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
2023

Оглавление

Введение

Количество информации, которая генерируется и хранится в компьютерах пользователей, постоянно возрастает: текущий объем информации измеряется в зеттабайтах [4]. 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.

Больша́я часть данных, с которыми системы должны работать, являются неструктурированными. Одним из ярких примеров хранения неструктурированных данных являются системы, отвечающие за безопасность, или так называемые DLP-системы [1], которые отвечают за безопасность информационного обмена и сигнализируют о возможных утечках. Такие системы оперируют большим количеством данных пользователей, в число которых входят: документы, изображения, посылаемой электронной почтой, обмен через файловых хранилища и так далее. 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 нетривиальной задачей, которую разработчики должны решать при разработке информационной системы.

Работа с файлами в СУБД – распространенный вариант использования. Существует множество различных решений, который позволяют разработчикам реализовывать программные компоненты для связи информации в базах данных и файловых хранилищах. Среди таких решений можно выделить BLOB, механизм Large Object, bytea, SQL/MED. Каждый из механизмов обладает определенными недостатками, препятствующими полноценно использовать произвольные неструктурированные данные в реляционных базах данных. Поэтому задача реализации эффективных и простых инструментов для разработчика, поз-

воляющих оперировать большими объемами неструктурированной информации (изображения, файлы, документы и т.п.) в базах данных является актуальной. Также данные форматы работы с бинарными данными не позволяют эффективно реализовывать версионирование объектов базы данных, вследствие чего возникла идея использовать в решении данной проблемы файловую систему, позволяющую эффективно версионировать данные.

В рамках этой работы будет рассмотрена и реализована идея подключения к СУБД хранилища файлов на файловой системе ZFS с целью предоставить унифицированный способ работы с большим количеством бинарных файлов. Данная работа выполняется в команде из двух человек, и целью данной работы будет реализация расширения с интерфейсом взаимодействия с файловой системой ZFS на базе PostgreSQL.

1. Постановка цели и задач

Данная работа выполняется в рамках проекта, над которым работают два человека. Целью именно этой работы является реализация расширения для PostgreSQL, позволяющего управлять данными в файловой системе ZFS с реализованными возможностями транзакционности и версионирования, при помощи синтаксиса SQL.

Для достижения цели были поставлены следующие задачи:

- Изучить 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

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, однако он обладает низкой производительностью по сравнению с работой с данными в файловой системе, для работы с ним необходимо использовать интерфейс, отличающийся от стандартного, и данный механизм не предусмотрен для версионирования данных.

2.2. 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.

[1]Binary Large Object

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

3. Предлагаемое решение

3.1. Описание решения

В данной работе предлагается решить проблему хранения и версионирования бинарных данных в СУБД PostgreSQL следующим образом: заводится сетевое файловое хранилище с файловой системой, которая позволяет на её основе реализовать необходимый для отслеживания версий файлов функционал. Предполагается, что у сервера базы данных есть доступ к этому сетевому хранилищу.

На основе файловой системы на сетевом хранилище будет реализован API, позволяющий взаимодействовать с файлами через стандартный синтаксис SQL, то есть поддерживающий стандартные операции SELECT, INSERT, UPDATE и DELETE. Далее будет написано расширение для PostgreSQL, использующее этот API и позволяющее сохранять и версионировать бинарные файлы из СУБД. Благодаря этому будет возможность взаимодействовать с файлами стандартными средствами СУБД, не заботясь о том, как эти файлы на самом деле хранятся.

В качестве файловой системы была выбрана ZFS [7], так как она обладает следующими атрибутами:

- Открытый API для взаимодействия с файловой системой
- Поддержка создания снапшотов для реализации алгоритма версионирования данных
- Поддержка транзакционности через механизм copy-on-write

3.2. Расширение PostgreSQL для работы с ZFS

Для работы PostgreSQL с файловой системой ZFS начата реализация расширения, представляющего собой Foreign Data Wrapper для объектов файловой системы. Оно использует API, реализованный в работе, связанной с реализацией алгоритма версионирования, и позволять

пользователю обращаться к данным в ZFS–хранилище с помощью SQL-команд как к данным во внешней таблице.

Для реализации Foreign Data Wrapper необходимо было реализовать функцию-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

На данный момент реализованы основные функции для работы расширения с файловой системой ZFS посредством API, реализуемого в работе Александра Семёнова. Также зафиксировано взаимодействие между API для ZFS и разрабатываемым расширением.

4. Тестирование

Тестирование полученного решения предполагается проводить по следующему плану:

- Модульное тестирование: будут написаны регресс-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:

- Изучены существующие подходы к решения задачи версионирования бинарных данных в PostgreSQL.
- Определен подход к реализации взаимодействия с файловой системой ZFS в PostgreSQL.
- Начата реализация расширения 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.

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.