

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

|  |
| --- |
|  |

Институт информационных технологий

Кафедра инструментального и прикладного программного обеспечения (ИППО)

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №9**

**по дисциплине**

«Технологии обработки транзакций клиент-серверных приложений»

Выполнил студент группы ИКБО-16-21 Нурулла А.

Принял Маличенко С. В.

Практическая работа выполнена «\_\_» \_\_\_\_\_\_\_ 2024 г.

«Зачтено» «\_\_» \_\_\_\_\_\_\_ 2024 г.

Москва 2024

**ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ**

Необходимо выполнить следующие шаги:

1. Написать тест, проверяющий, что обработка очереди, показанная в демонстрации, работает корректно при выполнении в несколько потоков. Убедиться, что тест не проходит, если убрать предложение FOR UPDATE SKIP LOCKED.
2. Добавить в реализацию проверку «зависших» сообщений. Если такая ситуация будет обнаружена, зависшее сообщение должно быть снова принято в работу.
3. Вставить в таблицу сообщений большое количество строк и проверить, что:
4. было обработано каждое сообщение;
5. каждое сообщение было обработано ровно один раз.
6. Убрать из реализации секундную задержку (имитацию работы), чтобы тест выполнялся быстрее и с достаточным уровнем конкурентности между процессами.

**РЕФЕРАТ**

Нурулла А., Практическая работа направления подготовки «Программная инженерия» на тему «Транзакции. Асинхронная обработка»: М. 2024 г., МИРЭА – Российский технологический университет (РТУ МИРЭА), Институт информационных технологий (ИИТ), кафедра инструментального и прикладного программного обеспечения (ИиППО) – 16 стр., 19 рис., 10 источн.

Ключевые слова: POSTGRESQL, БАЗА ДАННЫХ, ТАБЛИЦА, SQL, ТРАНЗАКЦИЯ, БЛОКИРОВКА, АСИНХРОННАЯ ОБРАБОТКА.

Целью работы является реализация асинхронных механизмов обработки транзакций для повышения эффективности системы, исследование проблем в области транзакционной асинхронной обработки.

Nurulla A., Practical work of the direction of training “Software Engineering” on the theme “Transactions. Asynchronous Processing”: M. 2024, MIREA – Russian Technological University (RTU MIREA), Institute of Information of Instrumental and Applied Software (IAPS) – 16 p., 19 fig., 10 sources.

Keywords: POSTGRESQL, DATABASE, TABLE, SQL, TRANSACTION, LOCK, ASYNCHRONOUS PROCESSING.

The aim of the work is to implement asynchronous transaction processing mechanisms to improve the efficiency of system and research problems in the field of transactional asynchronous processing.

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ**

|  |  |  |
| --- | --- | --- |
| SQL | — | Structured Query Language |
| БД | — | База данных |
| СУБД | — | Система управления базой данных |
|  |  |  |

**ПЕРЕЧЕНЬ ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ**

1. **База данных** – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.
2. **Таблица** – совокупность связанных данных, хранящихся в структурированном виде в базе данных.
3. **SQL-скрипт** – сохраненная в текстовом формате SQL инструкция, с помощью которой создаются объекты базы данных.
4. **Транзакция** – последовательность операторов языка SQL, которая рассматривается как некоторое неделимое действие над базой данных, осмысленное с точки зрения пользователя.
5. **Блокировка** – метод предотвращения одновременного доступа к данным в БД, позволяющий обеспечить отсутствие противоречивых результатов.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc163060922)

[ХОД РАБОТЫ 8](#_Toc163060923)

[ЗАКЛЮЧЕНИЕ 14](#_Toc163060924)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 15](#_Toc163060925)

# ВВЕДЕНИЕ

В современных клиент-серверных приложениях часто возникает необходимость в асинхронной обработке данных и транзакций. Асинхронная обработка позволяет разнести во времени возникновение события и его обработку, что дает ряд следующих преимуществ:

1. Клиент не вынужден ожидать завершения обработки, что повышает отзывчивость приложения.
2. Возможность управлять ресурсами и скоростью обработки событий в соответствии с требованиями и нагрузкой.
3. Распараллеливание обработки событий для повышения производительности.

