEY UPDATE*, *SELECT FOR SHARE* или *FOR KEY SHARE* с этими строками, будут заблокированы до завершения текущей транзакции.

FOR NO KEY UPDATE

Действует подобно *FOR UPDATE*, но запрашиваемая в этом режиме блокировка слабее: она не будет блокировать команды *SELECT FOR KEY SHARE*, пытающиеся получить блокировку тех же строк. Также запрашивается любой командой *UPDATE*, которая не требует блокировки *FOR UPDATE*.

FOR SHARE

Действует подобно *FOR NO KEY UPDATE*, за исключением того, что для каждой из полученных строк запрашивается разделяемая, а не исключительная блокировка. Разделяемая блокировка не позволяет другим транзакциям выполнять с этими строками *UPDATE, DELETE, SELECT FOR UPDATE* или *SELECT FOR NO KEY UPDATE*, но допускает *SELECT FOR SHARE* и *SELECT FOR KEY SHARE*.

FOR KEY SHARE

Действует подобно *FOR SHARE*, но устанавливает более слабую блокировку: блокируется *SELECT FOR UPDATE*, но не *SELECT FOR NO KEY UPDATE*. Блокировка разделяемого ключа не позволяет другим транзакциям выполнять команды *DELETE* и *UPDATE*, только если они меняют значение ключа (но не другие UPDATE), и при этом допускает выполнение команд *SELECT FOR NO KEY UPDATE*, *SELECT FOR SHARE* и *SELECT FOR KEY SHARE*.

2.1.2 Табличная блокировка

Табличная блокировка (*гранулярная*) — действует на всю таблицу или всю страницу и все строки. Блокировка, ограничивающая манипуляции со страницей данных в таблице.

В PostgreSQL существуют различные режимы блокировок, применяемые на уровне таблицы. Несмотря на то, что названия некоторых режимов могут казаться связанными с отдельными строками данных, фактически эти режимы применяются ко всей таблице. Каждый режим блокировки имеет определенные характеристики и контексты применения, определяемые конфликтующими режимами.

Единственное, что действительно отличает один режим блокировки от другого, это набор режимов, с которыми конфликтует каждый из них. Конфликтующие блокировки на уровне таблицы можно посмотреть в таблице 2.2. Две транзакции не могут одновременно владеть блокировками конфликтующих режимов для одной и той же таблицы.

Таблица 2.2 – Конфликтующие блокировки на уровне таблицы

Запрашиваемый режим блокировки	**Текущий режим блокировки**							
	ACCESS SHARE	ROW SHARE	ROW EXCLUSIVE	SHARE UPDATE EXCLUSIVE	SHARE	SHARE ROW EXCLUSIVE	EXCLUSIVE	ACCESS EXCLUSIVE
ACCESS SHARE								X
ROW SHARE							X	X
ROW EXCLUSIVE					X	X	X	X
SHARE UPDATE EXCLUSIVE				X	X	X	X	X
SHARE			X	X		X	X	X
SHARE ROW ROW EXCLUSIVE			X	X	X	X	X	X
EXCLUSIVE		X	X	X	X	X	X	X
ACCESS EXCLUSIVE	X	X	X	X	X	X	X	X

Представлены основные режимы блокировок на уровне таблицы.

ACCESS SHARE

Конфликтует только с режимом блокировки *ACCESS EXCLUSIVE*.

Команда *SELECT* получает такую блокировку для таблиц, на которые она ссылается. В ообщем, блокировку в этом режиме получает любой запрос, который только читает таблицу, но не меняет её данные.

ROW SHARE

Конфликтует с режимами блокировки:

- EXCLUSIVE;
- ACCESS EXCLUSIVE.

Команды *SELECT FOR UPDATE* и *SELECT FOR SHARE* получают такую блокировку для своих целевых таблиц.

ROW EXCLUSIVE

Конфликтует с режимами блокировки:

- SHARE;
- SHARE ROW EXCLUSIVE;

- EXCLUSIVE;
- ACCESS EXCLUSIVE.

