Санкт-Петербургский Государственный Университет

Математическое обеспечение и администрирование информационных систем

Зиннатулин Тимур Раифович

Интеграция файловых хранилищ ZFS с СУБД PostgreSQL

Производственная практика

Научный руководитель:
к. ф.-м. н., доцент кафедры системного программирования Луцив Д. В.

Санкт-Петербург
2024

Оглавление

Введение

Количество информации, которая генерируется и хранится в компьютерах пользователей, постоянно возрастает: по оценкам некоторых источников [4] объем информации, доступной в сети Интернет, измеряется в зеттабайтах. Эти объемы информации требуют круглосуточного хранения и постоянной обработки. Одним из основных способов хранения и работы с данными являются системы управления и обработки информации (СУБД). Учитывая различный характер видов информации, которыми оперируют пользователи, и способов их обработки, существует большое количество различных СУБД, но наибольшее распространение и развитие получили реляционные СУБД.

Современные реляционные СУБД имеют широкие возможности для представления, хранения и обработки разнородных данных пользователей. Существенная доля этих данных вносится в СУБД в неструктурированном (не реляционном) формате. Яркими примерами систем хранения и обработки неструктурированной информации являются DLP-системы и торговые системы. DLP-системы [1] отвечают за безопасность информационного обмена и сигнализируют о возможных утечках. Такие системы оперируют большим количеством данных пользователей, в число которых входят: документы, изображения, обмен информацией через файловые хранилища и другие действия пользователей. Различные торговые системы часто хранят тысячи и сотни тысяч изображений товаров и должны иметь возможность оперативно работать с этими данными. Организация хранения неструктурированных бинарных данных в БД является сложной и актуальной задачей, которую разработчики должны решать при создании информационной системы.

Распространенным способом хранения неструктурированной информации в компьютерных системах являются файлы. СУБД также имеют множество различных решений, которые позволяют разработчикам реализовывать программные компоненты для связи информации в базах данных и файловых хранилищах. Можно выделить такие решения,

как наличие в СУБД специализированных типов данных и модулей для хранения произвольных бинарных данных, возможность интеграции с внешними объектными хранилищами, поддержка в СУБД стандарта SQL/MED и другие решения. Представленные подходы не лишены недостатков, их использование усложняет процесс разработки информационных систем и связанных с ним задач хранения и обработки произвольных неструктурированных данных пользователей. Поэтому задача реализации эффективных и простых инструментов для разработчика, позволяющих оперировать большими объемами неструктурированной информации (изображения, файлы, документы и т.п.) в базах данных является актуальной. Обязательным условием реализации этих инструментов должна быть поддержка основных гарантий транзакционных систем при хранении и обработке неструктурированных данных пользователя.

В работе предложен и реализован новый способ прямого доступа из СУБД к файлам пользователей, расположенных на подключаемом внешнем хранилище. В качестве СУБД в работе выбрана PostgreSQL, файлы пользователей размещаются в пулах файловой системы ZFS. Работа выполняется в команде из двух человек. В данной работе рассматривается верхний уровень всей системы - создание расширения для СУБД, которое позволит разрабатывать интерфейсы прямого доступа к файловым хранилищам ZFS на языке SQL.

1. Постановка цели и задач

Целью работы является создание расширения для СУБД PostgreSQL, позволяющего использовать функции языка SQL для разработки интерфейсов прямого доступа к файлам пользователей на файловых хранилищах ZFS.

Для достижения цели были поставлены следующие задачи:

- Рассмотреть существующие подходы к решению задачи работы с бинарными данными в PostgreSQL.
- Определить подход к реализации взаимодействия с файловой системой ZFS в PostgreSQL.
- Реализовать расширение PostgreSQL для взаимодействия с файловой системой ZFS.
- Провести тестирование полученного решения и сравнение с аналогами.

2. Обзор

В этом разделе будут рассмотрены методы работы с неструктурированными данными в СУБД PostgreSQL.

2.1. PostgreSQL Large Objects

В PostgreSQL есть возможность хранить бинарные данные с помощью так называемых "Больших объектов", или BLOB[1].