Одним из способов реализации асинхронной обработки является использование очередей сообщений, в которые помещаются события, требующие обработки. События извлекаются из очереди отдельными процессами-обработчиками и обрабатываются асинхронно по отношению к основному потоку приложения.

В данной практической работе рассматривается реализация очереди сообщений средствами PostgreSQL для асинхронной обработки транзакций в клиент-серверных приложениях. Будут изучены различные аспекты такой реализации, включая обработку исключительных ситуаций, оптимизацию производительности и обеспечение целостности данных при конкурентном доступе.

# ХОД РАБОТЫ

Для начала будет подготовлены таблицы и функции для изучения и реализации механизмов синхронной обработки данных (рис. 1-6).

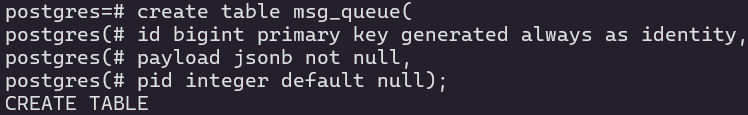


Рисунок 1 – Создание таблицы под очередь сообщений

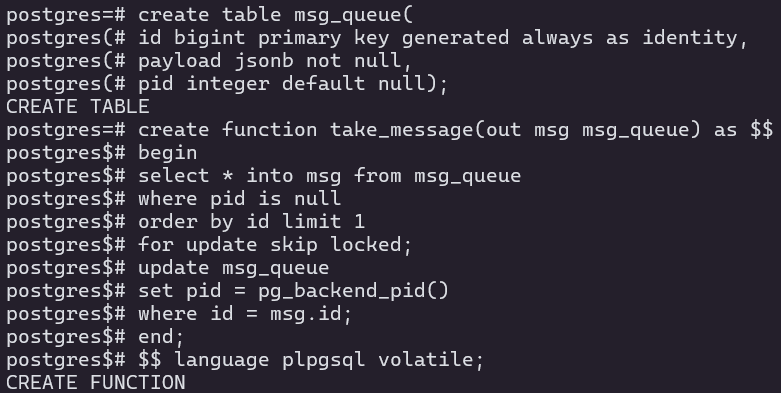


Рисунок 2 – Создание функции для получения и блокирования сообщения

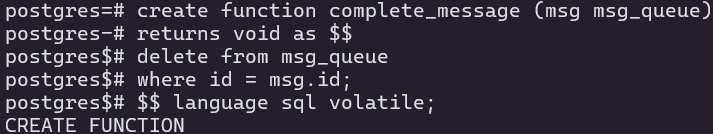


Рисунок 3 – Создание функции завершения работы с сообщением

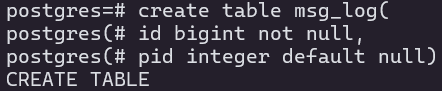


Рисунок 4 – Создание таблицы для подсчета обработанных сообщений

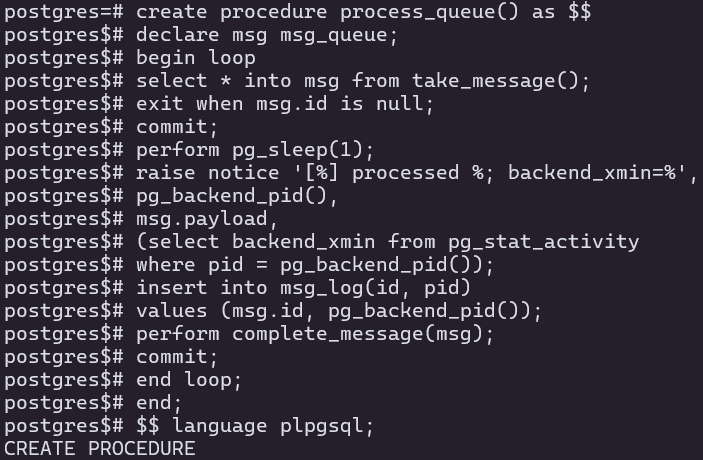


Рисунок 5 – Создание процедуры обработки сообщений

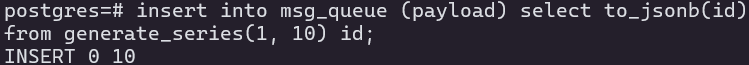


Рисунок 6 – Заполнение очереди сообщений

На Рисунках 7-8 вызывается обработка очереди в два потока и проверяется работа обработчика.