Команды *UPDATE, DELETE* и *INSERT* получают такую блокировку для целевой таблицы. В ообщем, блокировку в этом режиме получает любая команда, которая изменяет данные в таблице.

SHARE UPDATE EXCLUSIVE

Конфликтует с режимами блокировки:

- SHARE UPDATE EXCLUSIVE;
- SHARE;
- SHARE ROW EXCLUSIVE;
- EXCLUSIVE;
- ACCESS EXCLUSIVE.

Этот режим защищает таблицу от параллельного изменения схемы и запуска процесса *VACUUM*.

Запрашивается командами:

- VACUUM;
- ANALYZE;
- CREATE INDEX CONCURRENTLY;
- CREATE STATISTICS;
- COMMENT ON.

Также запрашивает *ALTER TABLE VALIDATE* и другими видами *ALTER TABLE*.

SHARE

Конфликтует с режимами блокировки:

- ROW EXCLUSIVE;
- SHARE UPDATE EXCLUSIVE;
- SHARE ROW EXCLUSIVE;
- EXCLUSIVE;
- ACCESS EXCLUSIVE.

Этот режим защищает таблицу от параллельного изменения данных. Запрашивается командой *CREATE INDEX*.

SHARE ROW EXCLUSIVE

Конфликтует с режимами блокировки:

- ROW EXCLUSIVE;
- SHARE UPDATE EXCLUSIVE;
- SHARE ROW EXCLUSIVE;
- EXCLUSIVE;
- ACCESS EXCLUSIVE.

Этот режим защищает таблицу от параллельных изменений данных и при этом он является самоисключающим, так что такую блокировку может получить только один сеанс.

Запрашивается командой *CREATE COLLATION*, *CREATE TRIGGER* и многими формами *ALTER TABLE*.

EXCLUSIVE

Конфликтует с режимами блокировки:

- ROW SHARE;
- ROW EXCLUSIVE;
- SHARE UPDATE EXCLUSIVE;
- SHARE;
- SHARE ROW EXCLUSIVE;
- EXCLUSIVE;
- ACCESS EXCLUSIVE.

Этот режим совместим только с блокировкой *ACCESS SHARE*, то есть параллельно с транзакцией, получившей блокировку в этом режиме, при этом, допускается только чтение таблицы.

Запрашивается командой *REFRESH MATERIALIZED VIEW CONCURRENTLY*.

ACCESS EXCLUSIVE

Конфликтует с режимами блокировок:

- ACCESS SHARE;
- ROW SHARE;
- ROW EXCLUSIVE;
- SHARE UPDATE EXCLUSIVE;
- SHARE;
- SHARE ROW EXCLUSIVE;
- EXCLUSIVE;

- ACCESS EXCLUSIVE.

Этот режим гарантирует, что кроме транзакции, получившей эту блокировку, никакая другая транзакция не может обращаться к таблице каким-либо способом.

Запрашивается командами *DROP TABLE, TRUNCATE, REINDEX*, *CLUSTER, VACUUM FULL* и *REFRESH MATERIALIZED VIEW*. Блокировку на этом уровне запрашивают также многие виды *ALTER TABLE*. В этом режиме по умолчанию запрашивают блокировку, и операторы *LOCK TABLE*, если не был выбран другой режим.

2.1.3 Блокировка намерений

Блокировки намерений, также известные как блокировки намерений для обновления, указывают на намерение изменить определенную строку [8]. Блокировки намерений приобретаются, когда транзакция:

- выполняет операцию FETCH FOR UPDATE;
- выполняет операцию SELECT...FOR UPDATE BY LOCK;
- использует SQL_CONCUR_LOCK в качестве основы конкурентности в приложении ODBC (устанавливается с помощью параметра SQL_ATTR_CONCURRENCY вызова API ODBC SQLSetStmtAttr).

Блокировки намерений не конфликтуют с блокировками чтения, поэтому приобретение блокировки намерений не блокирует другие транзакции от чтения той же строки. Однако блокировки намерений предотвращают другие транзакции от приобретения как блокировки намерений, так и блокировки записи для той же строки, гарантируя, что строка не может быть изменена другой транзакцией перед обновлением.