Postgres назначает каждому BLOB свой oid (4-байтовое целочисленное значение), разделяет его на куски по 2кБ и помещает в системную таблицу pg_largeobject. Механизм BLOB предоставляет потоковый доступ к бинарным данным и поддерживает гарантии транзакционности и версионирование данных. Однако этот механизм обладает низкой производительностью по сравнению с работой с данными в файловой системе, для работы с ним необходимо использовать интерфейс, отличающийся от стандартного.

2.2. bytea

В PostgreSQL для хранения бинарных данных также предусмотрен тип bytea. Данный тип является двоичной строкой переменной длины. Тип bytea использует механизм TOAST, который позволяет хранить в одной записи большие значения атрибутов. Так как размер записи одной таблицы не может превышать размер страницы (по умолчанию 8кб), большие значения атрибутов делятся на небольшие сегменты и помещаются в специальные таблицы pg_toast.

Данный механизм надежно работает и не требует специальных действий пользователя для настройки работы. Но вследствие того, что TOAST был разработан давно и стал частью ядра PostgreSQL, он плохо адаптирован к условиям огромных объемов данных и постоянной высокой нагрузки на СУБД. В частности, при вызове команд UPDATE и

[1]Binary Large Object

INSERT механизм TOAST дублирует все записи данных, которых касается команда, из-за чего при активной работе с этими данными таблицы TOAST быстро разрастаются и падает производительность.

2.3. Работа с метаданными файлов

Для работы с внешними данными для стандарта SQL был разработан SQL/MED [6]. Данное расширение стандарта предоставляет возможность работать с данными вне СУБД посредством хранения ссылок на объекты данных. Для этого в стандарт добавлен тип DATALINK, хранящий ссылку на объект.

Очевидным преимуществом данного подхода является то, что возможный объем хранимых данных не зависит от возможностей самой базы данных, но напрямую от возможностей системы, на которой хранятся эти данные. Недостатками же нужно назвать отсутствие контроля за синхронностью файлов в DATALINK и в системе: ссылки могут быть устаревшими, файлы могут меняться извне, и пользователь об этом узнает, только когда обратится по старой ссылке.

Данный стандарт полноценно реализован только в ограниченном числе СУБД, основным среди которых можно выделить IBM Db2. В рамках PostgreSQL данный стандарт реализован только частично.

2.3.1. Foreign Data Wrapper

В рамках ограниченной поддержки стандарта SQL/MED PostgreSQL предоставляет пользователям возможность как обращаться к таблицам из внешних баз данных, так и ко внешним объектам в целом (файлам, веб-сервисам и др.) с помощью Foreign Data Wrapper [5]. В частности, модуль postgres_fdw, являющийся частью стандартной сборки PostgreSQL, позволяет обращаться к данным на локальных и внешних серверах [2].

Механизм использования Foreign Data Wrapper в PostgreSQL стандартно выглядит следующим образом:

1. Создание расширения:

```
CREATE EXTENSION *fdw_extension*;
```

2. Указание параметров подключения к внешним данным:

```
CREATE SERVER *foreign_server*
    FOREIGN DATA WRAPPER *fdw_extension*
    OPTIONS (...);
```

3. Указание внешнего пользователя, к которому должен обращаться пользователь PostgreSQL при работе с внешними данными:

```
CREATE USER MAPPING FOR *user_name*
    SERVER *foreign_server*
    OPTIONS (USER ..., password ...);
```

4. Создание внешней таблицы, с которой будет работать PostgreSQL при обращении ко внешним данным:

```
CREATE FOREIGN TABLE *table_name*
    (
        ...
    )
    SERVER *foreign_server*
    OPTIONS (...);
```

Структура PostgreSQL позволяет реализовывать пользовательские модули Foreign Data Wrapper. На данный момент существует множество различных модулей, позволяющих обращаться к различным внешним базам данных (MySQL, Oracle, Redis, Neo4j), файлам и другим

данным [3]. С их помощью администратор базы данных получает возможность выделять структуру из неструктурированной информации и интегрировать ее в базу.

