МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«**МИРЭА − Российский технологический университет**»

РТУ МИРЭА

Институт комплексной безопасности и цифровых технологий (ИКБ)
Кафедра КБ-14 «Цифровые технологии обработки данных»

Направление подготовки

09.03.02 Информационные системы и технологии

Практическая работа №2

Задание 1:

1. Создайте новую базу данных в PostgreSQL включающие две таблицы: "accounts" и "transactions". Таблица "accounts" должна содержать следующие поля: id (уникальный идентификатор), name (имя), balance (баланс). Таблица "transactions" должна содержать следующие поля: id (уникальный идентификатор), account_id (ссылка на id в таблице "accounts"), amount (сумма).
2. Проведите проверку что PostgreSQL не допускается аномалия **грязного чтения**, объясните почему.
3. Проверьте, что на уровне изоляции Read Committed не предотвращается аномалия фантомного чтения.
4. Начните транзакцию с уровнем изоляции Repeatable Read(и пока не выполняйте в ней никаких команд). В другом сеансе удалите строку

и зафиксируйте изменения. Видна ли строка в открытой транзакции? Что изменится, если в начале транзакции выполнить запрос,но не обращаться в нем ни к одной таблице?

5. Напишите функцию, которая позволяет выполнить перевод средств с одного счета на другой, используя транзакции. Функция должна использовать уровень изоляции транзакции "Serializable". Протестируйте функцию с использованием нескольких параллельных сеансов, чтобы убедиться, что переводы не могут быть выполнены дважды.
6. Начните транзакцию Repeatable Read и выполните какой-нибудь запрос. В другом сеансе создайте таблицу. Видно ли в первой транзакции описание таблицы в системном каталоге? Можно ли в ней прочитать строки таблицы?
7. Убедитесь, что команда DROP TABLE транзакционна.

Задание 2:

1. Установите расширение pageinspect.
2. Создать базу данных с именем versions_db. Создать таблицу users со следующими полями:
    - id: уникальный идентификатор пользователя (integer, primary key, auto-increment).
    - username: имя пользователя (varchar(255)).
    - email: электронный адрес пользователя (varchar(255)).
    - version: версия строки (integer).
3. Создать триггер, который будет автоматически увеличивать версию строки при любом обновлении.
4. Вставить в таблицу users строку с различными данными а затем обновите.
5. При помощи следующего запроса:

```
SELECT '(0,'||lp||')' AS ctid,
    t_xmin as xmin,
    t_xmax as xmax,
    CASE WHEN (t_infomask & 256) > 0  THEN 't' END AS xmin_c,
    CASE WHEN (t_infomask & 512) > 0  THEN 't' END AS xmin_a,
    CASE WHEN (t_infomask & 1024) > 0 THEN 't' END AS xmax_c,
    CASE WHEN (t_infomask & 2048) > 0 THEN 't' END AS xmax_a
FROM heap_page_items(get_raw_page('users',0))
ORDER BY lp;
```

Где,

- **ctid** является ссылкой на следующую, более новую, версию той же строки. У самой новой, актуальной, версии строки ctid ссылается на саму эту версию
- **xmin** и **xmax** определяют видимость данной версии строки в терминах начального и конечного номеров транзакций.
- **xmin_c, xmin_a, xmax_c, xmax_a** содержит ряд битов, определяющих свойства данной версии

Выведите информацию о версиях строк, узнав сколько версий строк щас находится в таблице и сравнить их с атрибутом (version)

6. Опустошим таблицу при помощи `TRUNCATE;`
7. Начините транзакцию и вставьте новую строку и узнайте номер текущий транзакции (это можно сделать при помощи след команды: `INSERT INTO users(...) VALUES (...) RETURNING *, ctid, xmin, xmax;`
8. Поставьте точку сохранения и добавьте новую строку использовав команду из пункта 7.
9. Откатимся к точке сохранения и добавим новую строчку аналогично 7 и 8 пункту.
10. Выведите сведения о версиях строк.

Задание 3:

1. Создать таблицу t с полями id(integer) и name (char(2000)) с параметром filfactor = 75%.
2. Создать индекс над полем t(name)
3. Установите расширение pageinspect.
4. Создать представление которое будет включать в себя информацию о версиях строк при помощью след запроса:

```
CREATE VIEW t_v AS
SELECT '(0,'||lp||')' AS ctid,
    CASE lp_flags
      WHEN 0 THEN 'unused'
      WHEN 1 THEN 'normal'
      WHEN 2 THEN 'redirect to '||lp_off
      WHEN 3 THEN 'dead'
    END AS state,
    t_xmin || CASE
      WHEN (t_infomask & 256) > 0 THEN ' (c)'
```

```
    WHEN (t_infomask & 512) > 0 THEN ' (a)'
    ELSE ''
  END AS xmin,
  t_xmax || CASE
    WHEN (t_infomask & 1024) > 0 THEN ' (c)'
    WHEN (t_infomask & 2048) > 0 THEN ' (a)'
    ELSE ''
  END AS xmax,
  CASE WHEN (t_infomask2 & 16384) > 0 THEN 't' END AS hhu,
  CASE WHEN (t_infomask2 & 32768) > 0 THEN 't' END AS hot,
  t_ctid
FROM heap_page_items(get_raw_page('t',0))
ORDER BY lp;
```

- флаг Heap Hot Updated показывает, что надо идти по цепочке ctid,
- флаг Heap Only Tuple показывает, что на данную версию строки нет ссылок из индексов.

5. Спроецировать ситуацию в таблице t, при которой произойдет внутристраничная очистка без участия HOT-обновлений.
6. После воспроизвести ситуацию но уже с HOT-обновлением

★ Использовать фактор заполнения на 80% и на 50%, указать в отчете в чем разница между разными факторами

Отчет оформить в формате doc (docx) или pdf и выслать на проверку

Вспомогательная литература:

1) Документация PostgreSQL 13 https://postgrespro.ru/docs/postgresql/13/sql
2) Postgres. Первое знакомство. П.Лузанов, Е.Рогов., И.Лёвшин https://drive.google.com/file/d/1qP3T0MXwvKE2X0NjpTg_r0AzyX-aSaQg/view?usp=sharing
3) PostgreSQL. Основы языка SQL. Е.Моргунов https://drive.google.com/file/d/1ROwk4yvymZImpDcFoY9Q1LR4a7bBDs95/view?usp=sharing

4) SQL. Сборник Рецептов. Энтони Молинаро
https://drive.google.com/file/d/1SQlmajycyVggyW6wZO2gMUW0C6g7KlaR/view?usp=sharing
5) Управление данным. И.Иванова
https://drive.google.com/file/d/1HS59TzRC5tsnev0r-SQhH85q52vmM-v4/view?usp=sharing
6) * И.Задворьев. Язык PL\SQL
https://drive.google.com/file/d/1OSCUEgnbefoccsb8JjlWSIEJgAVk51MO/view?usp=sharing