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:*

## *«Описание метода замещения страниц в разделяемом кэш буфере postgres»*

Студент ИУ7-33М (Группа) _____ (Подпись, дата) Мицевич М. Д. (И. О. Фамилия)

Руководитель НИР _____ (Подпись, дата) Тассов К. Л. (И. О. Фамилия)

*2024 г.*

# РЕФЕРАТ

Расчетно-пояснительная записка к научно-исследовательской работе содержит 15 страниц, 2 иллюстраций, 0 таблиц, 6 источников.

Научно-исследовательская работа представляет собой разработку метод замещения страниц в разделяемом кэш буфер PostgreSQL. Рассмотрены особенности метода замещения страниц в PostgreSQL. Представлено описание метода в виде детализированной idef0 диаграммы.

Ключевые слова: страница, замещение, кэш буфер, PostgreSQL.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Современные базы данных, такие как PostgreSQL, сталкиваются с постоянно растущими требованиями к производительности и эффективности управления ресурсами. Одним из ключевых аспектов работы базы данных является управление памятью, включая работу с разделяемым кэш буфером [1]. Разделяемый кэш буфер используется PostgreSQL для хранения данных, которые часто запрашиваются, что позволяет сократить количество операций чтения с диска и ускорить доступ к данным.

Однако, по мере роста объема данных и количества запросов, разделяемый кэш буфер может оказаться переполненным, что приводит к необходимости вытеснения старых или менее используемых данных. Метод замещения страниц играет важную роль в этом процессе, определяя, какие данные должны быть удалены из кэша, чтобы освободить место для новых.

Целью данной работы является разработка метода замещения страниц в разделяемом кэш буфере PostgreSQL.

Для достижения поставленной цели требуется выполнить следующие задачи:

— изложить особенности разрабатываемого метода;

— описать основные этапы метода в виде детализированной idef-0 диаграммы;

— спроектировать структуру программного обеспечения.

# 1 Разработка метода

## 1.1 Разделяемый кэш буфер PostgreSQL

Кэширование используется в современных компьютерных системах повсеместно: один только процессор имеет три или четыре уровня кэша. В общем случае кэш нужен для того, чтобы сгладить разницу в производительности двух видов памяти, один из которых в разы быстрее, но меньше по размеру, а другой имеет обратные характеристики размера и времени доступа. Буферный кэш сохраняет страницы в оперативной памяти, доступ к которой в сотни тысяч раз быстрее, чем к дисковому хранилищу, где содержится вся информация о состоянии базы данных.

В операционной системе также есть дисковый кэш, который решает ту же проблему, поэтому системы управления базами данных обычно стараются избежать двойного кэширования, обращаясь к дискам напрямую, а не через кэш ОС. В случае с PostgreSQL это не так: все данные читаются и записываются с помощью обычных файловых операций [2]. При разработке метода важно учитывать, что контроллеры дисковых массивов и даже сами диски также имеют свой собственный кэш.

Буферный кэш является списком буферов, каждый из которых состоит из блока данных и заголовка. Заголовок содержит:

— номер блока страницы;

— индикатор того, что страница была изменена, но еще не записана на диск;

— число обращений к буферу;

— число активных операций или транзакций, которые используют буфер.

При старте все буферы кэша помещаются в список свободных. Для поиска нужной страницы используется хэш таблица. В качестве ключа используются номер файла и номер страницы в файле.

При обращении к памяти процесс сначала пытается найти страницу в кэше. Если она уже загружена, то счетчик обращений в заголовке соответ-

ствующего буфера увеличивается на единицу. До тех пор, пока это счетчик не равен нулю, страница не может быть выгружена из кэша.

Если страница не была найдена в кэше, то она должна быть прочитана с диска в какой-то буфер. Если список свободных буферов не пуст, то будет взят первый буфер из него, иначе требуется выбрать буфер, который будет вытеснен из кэша.

В PostgreSQL для выбора кандидата на замещение анализируются два счетчика – число обращений и количество использований. Алгоритм часы поочередно проходит по всем буферам в кэше и, если оба счетчика равны нулю, то текущий буфер будет замещен, иначе оба счетчика уменьшаются на единицу. Для избежания большого числа проходов по всем буферам в поисках кандидата на замещение по умолчанию счетчики не могут быть больше пяти.