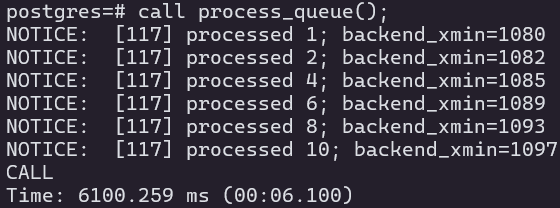


Рисунок 7 – Вызов процедуры обработки (первая сессия)

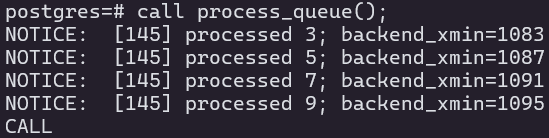


Рисунок 8 – Вызов процедуры обработки (вторая сессия)

Далее изменяется функция получения сообщения – убирается проверка на блокировку строки другим процессом и проводится подсчет обработанных сообщений (рис. 9-12).

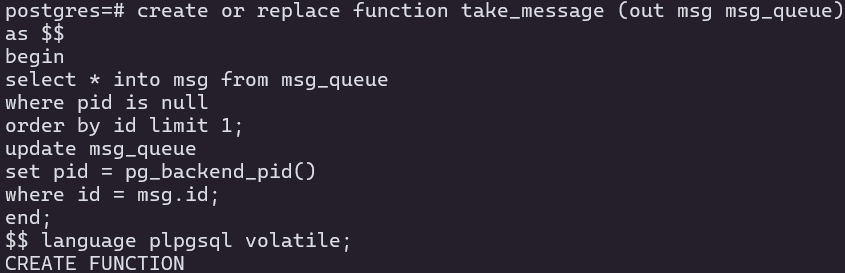


Рисунок 9 – Изменение функции получения и блокировки сообщения. Удаление FOR UPDATE SKIP LOCKED

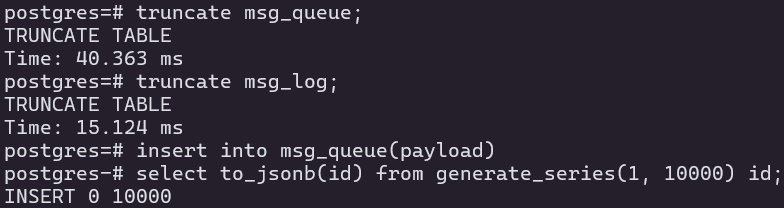


Рисунок 10 – Перезаполнение таблиц

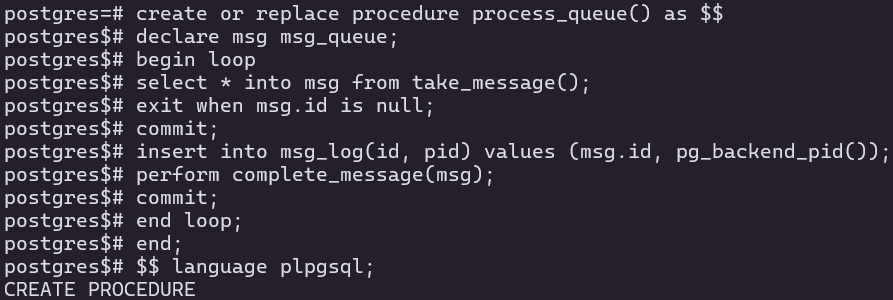


Рисунок 11 – Изменение процедуры обработки очереди

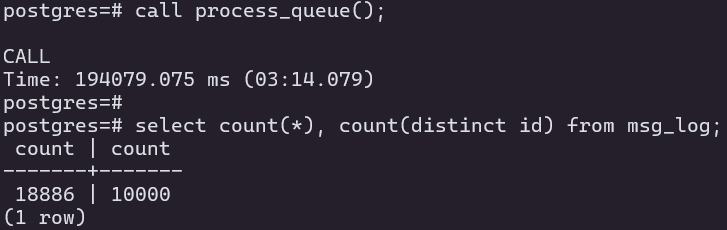


Рисунок 12 – Выполнение процедуры без FOR UPDATE SKIP LOCKED

На рисунке 13 выведена информация о повторных обработках сообщений несколькими процессами.