Среди модулей Foreign Data Wrapper, позволяющих обращаться к внешним файлам, стоит выделить file_fdw[2]. Этот модуль позволяет обращаться к данным в локальной файловой системе в формате, который может распознать команда COPY FROM (csv, text, binary). Также он работает только read-only, и не поддерживает изменение данных.

2.4. Внешние хранилища данных

В качестве распространенного решения задачи работы с неструктурированными данными необходимо отметить использование внешних хранилищ данных. Такие сервисы, как AWS S3, Google Cloud Storage, Yandex Object Storage и другие, предоставляют разработчикам возможность обращаться к неструктурированным объектам по их метаданным, в то время как сами объекты находятся в облачном хранилище.

Производительность данного решения зависит от производительности используемого хранилища, которые в свою очередь оптимизированы для работы с данными любого типа. В то же время при использовании данных решений разработчикам требуется прикладывать дополнительные усилия для сохранения целостности продукта.

[2]file_fdw: `https://www.postgresql.org/docs/current/file-fdw.html`

3. Предлагаемое решение

3.1. Описание решения

Для решения поставленных задач в работе предлагается следующий подход. Представим элементы файловой системы в виде реляционных таблиц определенной структуры, включающей поля как для хранения атрибутов файлов, так и для хранения их содержимого. Используя расширенные возможности языка SQL и механизм доступа к внешним источникам данных в PostgreSQL, реализуем прямой доступ из SQL к элементам файловой системы, скрывая для пользователя все внутренние сложности и предоставляя необходимые ACID-гарантии транзакционного доступа. Архитектурная диаграмма решения представлена на рис.1.