Если транзакция, выполняющаяся в изоляции снимка, запрашивает блокировку намерений, блокировка намерений приобретается только в том случае, если строка является неизмененной строкой в базе данных и общей для всех одновременных транзакций. Однако, если строка является копией снимка, блокировка намерений не приобретается, поскольку исходная строка

уже была изменена другой транзакцией. Любая попытка снимковой транзакции обновить эту строку завершится неудачей, и будет возвращена ошибка конфликта обновления снимка.

2.1.4 Страничная блокировка

Блокировка данных на уровне страницы подразумевает блокировку группы записей, обычно содержащихся в одной странице базы данных.

Блокировка на уровне страницы может быть полезной в ситуациях, когда несколько транзакций могут одновременно читать и модифицировать разные записи в пределах одной страницы. Однако, это также может привести к проблемам с производительностью, таким как «эффект фантомных записей».

Эффект фантомных записей (англ. "phantom read") является одним из аномальных видов чтения данных в многозадачных системах баз данных, где одна транзакция видит изменения, внесенные другой транзакцией, после того как она начала свое выполнение.

Когда транзакция читает ряд данных, который соответствует какому-то условию, и в процессе выполнения другая транзакция добавляет или удаляет записи, соответствующие этому условию, то транзакция, выполняющая чтение, может «увидеть» новые записи или не увидеть удаленные записи, что создает впечатление существования «фантомных» (воображаемых) записей.

Пример: Транзакция A выполняет запрос SELECT * FROM Orders WHERE Status = 'Новый'; и получает результат. Транзакция B добавляет новый заказ с Status = 'Новый';. Транзакция A выполняет тот же запрос снова и видит новый заказ, который был добавлен транзакцией B (фантомный эффект).

Эффект фантомных записей может вызвать проблемы согласованности данных в многозадачной среде. Решения включают в себя использование различных уровней изоляции транзакций, таких как SERIALIZABLE, для предотвращения таких конфликтов.

2.1.5 Взаимоблокировка в многозадачных средах: детекция и методы разрешения

После обнаружения взаимных блокировок, система должна разрешить их чтобы продолжить выполнение транзакций. Описаны несколько методов разрешения блокировок, такие как:

- автоматическое разрешение: СУБД может автоматически определить какую из блокировок следует откатить, чтобы разрешить конфликт и продолжить выполнение транзакцией;
- ручное разрешение: администратор базы данных может вмешаться и вручную откатить определенную транзакцию, чтобы разрешить блокировку.

Для ручного управления блокировками в реляционных базах данных существует механизм уровня изоляции, который присваивается нужной транзакции и выдает при её работе соответствующие блокировки[2].

Виды уровни изоляции:

- Read Uncommitted;
- Read Committed;
- Snapshot;
- Repeatable Read;
- Serializable.

Read Uncommitted – полное отсутствие исключительных и разделяемых блокировок. Максимальная производительность работы с такими данными, которые могут быть изменены любой транзакцией и вызвать недействительное чтение.

Read Committed – разделяемые блокировки присутствуют, пока считываются данные.

Snapshot – с данных, к которым обращается транзакция, берется копия. Каждая транзакция может делать это повторно и работать только со своей копией данных.

Repeatable Read – разделяемые блокировки накладываются на все данные, с которыми работает транзакция.

Serializable – блокировки диапазона данных с которыми работает транзакция. Во время такой блокировки запрещены вставки и обновления записей другими пользователями.

Вывод

Блокировки баз данных играют важную роль в обеспечении целостности данных при параллельном доступе нескольких пользователей или приложений к общим данным. Они помогают предотвратить конфликты доступа к данным и поддерживают целостность информации.

Блокировка данных ограничивает возможность другим пользователям или приложениям получать доступ или изменять данные во время выполнения операций обновления или изменения. Однако использование блокировок на уровне таблицы может снизить производительность приложения из-за ограничения доступа ко всей таблице.

Понимание типов блокировок и их контекстов использования важно для эффективного управления доступом к данным и поддержания целостности базы данных в многопользовательской среде.

