Лабораторная работа №03: Модель многопользовательского доступа: MVCC

Цель работы: Изучить принципы многоверсионного управления конкурентным доступом (MVCC) в PostgreSQL. Получить практические навыки наблюдения за работой MVCC, анализа версий строк, снимков данных и уровней изоляции транзакций. Освоить использование расширений и системных представлений для исследования внутренней структуры данных.

Стек технологий:

- **ОС:** Xubuntu 24.04 64-bit (предустановлена в виртуальной машине)
- **СУБД:** PostgreSQL 16
- **Утилиты:** `psql`
- **Ключевые понятия:** MVCC, Транзакция, Уровни изоляции (Read Committed, Repeatable Read), XMIN, XMAX, CTID, Снимок данных (Snapshot), Аномалии параллелизма (Фантомное чтение, Неповторяемое чтение), `pageinspect`, `ON_ERROR_ROLLBACK`, Экспорт/импорт снимков

Теоретическая часть (краткое содержание): MVCC (Multiversion Concurrency Control) — механизм, позволяющий нескольким транзакциям работать с одними и теми же данными одновременно, минимизируя блокировки. Каждая транзакция видит согласованный «снимок» данных на момент своего начала.

- **Версии строк:** При изменении строки создается ее новая версия. Старая версия остается в таблице до очистки.
- **Системные поля:**
  - `xmin` — идентификатор транзакции, создавшей версию строки.
  - `xmax` — идентификатор транзакции, удалившей версию строки (или заблокировавшей ее для обновления).
  - `ctid` — физическое расположение версии строки в таблице (номер страницы и позиции в ней).
- **Уровни изоляции:** Определяют, какие аномалии параллелизма допустимы:
  - `Read Committed` (По умолчанию): Виден только зафиксированный данные. Возможны неповторяемое чтение и фантомное чтение.
  - `Repeatable Read`: Гарантирует, что данные, прочитанные в транзакции, не изменятся. Предотвращает неповторяемое чтение, возможны фантомы.
  - `Serializable`: Самый строгий уровень, предотвращает все аномалии.
- **Снимок данных (Snapshot):** Набор идентификаторов транзакций, активных на момент начала текущей транзакции. Определяет, какие версии строк видимы текущей транзакции.

Задание на практическую реализацию:

Модуль 1: Уровни изоляции и аномалии

1. **Read Committed vs Удаление:**
   - Создайте таблицу `iso_test (id INT, data TEXT)` и вставьте одну строку.
   - В сеансе 1 начните транзакцию с уровнем `READ COMMITTED` и выполните `SELECT * FROM iso_test;`.
   - В сеансе 2 удалите строку и зафиксируйте изменения (`DELETE ...; COMMIT;`).

    - В сеансе 1 выполните тот же `SELECT` повторно. Сколько строк увидите? Завершите транзакцию в сеансе 1.
2. **Repeatable Read vs Удаление:**
    - Повторите предыдущий эксперимент, но в сеансе 1 начните транзакцию с `BEGIN ISOLATION LEVEL REPEATABLE READ;`.
    - Объясните разницу в результатах между двумя уровнями изоляции.
3. **Создание таблицы в транзакции:**
    - В сеансе 1 начните транзакцию и создайте новую таблицу `new_table`, вставьте в нее строку. Не фиксируйте.
    - В сеансе 2 выполните `SELECT * FROM new_table;`. Что произойдет?
    - Зафиксируйте транзакцию в сеансе 1. Повторите запрос в сеансе 2.
    - Повторите процесс, но вместо фиксации откатите транзакцию в сеансе 1. Что изменилось?
4. **Блокировка DDL:**
    - В сеансе 1 начните транзакцию и выполните `SELECT * FROM iso_test;` (даже если таблица пуста).
    - Попытайтесь в сеансе 2 выполнить `DROP TABLE iso_test;`. Получится ли? Объясните, почему.

Модуль 2: Фантомное чтение и снимки

1. **Фантомное чтение (Read Committed):**
    - Создайте пустую таблицу `phantom_test (id INT)`.
    - Продемонстрируйте на уровне `Read Committed`, что аномалия "фантомное чтение" не предотвращается (вставка новых строк в другом сеансе становится видимой).
2. **Невидимость удалений (Repeatable Read):**
    - В сеансе 1 начните транзакцию с уровнем `Repeatable Read` (пока без запросов).
    - В сеансе 2 удалите все строки из `phantom_test` и зафиксируйте.
    - В сеансе 1 выполните `SELECT * FROM phantom_test;`. Увидятся ли удаленные строки?
    - Выполните в сеансе 1 запрос `SELECT * FROM pg_database;` (не касаясь `phantom_test`). Повлияет ли это на видимость строк в `phantom_test` при последующем запросе?
3. **Транзакционность DDL:** Убедитесь, что `DROP TABLE` является транзакционной операцией (можно откатить).