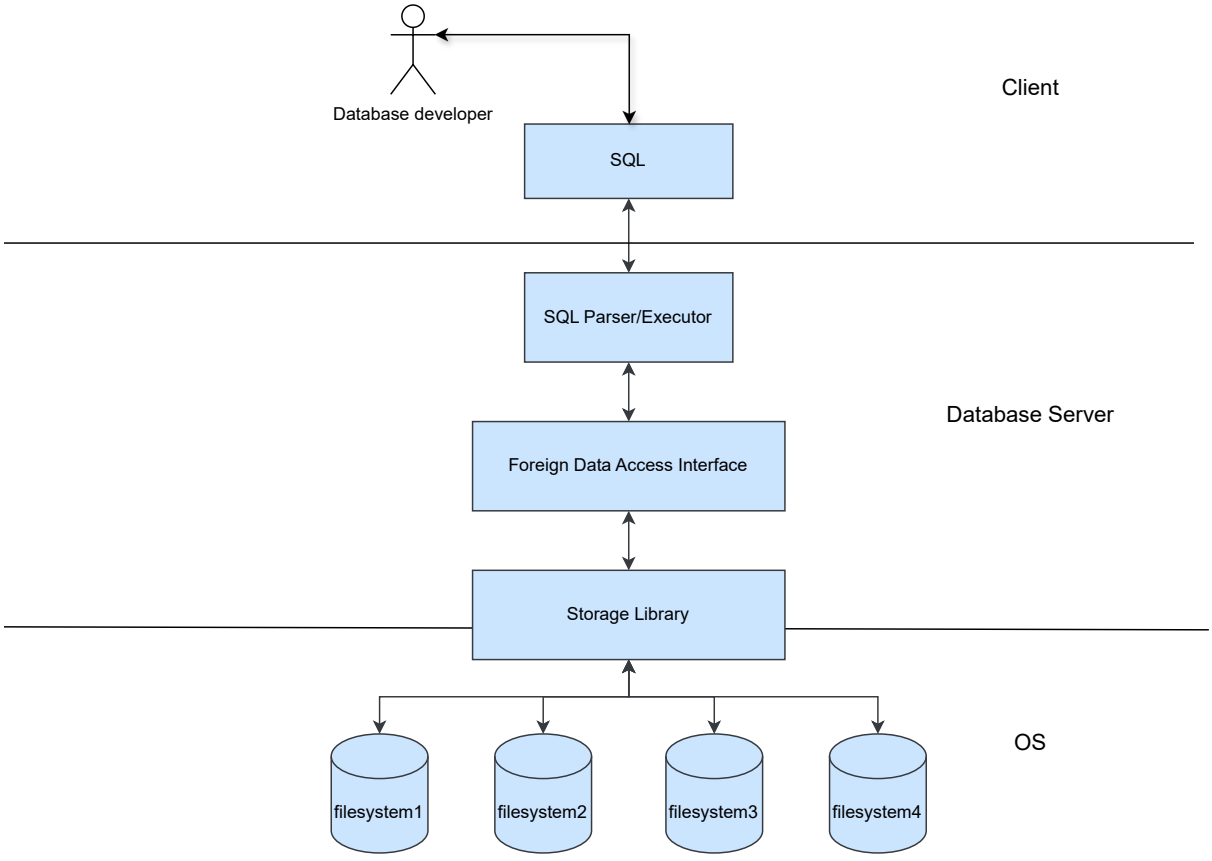

Рис. 1: Пример работы с реализованным расширением в PostgreSQL

В предлагаемом подходе пользователю предоставляется новый модуль PostgreSQL, реализующий интерфейс доступа к внешнему источ-

нику данных — файловой системе. Данный модуль создает новый SQL-объект вида “Обертка сторонних данных” (команда CREATE FOREIGN DATA WRAPPER). Данный объект позволяет организовать с помощью команд CREATE SERVER и CREATE FOREIGN TABLE подключение к внешнему источнику данных и представление внешнего источника данных в виде реляционных таблиц с поддержкой основных SQL-операций (SELECT, INSERT, UPDATE, DELETE) и выполнением ACID-гарантий. Внешний источник данных использует специальную библиотеку доступа к файловой системе, реализующую все необходимые низкоуровневые операции. К таким операциям относятся получение списков имен файлов и директорий, получение дополнительных атрибутов файлов (размер, время создания, время доступа и т.п.), доступ к содержимому файлов, а также создание новых файлов и директорий. Задача библиотеки — преобразовывать транзакционные операции над реляционным источником данных в конкретные файловые операции конкретной файловой системы.

В данной работе качестве файловой системы используется ZFS [7], так как она обладает следующими характеристиками:

- Открытый API для взаимодействия с файловой системой
- Поддержка механизма создания снапшотов для фиксирования консистентных состояний файлов с данными пользователя при параллельных операциях доступа к этим данным
- Поддержка механизма copy-on-write для выполнения транзакционных изменений в файлах

3.2. Расширение PostgreSQL для работы с ZFS

Опишем структуру и реализацию предлагаемого модуля обертки внешних данных для ZFS в PostgreSQL. Модуль создает FDW-объект и набор вспомогательных функций:

```
CREATE FUNCTION zfs_fdw_handler()
    RETURNS fdw_handler
```

```
    AS 'MODULE_PATHNAME'
    LANGUAGE C STRICT;

CREATE FUNCTION zfs_fdw_validator(text[], oid)
    RETURNS void
    AS 'MODULE_PATHNAME'
    LANGUAGE C STRICT;

CREATE FOREIGN DATA WRAPPER zfs_fdw
    HANDLER zfs_fdw_handler
    VALIDATOR zfs_fdw_validator;
```

Главная функция — обработчик, который является связующим интерфейсом между SQL Parser/Executor и данными пользователя. Функция валидации предназначена для проверки корректности опций, которые задаются пользователем в вызовах CREATE SERVER и CREATE FOREIGN TABLE. В данном решении для связки с файловой системы предлагается следующий набор опции: *pool_ name* — имя пула ZFS, *filesystem* — точка монтирования файловой системы в пуле ZFS. Значения данных опций передаются в библиотеку Storage Library для инициализации доступа к ZFS.