Когда кандидат на замещение найден, счетчик использований ассоциированного с ним буфера устанавливается в 1, чтобы другие процессы не могли его использовать. Если буфер содержит не записанную диск информацию, то запускается фоновый процесс переписывания страницы на диск. После этого новая страница загружается в буфер и для нее выставляется счетчик обращений в единицу.

## 1.2 Постановка задачи

Пусть, размер разделяемого кэш буфера в СУБД равен $B$. Тогда в каждый момент времени $t$, когда возникает новое обращение к странице в памяти, СУБД пытается найти нужный буфер в кэше. Если он отсутствует и кэш заполнен, то при помощи алгоритма замещения должна быть выбрана страница, которая будет вытеснена из буфера. Результатом работы метода является $a^t \in \{0, 1, ..., B-1\}$ – индекс страницы, которая будет вытеснена.

Для оценки разработанного алгоритма будут использованы следующие метрики качества:

— коэффициента попадания – отношение числа обращений к страницам, которые уже загружены в буфере, к общему числу обращений;

— коэффициент совпадения – показывает отношение количества совпавших

с оптимальным алгоритмов кандидатов на замещение с общим числом запросов поиска буфера для вытеснения.

## 1.3 Описание модели

Разрабатываемая модель состоит из четырех компонентов [3]:

1. Кодировщик запроса доступа к странице – полносвязная нейронная сеть, которая на вход получает атрибуты запрашиваемой страницы, а на выходе выдает вектор $e_t$ размерности $d_e$.
2. Кодировщик страниц в буфере – полносвязная нейронная сеть, на вход которой поступают атрибуты страниц в буфере и результатом которой является список векторов для каждой страницы в буфере. Размерность каждого вектора равна $d_b$.
3. Кодировщик истории обращений к страницам;
4. Модуль выбора страницы для замещения.

**Кодировщик запроса доступа к странице.** При каждом запросе страницы из запроса извлекаются следующие параметры:

— $rel\_id$ – идентификатор отношения;

— $is\_local\_temp$ – флаг, который отвечает показывает является отношение временным в текущей сессии;

— $fork\_num$ – тип физического хранилища;

— $blk\_num$ – номер блока в файле;

— $mode$ – метод чтения страницы;

— $rel\_am$ – тип индекса;

— $rel\_file\_node$ – идентификатор физического хранилища;

— $has\_index$ – флаг наличия индекса;

— $rel\_persistence$ – тип хранения объекта (постоянный, временный, объект, который не ведет журнал транзакций);

— $rel\_kind$ – тип отношения: обычная таблица, индекс, последовательность, таблица с специальной техникой хранения больших объектов, отображение, сложный тип, внешняя таблица, разделенная таблица, разделенный индекс;

— $rel\_natts$ – список пользовательских атрибутов;

— $relfrozenxid$ – идентификатор транзакции, который указывает на момент, когда все старые версии строк в данной таблице были заморожены;

— $relminmxid$ – минимальный идентификатор многоверсионной транзакции для данной таблицы.

Помимо описанных выше атрибутов к входным параметрам модели добавляется индекс страницы в буфере, если она там есть, или число равное размеру буфера, как признак отсутствия страницы в нем.

Для представления категориальных данных применяется техника однозначного кодирования [4], при которой каждая категория представляет бинарным вектором, размер которого совпадает с числом категорий. Такой подход позволяет представить категории в виде дискретных типов без введения отношения порядка над ними.

Входные данные поступают на вход полносвязной нейронной сети с одним скрытым слоем [5]. Результатом работы сети является вектор признаков размерностью $d_e$.

**Кодировщик страниц в буфере.** Кодировщик страниц в буфере работает аналогично кодировщику запроса доступа. Единственным отличием является другой набор обучаемых весов и входных параметров. На вход нейронной сети помимо атрибутов каждой страницы также поступает закодированное представление индекса в буфере. Для кодирования индекса сначала применяется техника однозначного кодирования, а затем преобразование его в вектор размерности $d_f$ путем умножения на обучающиеся матрицы смежности.