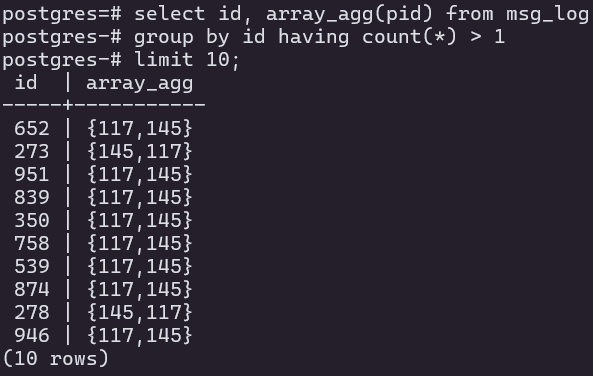


Рисунок 13 – Результат работы без FOR UPDATE SKIP LOCKED

Наконец, возвращается исходное состояние функции и проводятся повторные замеры (рис. 14-16).

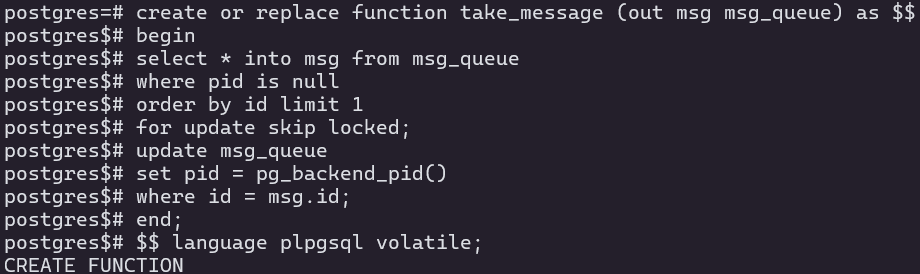


Рисунок 14 – Возврат исходной функции

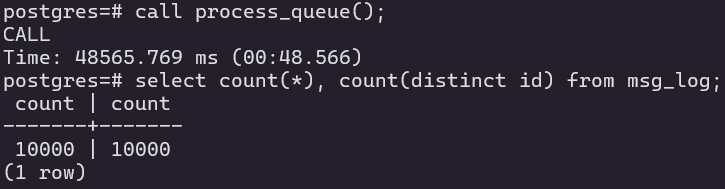


Рисунок 15 – Вызов процедуры с FOR UPDATE SKIP LOCKED

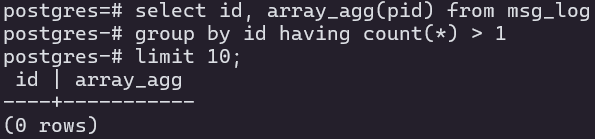


Рисунок 16 – Результат работы исходной функции

В дополнение к FOR UPDATE SKIP LOCKED, добавляется проверка «зависших сообщений» в функцию take\_message. Таблицы заполняются большим количеством записей и проводятся замеры (рис. 17-19).

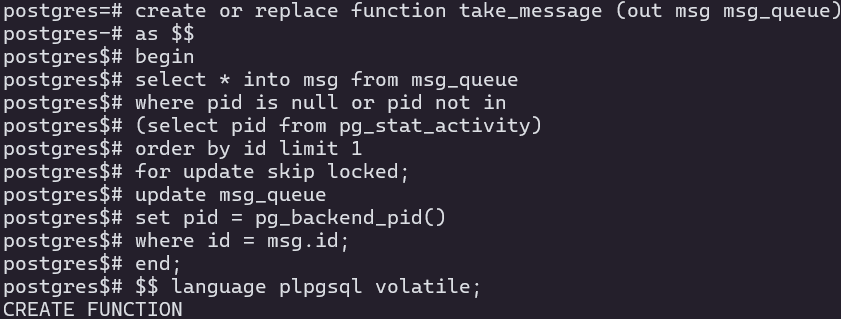


Рисунок 17 – Добавление в функцию получения и блокировки сообщения проверки «зависших» сообщений