ЗАКЛЮЧЕНИЕ

В результате исследования методы обнаружения и разрешения взаимных блокировок, а также анализа уровней изоляции в системах управления базами данных, становится ясным, что эти аспекты играют критическую роль в обеспечении эффективности и надежности работы с данными в многопользовательских сценариях.

Одним из ключевых выводов является необходимость постоянного совершенствования методов обнаружения взаимных блокировок, чтобы минимизировать негативное воздействие на производительность и обеспечить бесперебойную работу системы. Современные технологии исследований в области баз данных продолжают активно разрабатывать новые методы, направленные на повышение эффективности обнаружения подобных конфликтов.

В контексте уровней изоляции важно отметить, что выбор подходящего уровня изоляции зависит от конкретных требований консистентности данных и производительности приложения.

От уровня READ UNCOMMITTED, обеспечивающего минимальную изоляцию, до SERIALIZABLE, обеспечивающего максимальную, каждый уровень предоставляет свой баланс между видимостью данных и защитой от взаимных блокировок.

Заключение исследования подчеркивает важность тщательного анализа требований приложения при выборе методов обнаружения блокировок и уровней изоляции. Осознанный и грамотный подход к этим аспектам способствует созданию устойчивых, высокопроизводительных и надежных систем управления базами данных в условиях динамичного многопользовательского окружения.

Кроме того, важно отметить, что блокировки баз данных играют важную роль в обеспечении целостности данных при параллельном доступе нескольких пользователей или приложений к общим данным. Они помогают предотвратить конфликты доступа к данным и поддерживают целостность информации.

Блокировка данных ограничивает возможность другим пользователям или приложениям получать доступ или изменять данные во время выполнения операций обновления или изменения. Однако использование блокировок на уровне таблицы может снизить производительность приложения из-за

ограничения доступа ко всей таблице.

Понимание типов блокировок и их контекстов использования важно для эффективного управления к доступам данных и поддержания целостности базы данных в многопользовательской среде.

В ходе данной работе были:

1) проанализированы существующие методы управления блокировками, механизмы обеспечения целостности данных и методы уровней изоляции в реляционных базах данных;

2) установлены требования и ограничения для обеспечения бесперебойной работы баз данных в многопользовательской среде, учитывая особенности конкретной системы;

3) разработана стратегия управления блокировками, включая определение оптимальных уровней изоляции и применение соответствующих методов обнаружения и разрешения взаимных блокировок;

4) подготовлен отчет по данной работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Блокировки в БД [Электронный ресурс]. — Режим доступа: https://uthark.github.io/2009/04/22/blog-post_22/ (дата обращения: 10.01.2024).

[2] Разработка имитационной модели системы эффективных блокировок в централизованных базах данных [Электронный ресурс]. — Режим доступа: https://nauchkor.ru/uploads/documents/5f392531cd3d3e0001babe8b.pdf (дата обращения: 11.11.2023).

[3] Теория транзакций с примерами из Microsoft SQL Server [Электронный ресурс]. — Режим доступа: https://alexeykalina.github.io/technologies/transactions.html?ysclid=lonfn0hfzk336258152 (дата обращения: 11.11.2023).

[4] Обнаружение дефекта взаимной блокировки с помощью статического анализа [Электронный ресурс]. — Режим доступа: https://ispranproceedings.elpub.ru/jour/article/view/1344/1145 (дата обращения: 11.11.2023).

[5] Deadlock в СУБД [Электронный ресурс]. — Режим доступа: https://www.boardinfinity.com/blog/deadlock-in-dbms/ (дата обращения: 11.11.2023).

[6] Обнаружение взаимоблокировки [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/windows-hardware/drivers/devtest/deadlock-detection (дата обращения: 11.11.2023).

[7] PostgresPro [Электронный ресурс]. — Режим доступа: https://postgrespro.ru/docs/postgrespro/10/explicit-locking (дата обращения: 15.12.2023).

[8] Intent locks [Электронный ресурс]. — Режим доступа: https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.

sqlanywhere.12.0.1/dbusage/transact-s-5031849.html (дата обращения: 10.01.2024).