**Кодировщик истории обращения к страницам.** Для запоминания истории запросов к буферу использована рекуррентная нейронная сеть LSTM [6]. На вход сети поступает вектор из кодировщика запросов доступа страницы.

Результатом работы сети является вектор $h^t$ размера $d_h$, который описывает историю обращений.

Сеть LSTM состоит из четырех компонентов. Ключевой из них – состояние ячейки, которая переходит между повторяющимися модулями сети, подвергаясь преобразованиям. Три других компонента отвечают за забывание прошлого состояния ячейки, обновление состояния на основе входных данных и выхода из прошлого модуля, а также за получение выходного значения из текущего блока.

Первый компонент нужен для определения того, какую часть информации можно выбросить из состояния ячейки. На вход к нему поступают входные данные в текущий блок и выходной вектор из прошлого модуля. На выходе при помощи сигмоидного фильтра для каждого значения в состоянии ячейки вычисляется число от 0 до 1, после чего происходит поточечное умножение чисел в ячейке на полученных из фильтра забывания значения.

Задача следующего компонента – определить какая новая информация будет сохранена в ячейке. Для этого сначала при помощи сигмоидного входного фильтра определяются значения, которые будут сохранены в ячейке, а затем с использованием слоя гиперболического тангенса вычисляются новые значения кандидатов на попадание в ячейку. После этих операций выполняется поточечное суммирование элементов в ячейке с полученными кандидатами.

Задача последнего компонента – определить, какая информация будет на выходе из текущего модуля. Для этого используется поточечное умножение текущего состояния ячейки, пропущенного через фильтр гиперболического тангенса, и входных данных, объединенных с выходом из прошлого модуля и прошедших через сигмоидный фильтр.

**Модуль выбора страницы для замещения.** Для принятия решения о том, какая страница будет вытеснена из буфера используется еще одна полносвязная нейронная сеть. На вход сети поступают выходной вектор из кодировщика истории обращения к страницам и список выходных векторов из кодировщика страниц в буфере. Результатом работы сети является вектором размером, совпадающим с количеством элементов в кэше, где каждый элемент показывает вероятность того, что та или иная страница должна быть замещена. Для замещения будет выбрана страница с наибольшей вероятностью.

## 1.4 Получение обучающей выборки

Для создания обучающей выборки необходимо получить последовательность обращений к страницам в буфере. Имея такую последовательность, каждый раз, когда требуется исключить страницу из буфера, можно выбрать кандидата при помощи оптимального алгоритма.

Такая последовательность была получена путем модификации исходного кода PostgreSQL и добавления логгирования в функцию ReadBufferExtended, которая вызывается каждый раз, когда необходимо прочитать страницу из буфера.

Для получения информации о свойствах запрашиваемого отношения используется поле rd_rel структуры Relation, которая передается в функцию в качестве входного аргумента. Тип физического хранилища, номер блока в файле и метод чтения страницы также являются входными аргументами функции. Для проверки того, является ли отношение временным, анализируется поле backend у структуры RelFileLocatorBackend. Для проверки того, загружена ли страница в буфер, анализируется возвращаемое значение из текущей реализации функции замещения страниц.

## 1.5 Детализированная IDEF-0

Детализированная IDEF-0 диаграмма представлена на рисунке 1.1.

