**Лабораторная работа 5**

**Задание:**

1. Чем отличаются типы данных char и varchar в части их хранения в странице таблицы? Какой тип выгоднее использовать и почему?

Посмотреть, как хранятся разные типы данных (int, bool, date, json) в странице (для этого нужно создать новые таблицы с другими типами данных)

2. Узнать текущий уровень изоляции транзакций. Как вы его определили?

(Подсказка: для этого нужно использовать параллельные транзакции)

3. Меняя уровни транзакций, посмотреть, какие проблемы параллельного доступа возможны на каждом уровне.

***Теория к заданию 1***

Инструмент в PostgreSQL, которые помогают посмотреть содержимое страницы: pageinspect

Как его включить и использовать?

1. Зайти под пользователем с админскими правами (postgres)
2. Выполнить команду для включения дополнительного модуля:

CREATE EXTENSION pageinspect;

Теперь будут доступны функции для исследования страниц БД

https://postgrespro.ru/docs/postgresql/12/pageinspect

Содержимое страницы можно посмотреть командой:

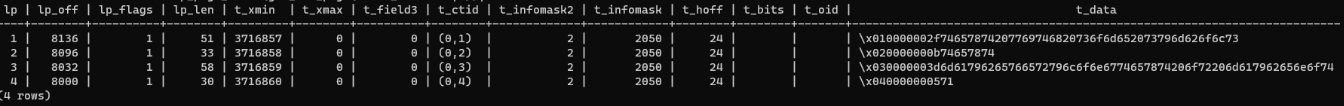
SELECT \* FROM heap\_page\_items(get\_raw\_page('test',0));

*test* - название таблицы,

*0* - номер страницы, информация о которой будет выводиться

Что можно увидеть в результате?

(пример вывода)



*lp line pointer* - номер указателя на запись внутри страницы

*lp\_off* - смещение до кортежа от начала страницы (в байтах)

*lp\_flags* - состояние line poiner (0 - не используется, 1 - используется, 3 - HOT

redirect, 3 - dead)

*lp\_len* - размер кортежа в байтах

*t\_xmin* - номер транзакции, которая добавила данные

*t\_xmax* - номер транзакции, которая удалила данные

*t\_field3* - id команды, которая добавила или удалила данные, или id команды

vacuum full

*t\_ctid* - id кортежа (номер страницы, номер кортежа в странице)

*t\_infomask2* - различные флаги\*

*t\_infomask* - различные флаги\*

*t\_hoff* - размер заголовка строки

*t\_bits* - bitmap of NULLs (указание, что в поле существует NULL значение)

t\_oid

*t\_data* - данные (сами данные или указатели на них)