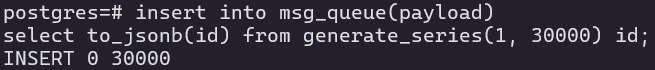
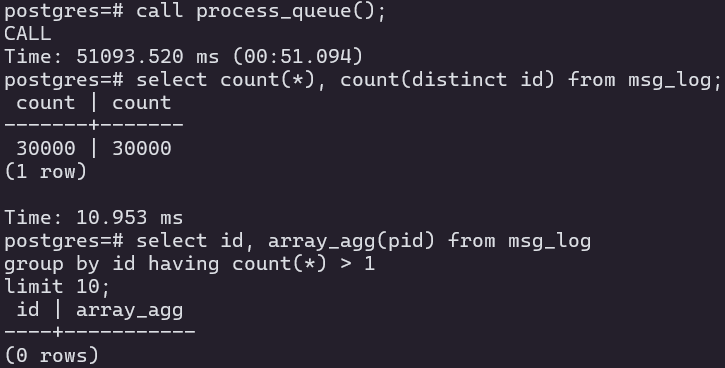


Рисунок 18 – Заполнение таблицы большим количеством сообщений

  
Рисунок 19 – Выполнение обработки с большим количеством строк

Как видно на Рисунке 19, обработка сообщений происходит корректно: каждое сообщение обработано ровно один раз.

# ЗАКЛЮЧЕНИЕ

В ходе практической работы изучены и реализованы асинхронные механизмы обработки транзакций для повышения эффективности системы, исследованы проблем в области транзакционной асинхронной обработки. Также рассмотрена реализация механизма асинхронной обработки данных и транзакций с использованием очереди сообщений. Были изучены различные аспекты такой реализации, включая эффективное получение очередного события из очереди, избежание долгих транзакций, обработку исключительных ситуаций и "зависших" сообщений.

При реализации очереди сообщений внутри PostgreSQL важно уделить внимание таким моментам, как эффективное извлечение и блокирование очередного сообщения, предотвращение долгих транзакций за счет периодической фиксации, корректная обработка исключительных ситуаций и "зависших" сообщений.

В целом, реализация асинхронной обработки транзакций с помощью очередей сообщений средствами PostgreSQL является простым и эффективным решением для многих задач, когда не требуется использовать сторонние системы управления очередями сообщений.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Новиков Б. А. Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. — 2-е изд. — М.: ДМК Пресс, 2020. — 582 с. – URL: <https://postgrespro.ru/education/books/dbtech> (дата обращения: 20.02.2024)
2. Новиков Б. А. Лекции Основы технологий баз данных. – URL: <https://postgrespro.ru/education/university/dbtech> (дата обращения: 17.12.2023)
3. Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.: ил. – URL: <https://postgrespro.ru/education/books/sqlprimer> (дата обращения: 20.02.2024)
4. Моргунов, Е. П. Технологии разработки программ на основе инструментария с открытым исходным кодом. Вводный курс: учеб. пособие / Е. П. Моргунов, О. Н. Моргунова, В. В. Тынченко; НИИ СУВПТ. – Красноярск, 2006. – 148 с. – URL: <http://www.morgunov.org/docs/free_soft_tech.pdf> (дата обращения: 20.02.2024)
5. Лузанов П.В. и др. Postgres. Первое знакомство. – 178 с. – URL: <https://postgrespro.ru/education/books/introbook> (дата обращения: 20.02.2024)
6. BEGIN, COMMIT, ROLLBACK (работа с транзакциями) – URL: <https://ydb.tech/docs/ru/postgresql/statements/begin_commit_rollback> (дата обращения: 20.02.2024)
7. Postgres Pro Standard – Явные блокировки – URL: <https://postgrespro.ru/docs/postgresql/16/explicit-locking> (дата обращения 12.03.2024)
8. Блокировки в PostgreSQL: 2. Блокировки строк – Erogov (Habr) – URL: <https://habr.com/ru/companies/postgrespro/articles/463819/> (дата обращения 12.03.2024)
9. Блокировки в PostgreSQL: 3. Блокировки других объектов – Erogov (Habr) – URL: <https://habr.com/ru/companies/postgrespro/articles/465263/> (дата обращения 12.03.2024)
10. Асинхронная обработка команд – Postgres Pro Standard – URL: <https://postgrespro.ru/docs/postgrespro/16/libpq-async> (дата обращения 03.04.2024)