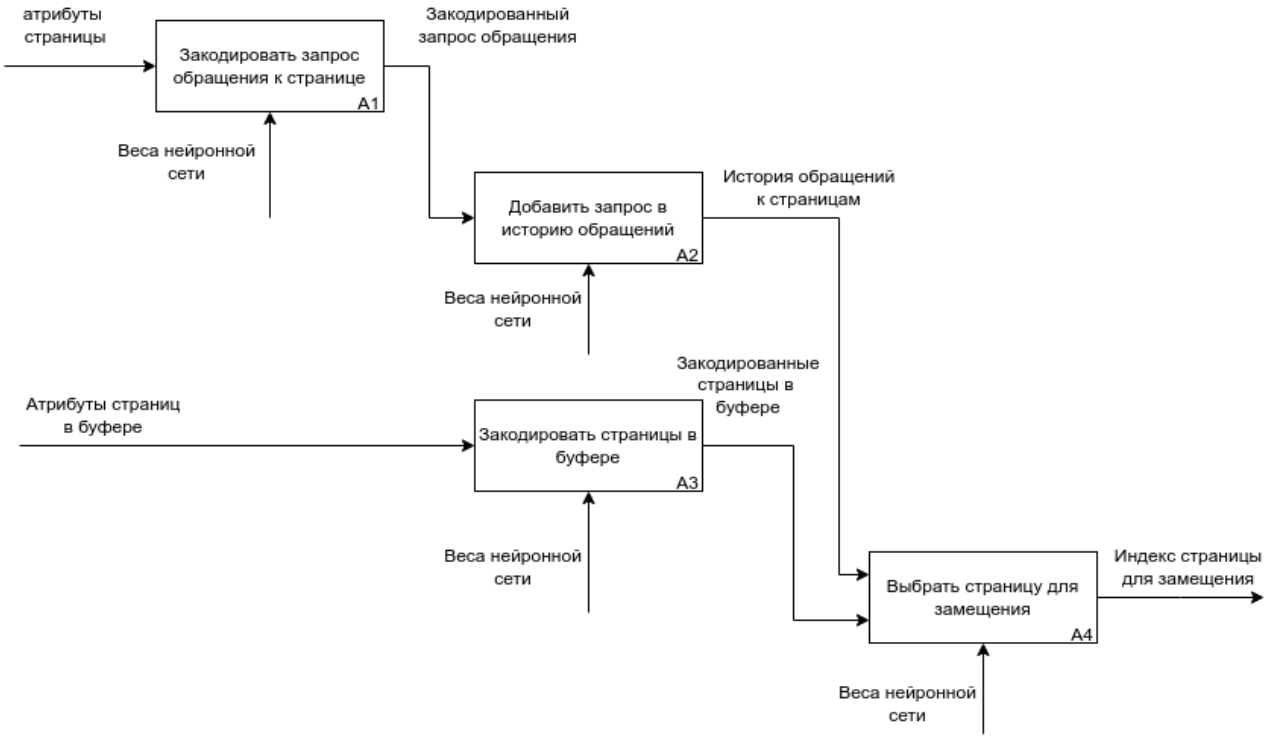


Рисунок 1.1 – Детализированная IDEF-0 диаграмма

Запрос обращения к странице проходит через кодировщик запросов и попадает в блок, который отвечает за историю обращений. Текущая история обращений к страницам и закодированные страницы, которые в данных момент находятся в буфере, попадают на вход блока, который отвечает за выбор страницы для замещений.

## 1.6 Структура ПО

Структура ПО приведена на рисунке 1.2. Каждый блок, представленный в детализированной idef0 диаграмме, является отдельным модулем в структуре программного обеспечения. Также отдельный модуль отвечает за получения обучающей выборки при моделировании типовой нагрузки на СУБД.