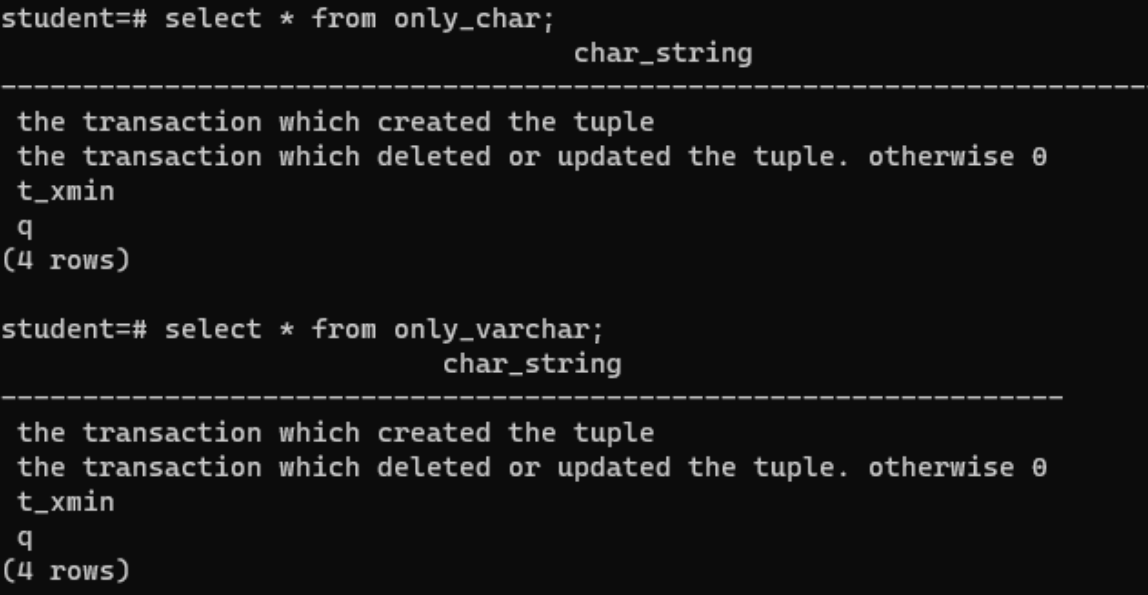
\* при желании можно посмотреть здесь

<https://github.com/postgres/postgres/blob/master/src/include/access/htup_details.h>

***Подготовка к заданию 1***

1. Создать две таблицы, в одной поле типа char(N), в другой - varchar(N).

2. Заполнить таблицы одинаковыми данными, 5-10 строк разной длины будет достаточно. Добавляемые строки должны быть разной длины. Пример заполненных таблиц:



3. Используя команду, представленную ниже, посмотреть, как устроена страница таблицы:

SELECT \* FROM heap\_page\_items(get\_raw\_page('test',0));

(здесь нужно использовать нужные данные)

***Теория к заданиям 2-3***

Пример транзакции:

BEGIN;

UPDATE accounts SET balance = balance -

100.00 WHERE name = 'Alice';

UPDATE accounts SET balance = balance +

100.00 WHERE name = 'Bob';

COMMIT;

- COMMIT - подтверждение транзакции

- ROLLBACK - откат транзакции

В дополнение к обычным данным таблицы можно посмотреть метаданные. Пример запроса:

SELECT ctid, xmin, xmax, \* FROM test;

Метаданные в строке:

• ctid - физическое расположение строки в таблице (номер страницы, номер кортежа в странице)

• xmin - id транзакции, добавившей эту строку

• xmax - id транзакции, удалившей строку, или 0 для неудалённой версии строки.

• cmin - номер команды (начиная с нуля) внутри транзакции, добавившей строку

• cmax - номер команды в удаляющей транзакции или ноль

\* на самом деле не только они, есть ещё разные флаги и т. д., их можно было увидеть в задании 1

**Уровни изоляции транзакций**:

1. Read uncommited (Чтение незафиксированных данных)

2. Read committed (Чтение зафиксированных данных)

3. Repeatable read (Повторяемое чтение)

4. Serializable (Сериализуемость)

**Проблемы параллельного доступа:**

1. «Грязное» чтение: транзакция читает данные, записанные параллельной незавершённой транзакцией.

2. Неповторяемое чтение: транзакция повторно читает те же данные, что и раньше, и обнаруживает, что они были изменены другой транзакцией (которая завершилась после первого чтения).

3. Фантомное чтение: транзакция повторно выполняет запрос, возвращающий набор строк для некоторого условия, и обнаруживает, что набор строк, удовлетворяющих условию, изменился из-за транзакции, завершившейся за это время.

4. Аномалия сериализации: результат успешной фиксации группы транзакций оказывается несогласованным при всевозможных вариантах исполнения этих транзакций по очереди.

**Изменение уровня изоляции транзакции**

SET TRANSACTION режим\_транзакции

или

SET SESSION CHARACTERISTICS AS TRANSACTION режим\_транзакции

где режим\_транзакции = ISOLATION LEVEL {READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }

Пример:

SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL REPEATABLE READ;

• Команда SET TRANSACTION устанавливает характеристики текущей транзакции. На последующие транзакции она не влияет

• SET SESSION CHARACTERISTICS устанавливает характеристики транзакции по умолчанию для последующих транзакций в рамках сеанса. Заданные по умолчанию характеристики затем можно переопределить для отдельных транзакций командой SET TRANSACTION.

https://postgrespro.ru/docs/postgrespro/12/sql-set-transaction