Transaction

Транзакция — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена:

- целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций
- не выполнена вообще, и тогда она не должна произвести никакого эффекта

Пример: connectors/postgres/tran.py

```
import asyncio
import datetime
import os
import typing

import asyncpg


async def get_connection_by_dsn():
    """
    POSTGRES_DSN = postgres://user:password@host:port/database
    """
    return await asyncpg.connect(os.getenv("POSTGRES_DSN"))


async def insert_many(conn) -> asyncpg.Record:
    async with conn.transaction():
        for i in range(10):
            name = f'name{i}'
            await conn.execute(
                "insert into users (first_name, last_name, is_tutor, created) values ($1, $2, $3, $4)",
                name, name, True, datetime.datetime.now()
            )


async def transaction_with_error(conn):
    async with conn.transaction():
        await insert_many(conn)
        1/0


async def select_many(conn) -> typing.AsyncIterator[asyncpg.Record]:
    async with conn.transaction():
        cursor = await conn.cursor("select * from users")
        chunk = await cursor.fetch(5)
        while chunk:
            for item in chunk:
                yield item
            chunk = await cursor.fetch(5)


async def run():
    conn = await get_connection_by_dsn()
    await insert_many(conn)
    try:
        await transaction_with_error(conn)
    except ZeroDivisionError:
        pass
    async for r in select_many(conn):
        print('select_many', r)


asyncio.run(run())
```

Запустить пример можно в mercury:

```
python3 connectors/postgres/tran.py
```

Разберемся, что тут происходит.

1. insert_many открывает транзакцию с помощью асинхронного контекстного менеджера async with conn.transaction() и выполняет 10 вставок в базу данных.

2. transaction_with_error открывает транзакцию и вызывает insert_many, а затем специально вызывает исключение (1/0). Посмотрите: транзакция открывается 2 раза, но на самом деле открывается 1 раз, самый первый. Дальше все операции делаются в рамках самой верхней транзакции. При этом, когда произойдет исключение, все данные, которые записала функция insert_many, откатятся.

3. select_many. Транзакции не всегда используются для вставок и обновлений данных. Они нужны также для получения результатов запроса частями. В функции открывается транзакция и создается cursor. По своему назначению это указатель на данные.
Если мы вызовем cursor.fetch(5), мы вернем 5 записей из результата запроса и сдвинем указатель на 5 позиций. Метод fetch будет возвращать записи, пока данные не закончились в cursor. Иначе вернет пустой список.