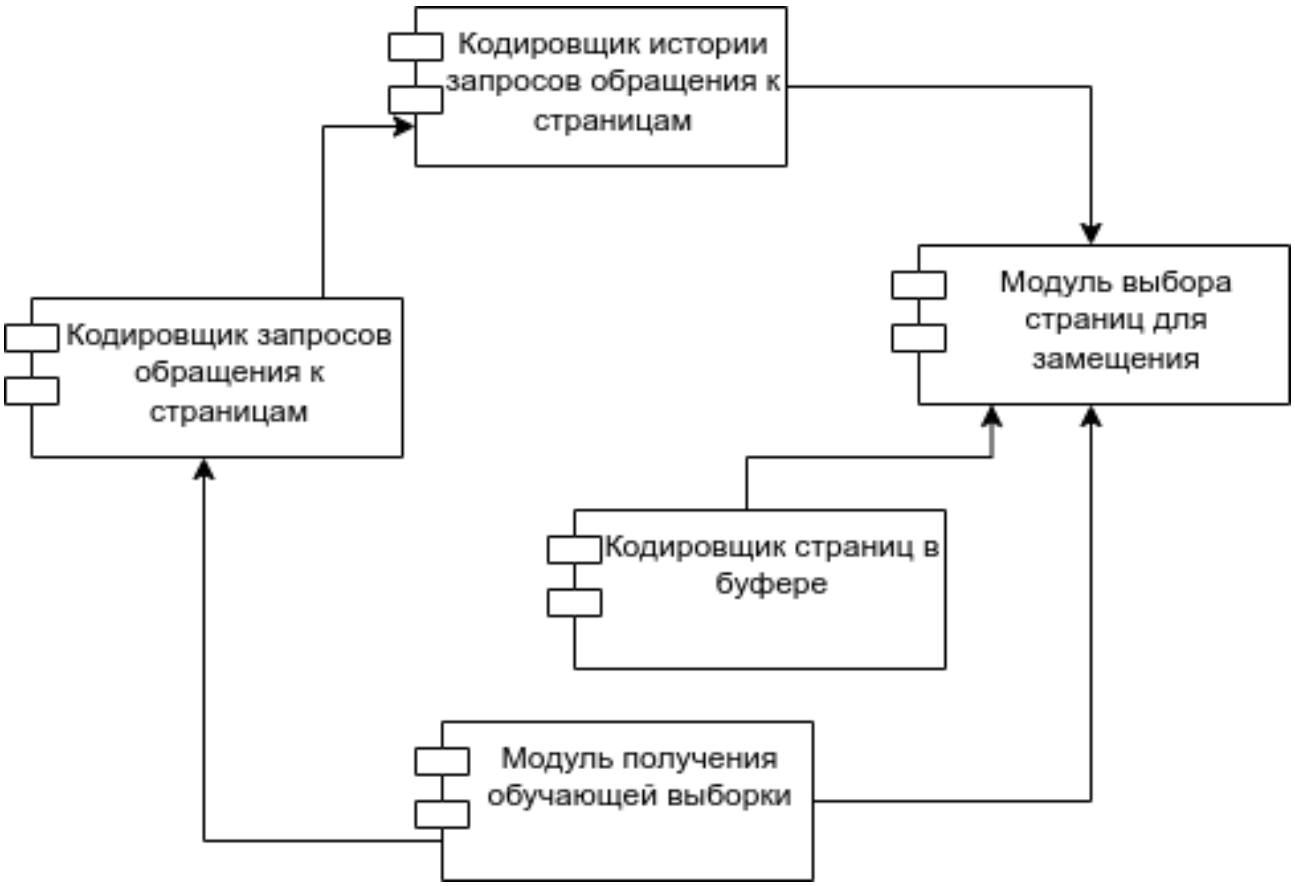


Рисунок 1.2 – Структура ПО

# ЗАКЛЮЧЕНИЕ

В данной научно-исследовательской работе были:

— изложены особенности разрабатываемого метода;

— описаны основные этапы метода в виде детализированной idef-0 диаграммы;

— спроектирована структуру программного обеспечения.

В рамках работы были выполнены все поставленные задачи. Цель работы была достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. *Peiquan Y.* Learned buffer replacement for database systems // Proceedings of the 2022 5th International Conference on Data Storage and Data Engineering. — 2022. — С. 18—25.
2. *Shaik B.* PostgreSQL Configuration: Best Practices for Performance and Security. — Apress, 2020.
3. *Chen X.* DeepBM: A Deep Learning-based Dynamic Page Replacement Policy. — 2019.
4. *Казенников А. О.* Сравнительный анализ методов сокращения линейных моделей машинного обучения для задач автоматической обработки текстов. — 2012.
5. *Митина О. А.* Перцептрон в задачах бинарной классификации // Национальная ассоциация ученых. — 2021. — № 66—1. — С. 39—44.
6. *Zhang J.* A review of recurrent neural networks: LSTM cells and network architectures // Neural computation. — 2019. — Т. 31, № 7. — С. 1235—1270.