Санкт-Петербургский Государственный Университет

Математическое обеспечение и администрирование информационных систем

Зиннатулин Тимур Раифович

Интерфейс работы с файловой системой ZFS в СУБД PostgreSQL

Учебная практика

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

Санкт-Петербург
2023

Оглавление

Введение

Сегодня сложно представить себе организацию, деятельность которой обходится без информационных систем. Одной из основных составляющих любой информационной системы являются данные. Для их хранения и управления используются различные технологии, например, базы данных. Для настраивания и администрирования баз данных применяется набор программных средств, называемый системой управления базами данных (СУБД).

Одной из самых популярных объектно-реляционных СУБД является PostgreSQL, которая широко используется в различных областях, включая финансы, науку и образование, а также имеет открытый исходный код, поддерживает репликации, хранимые процедуры и прочие полезные функции, а также совместима с набором требований ACID. Одним из требований ACID является атомарность, которая гарантирует, что транзакция либо будет выполнена полностью, либо не выполняется совсем. Это позволяет сохранять согласованность данных в базе, что может быть критично для многих информационных систем, например, для банковских или систем, взаимодействующих с рынком ценных бумаг.

Для хранения данных внутри базы в PostgreSQL предусмотрено множество типов, таких как integer, text, boolean и прочие. Эти типы данных позволяют эффективно хранить и обрабатывать структурированные данные, однако они не подходят для неструктурированных данных, например, бинарных. Основное отличие этих типов данных в том, что неструктурированные данные не соответствуют заранее определенной структуре данных, из-за этого подходы к хранению и обработке такого рода информации должны отличаться от соответствующих подходов работы со структурированными данными. Примерами бинарных данных, которые может быть полезно хранить информационной системе, могут быть PDF-документы, картинки, архивы, электронные письма и т.п.

Если бинарные данные имеют небольшой размер, то часто для их

версионирования заводят отдельную таблицу, в которой хранят метаданные, такие как дата последнего изменения файла и ссылка на конкретную версию файла. Для такого подхода необходимо хранить все версии файлов отдельными записями. Проблемы возникают, когда появляется необходимость хранить большие файлы, так как из-за множества их версий разрастается сама база данных, а также это будет сказываться на её производительности.

В данной работе будут рассмотрены подходы к версионированию бинарных файлов внутри СУБД PostgreSQL и предложено расширение для PostgreSQL, позволяющее взаимодействовать с данными в файловой системе, поддерживающей версионирование и транзакционность.

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, поддерживающей транзакционность и версионирование, при помощи синтаксиса SQL.

Для достижения цели были поставлены следующие задачи:

- Изучить существующие подходы к решению задачи версионирования бинарных данных в PostgreSQL.
- Определить подход к реализации взаимодействия с выбранной файловой системой в PostgreSQL.
- Реализовать расширение PostgreSQL для взаимодействия с файловой системой.
- 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. Foreign Data Wrapper

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. В частности, модуль postgres_fdw, являющийся частью стандартной сборки PostgreSQL, позволяет обращаться к данным на локальных и внешних серверах [1].

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*
```

[1]Binary Large Object

```
    FOREIGN DATA WRAPPER *fdw_extension*
    OPTIONS (...);
```

3. Указание внешнего пользователя, к которому должен обращаться пользователь PostgreSQL при работе с внещними 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 [2]. С их помощью администратор базы данных получает возможность выделять структуру из неструктурированной информации и интегрировать ее в базу.

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 [3], 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 будет реализовано расширение, представляющее собой Foreign Data Wrapper для объектов файловой системы. Оно будет использовать API, реализованный в работе, связанной с реализацией алгоритма версионирования, и позво-

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

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 выбранной файловой системой в PostgreSQL.

В процессе дальнейшей работы планируется:

- Реализовать расширение PostgreSQL для взаимодействия с выбранной файловой системой.
- Провести тестирование полученного решения и сравнить с аналогами.

Список литературы

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

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

[3] 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.