Модуль 3: Версии строк и pageinspect

1. **Жизненный цикл строки:**
    - Создайте таблицу `version_test (id INT)`. Вставьте одну строку.
    - Дважды обновите эту строку (`UPDATE ...`), а затем удалите ее (`DELETE`).
    - Используя расширение `pageinspect` (`heap_page_items`), определите, сколько версий строк находится сейчас в таблице. Объясните их состояние по полям `t_xmin`, `t_xmax`, `t_ctid`.
2. **Анализ системной таблицы:**
    - Определите, в какой странице (блоке) находится строка в `pg_class`, описывающая саму таблицу `pg_class`.
    - Используя `pageinspect`, подсчитайте количество актуальных (видимых) версий строк в этой странице.
3. **ON_ERROR_ROLLBACK:** Включите в `psql` параметр `ON_ERROR_ROLLBACK`. Создайте ситуацию с ошибкой в транзакции и убедитесь, что этот режим использует точки сохранения (SAVEPOINT), позволяя продолжить работу транзакции после ошибки.

Модуль 4: Снимки данных (Snapshots)

1. **Видимость удаленной строки:**
   - Воспроизведите ситуацию, при которой одна транзакция (A) видит строку, а другая (B), начавшаяся позже, — уже нет (строка удалена и зафиксирована после начала A, но до начала B).
   - Используйте функции `pg_current_snapshot()` и `pg_snapshot_xip(pg_current_snapshot())` для анализа снимков обеих транзакций.
   - Изучите значения `xmin` и `xmax` удаленной строки. Объясните разницу в видимости на основе анализа снимков.
2. **Снимки в функциях:**
   - Создайте функцию `STABLE`, возвращающую данные из таблицы.
   - Исследуйте, какой снимок данных используется для запроса внутри этой функции при разных уровнях изоляции (`Read Committed` и `Repeatable Read`). Повторите для функции `VOLATILE`.
   - Объясните разницу в поведении.
3. **Экспорт/импорт снимка:**
   - В транзакции 1 (уровень `Repeatable Read`) экспортируйте снимок данных с помощью `pg_export_snapshot()`.
   - В транзакции 2 измените какие-либо данные и зафиксируйте.
   - В транзакции 3 импортируйте снимок из транзакции 1 (`SET TRANSACTION SNAPSHOT '...';`). Убедитесь, что в транзакции 3 видны данные в состоянии на момент экспорта снимка, до изменений из транзакции 2.

Требования к оформлению и отчету:

1. **Полнота:** Должны быть выполнены ВСЕ задания из всех модулей.
2. **Скрипты:** Предоставить полную последовательность выполненных SQL-команд для каждого сеанса.
3. **Выводы:** Приложить вывод команд, подтверждающих каждое наблюдение (результаты SELECT, вывод `pageinspect`, снимки).
4. **Отчет:** Подробный отчет с ответами на все вопросы ("объясните", "почему", "проверьте", "сколько") из формулировок заданий. Отчет должен демонстрировать глубокое понимание работы MVCC.

Критерии оценки:

- **Удовлетворительно:** Полностью выполнены Модули 1 и 2. Даны ответы на основные вопросы.
- **Хорошо:** Дополнительно полностью выполнен Модуль 3. Отчет содержит развернутые объяснения по всем пунктам этих модулей.
- **Отлично:** Полностью выполнены ВСЕ модули (1, 2, 3, 4). Отчет демонстрирует исчерпывающее понимание MVCC, уровней изоляции, работы снимков и внутреннего устройства страниц.

Рекомендуемая литература:

1. Курс "Администрирование PostgreSQL 16. Базовый курс": https://postgrespro.ru/education/courses/DBA1
2. Курс "Администрирование PostgreSQL 16. Настройка и мониторинг": https://postgrespro.ru/education/courses/DBA2

3. Документация: Транзакции: https://postgrespro.ru/docs/postgresql/16/tutorial-transactions
4. Документация: Уровни изоляции: https://postgrespro.ru/docs/postgresql/16/transaction-iso
5. Документация: Системные поля: https://postgrespro.ru/docs/postgresql/16/ddl-system-columns
6. Документация: Функции снимков: https://postgrespro.ru/docs/postgresql/16/functions-admin#FUNCTIONS-SNAPSHOT-SYNCHRONIZATION
7. Документация: Расширение pageinspect: https://postgrespro.ru/docs/postgresql/16/pageinspect
8. Книги по PostgreSQL: https://postgrespro.ru/education/books