Работа со внешними данными в хранилищах ZFS посредством механизма Foreign Data Wrapper реализована на нескольких этапах. На этапе создания внешнего сервера задаются параметры работы с ZFS: имена пула и файловой системы в составе ZFS, с которыми непосредственно осуществляется работа. На этапе создания внешней таблицы задается представление данных файловой системы в виде отношения: имя файла, его размер, время создания и прочие параметры задаются как поля таблицы в SQL.

Для реализации Foreign Data Wrapper была реализована функция-обработчик, которая возвращает структуру с указателями на реализующие подпрограммы, которые вызываются планировщиком, исполнителем и служебными командами. Эти подпрограммы включают в себя в числе прочих:

- Функции для сканирования внешних таблиц

- Функции для обновления внешних таблиц
- Функции для EXPLAIN и ANALYZE

Среди функций, ответственных за сканирование внешних таблиц, выделяются функции, ответственные за планирование запроса:

- GetForeignRelSize: оценивает размер отношения во внешней таблице
- GetForeignPaths: формирует пути доступа для сканирования внешней таблицы
- GetForeignPlan: создает узел плана ForeignScan из выбранного планировщиком пути доступа для сканирования. Сформированный узел ForeignPlan является узлом дерева плана.

И функции, ответственные за исполнение запроса:

- BeginForeignScan: запускает сканирование таблицы по составленному плану
- IterateForeignScan: обрабатывает одну строку данных и возвращает ее в слоте таблицы кортежей. В данной функции из в рамках одного кортежа сторонних данных формируется строка таблицы
- ReScanForeignScan: перезапускает сканирование с начала
- EndForeignScan: заканчивает сканирование и высвобождает выделенные ресурсы

Данные функции были реализованы с использованием API для файловой системой ZFS. В рамках выполнения запроса к внешней таблице из ZFS запрашиваются данные о файлах, хранящихся в заданном пуле и файловой системе на момент последнего снапшота, и собираются в виде таблицы SQL.

На данный момент поддерживается выполнение запросов SELECT к файлам, находящимся в корневой директории файловой системы. Также реализовано создание файлов с определенным содержимым. Операции, которые меняют состояние файловой системы, вызывают создание нового снапшота, что позволяет поддерживать гарантии ACID.

Ниже представлена демонстрация работы созданного решения. Для демонстрации был создан пул ZFS с именем *mypool*, в котором была создана система *myfs*. Для данного примера Foreign Data Wrapper определяется следующим образом:

```
CREATE EXTENSION zfs_fdw;
CREATE SERVER zfs_server FOREIGN DATA WRAPPER zfs_fdw
    OPTIONS (pool_name 'mypool', filesystem 'myfs');

CREATE FOREIGN TABLE files (
    filename text,
    filesize bigint,
    creationtime text,
    accesstime text,
    modtime text,
    content_bytea bytea,
    content_text text
)
SERVER zfs_server;
```

В данной схеме создается внешняя таблица *files*, содержимое которой соответствует содержимому файловой системы, определенной для внешнего сервера *zfs_ server*. Пример работы с этой таблицей показан на рис. 2.

Основная часть работы по разработке расширения закончена. На данный момент в процессе реализация работы с файловой системой посредством функций INSERT, UPDATE и DELETE.

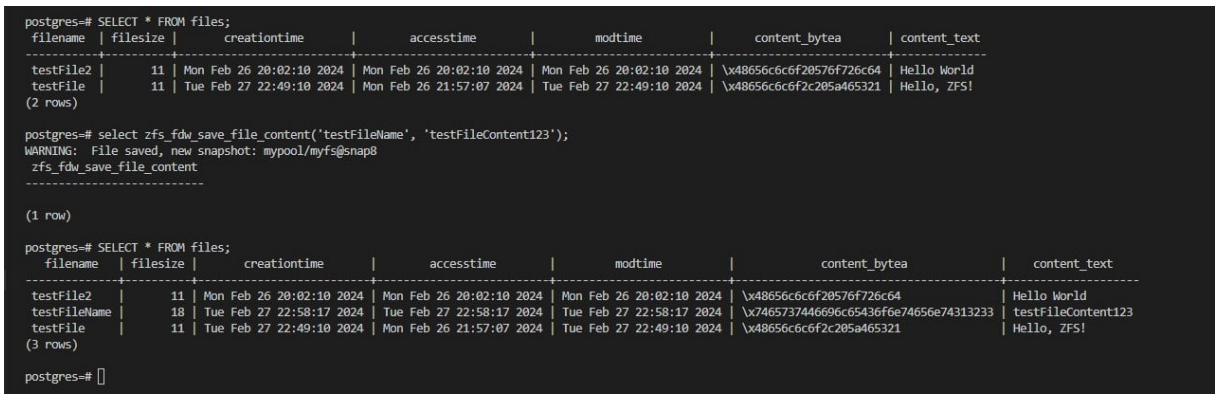

Рис. 2: Пример работы с реализованным расширением в PostgreSQL

4. Тестирование

Тестирование полученного решения будет проводиться по следующему плану:

- Модульное тестирование: будут написаны тесты на основные функции Foreign Data Wrapper с заглушками для функций API для ZFS.

- Интеграционное тестирование: будет подготовлен и запущен стенд с файловой системой ZFS, к которому подключится PostgreSQL с готовым расширением. Будет проверяться логика работы расширения с подключенным API для работы с файловой системой ZFS.

- Тестирование производительности: Будут проведены эксперименты по работе с бинарными файлами (сохранение, извлечение, удаление) при помощи встроенных механизмов PostgreSQL, таких как BLOB и bytea, и при помощи расширения. Будут замерены такие метрики как время выполнения запроса и количество транзакции в секунду.

Заключение

В результате работы над учебной практикой были выполнены следующие задачи:

- Рассмотрены существующие подходы к решения задачи версионирования бинарных данных в PostgreSQL.
- Определен подход к реализации взаимодействия с файловой системой ZFS в PostgreSQL.
- Частично реализовано расширение для взаимодействия с файловой системой ZFS.
- Намечен план тестирования и сравнения с аналогами разрабатываемого решения.

В процессе дальнейшей работы планируется:

- Завершить реализацию расширения PostgreSQL для взаимодействия с файловой системой ZFS.
- Провести тестирование полученного решения и сравнить с аналогами.

Код проекта закрыт и принадлежит компании ООО "Датаджайл".

Список литературы

[1] Arbel Lior. Data loss prevention: the business case // Computer Fraud Security. — 2015. — Vol. 2015, no. 5. — P. 13–16. — Access mode: https://www.sciencedirect.com/science/article/pii/S1361372315300373.

[2] Foreign data wrappers - PostgreSQL wiki. — https://www.postgresql.org/docs/current/postgres-fdw.html. — Accessed: 2023-04-04.

[3] Foreign data wrappers - PostgreSQL wiki. — https://wiki.postgresql.org/wiki/Foreign_data_wrappers. — Accessed: 2023-04-04.

[4] Krotov Vlad, Johnson Leigh. Big web data: Challenges related to data, technology, legality, and ethics // Business Horizons. — 2023. — Vol. 66, no. 4. — P. 481–491. — Access mode: https://www.sciencedirect.com/science/article/pii/S0007681322001252.

[5] Melton Jim. Chapter 5 - Foreign Servers and Foreign-Data Wrappers // Advanced SQL:1999 / Ed. by Jim Melton. — San Francisco : Morgan Kaufmann, 2003. — The Morgan Kaufmann Series in Data Management Systems. — P. 235–278. — Access mode: https://www.sciencedirect.com/science/article/pii/B9781558606777500067.

[6] SQL/MED: A Status Report / Jim Melton, Jan Eike Michels, Vanja Josifovski et al. // SIGMOD Rec. — 2002. — sep. — Vol. 31, no. 3. — P. 81–89. — Access mode: https://doi.org/10.1145/601858.601877.

[7] Меликов Георгий. ZFS: архитектура, особенности и отличия от других файловых систем // «Завтра облачно», журнал о цифровой трансформации от VK Cloud Solutions. — 2020. — https://mcs.mail.ru/blog/zfs-arhitektura-osobennosti-i-otlichija. — Accessed: 2023-04-04.