Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ
Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Базы данных

Тема «Репетиционная база»
Лабораторная работа №3
Разработка NoSQL базы данных и спецификаций прикладной программы

Студент: А.С. Бригадир
Преподаватель: С.С. Силич

МИНСК 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная лабораторная работа посвящена разработке конвертера из реляционной базы данных PostgreSQL в набор баз данных BerkeleyDB (NoSQL).

Работа основана на результатах лабораторной работы №1 (разработка серверной части и реляционной схемы «Репетиционная база»). В рамках текущего задания реализован консольный конвертер, который:

– подключается к существующей базе данных;
– для каждой таблицы создает отдельный файл .db;
– использует первичный ключ как ключ записи;
– сериализует остальные поля в JSON;
– для таблиц без PK формирует составной ключ.

1 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

1.1 Описание формата хранения данных

В таблице 1.1 приведено описание формата хранения данных, конвертированных из таблиц PostgreSQL в базы данных BerkeleyDB. В полях таблицы приведены названия столбцов таблиц PostgreSQL.

Таблица 1.1 – Формат хранения данных в BerkeleyDB

Таблица PostgreSQL	BerkeleyDB	
	Ключ	Значение (JSON)
booking	id	{"time", "cost", "creation_date", "status", "number_of_people", "id_room", "id_user", "duration"}
equipment	id	{"name", "type", "brand", "model", "condition", "id_rehearsal_point"}
service_booking	id_srv_id_bk	{}
equipment_booking	id_eq_id_bk	{}
rehearsal_points	id	{"rating", "contact_number", "schedule", "name", "address"}
rooms	id	{"name", "air_conditioner", "price", "recording_support", "area", "id_rehearsal_point"}
service	id	{"name", "price", "type", "requirements", "id_rehearsal_point"}
staff	id	{"full_name", "address", "experience", "phone", "age", "id_rehearsal_point"}
users	id	{"full_name", "phone", "email", "registration_date"}

Конвертер работает по следующему алгоритму:

1. Подключение к базе данных PostgreSQL.
2. Получение из схемы `main` списка таблиц, а также для каждой таблицы – списка столбцов, первичного ключа (или составного ключа) и всех строк данных.
3. Создание файлов BerkeleyDB (по одному на каждую таблицу) и заполнение их записями: ключ – значение первичного (или составного) ключа, значение – JSON-объект с остальными полями.
4. Закрытие открытых соединений с базами данных.

1.2 Описание необходимых запросов к PostgreSQL

Для извлечения информации из PostgreSQL в конвертере используются

следующие запросы:

1. Запрос для получения имен всех таблиц схемы `main`:

```
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'main' AND table_type = 'BASE TABLE'
ORDER BY table_name
```

2. Запрос для получения названий столбцов таблицы `table`:

```
SELECT column_name
FROM information_schema.columns
WHERE table_schema = 'main' AND table_name = 'table'
ORDER BY ordinal_position
```

3. Запрос для получения первичного ключа таблицы `table`:

```
SELECT kcu.column_name
FROM information_schema.table_constraints tc
JOIN information_schema.key_column_usage kcu
  ON tc.constraint_name = kcu.constraint_name
WHERE tc.constraint_type = 'PRIMARY KEY'
  AND tc.table_schema = 'main' AND tc.table_name = 'table'
ORDER BY kcu.ordinal_position
```

4. Запрос для получения всех данных таблицы `table`:

```
SELECT {columns} FROM main.table
```

2 РЕЗУЛЬТАТЫ РАБОТЫ

В качестве примера работы конверта, на рисунках 1.1 и 1.2 приведены результат просмотра файлов staff.db и service.db:

Рисунок 1.1 – Содержимое staff.db

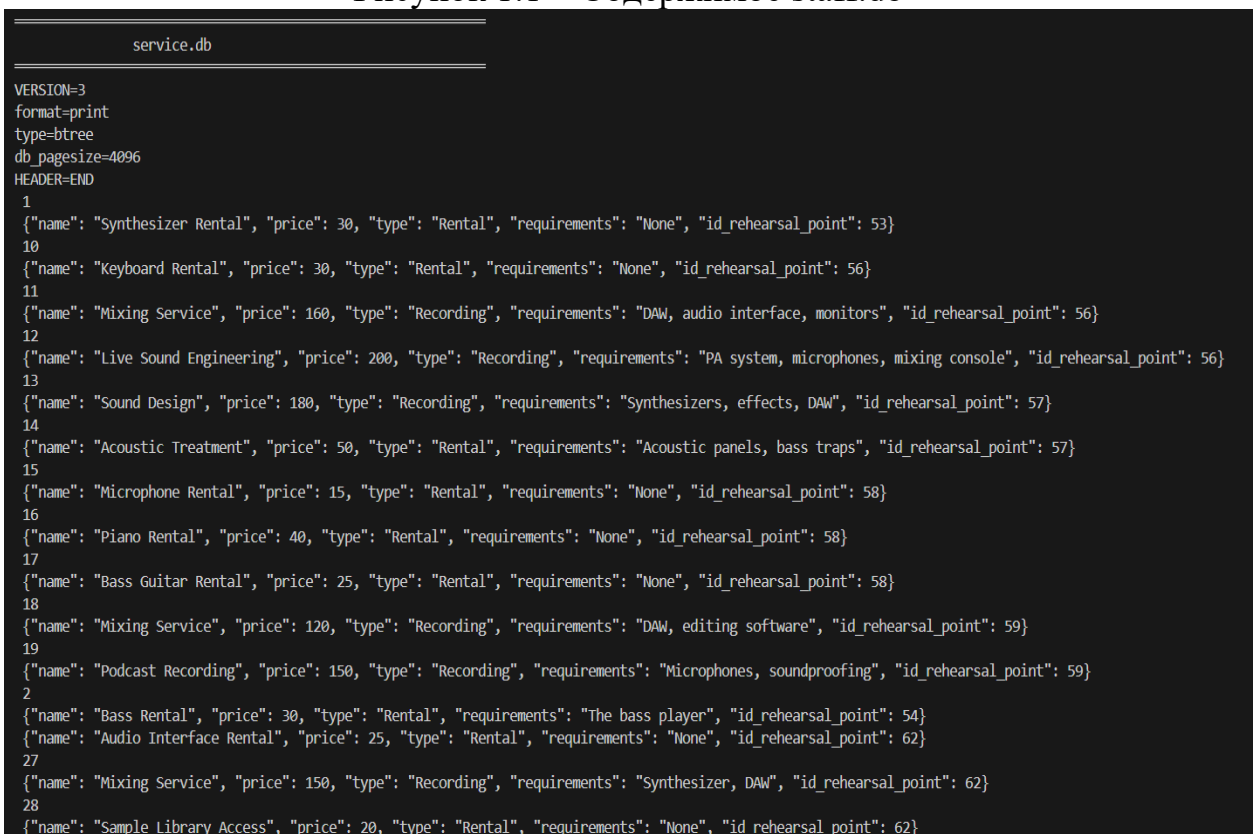

Рисунок 1.2 – Содержимое service.db

Реализованы все этапы преобразования: подключение к локальной СУБД через `host.docker.internal`, получение метаданных таблиц, выполнение выборки данных, формирование ключей и значений в формате JSON, а также запись в отдельные .db файлы. Для таблиц без первичного ключа (`equipment_booking`, `service_booking`) корректно сформированы составные ключи. Все операции выполняются в Docker-контейнере, что

исключает необходимость установки BerkeleyDB на хост-систему.

По завершении работы создано 9 независимых баз данных BerkeleyDB, сохранённых в папке berkeley_dbs/. Для проверки и просмотра результата реализован batch-файл view.bat, использующий утилиту db_dump, что позволяет просматривать содержимое любой базы одной командой.

Код конвертера и view.bat приведен в приложении А.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы был успешно реализован конвертер из реляционной базы данных PostgreSQL в набор баз данных BerkeleyDB. На основе существующей базы были проанализированы все таблицы, определены первичные и составные ключи, а также структура хранения данных в NoSQL-формате. Конвертер реализован на языке Python с использованием библиотеки psycopg2 для взаимодействия с PostgreSQL и berkeleydb для записи в BerkeleyDB, что обеспечило полную автономность и переносимость решения.

ПРИЛОЖЕНИЕ А

Листинг кода

Файл converter.py:

```
001 import psycopg2
002 import json
003 from berkeleydb import db
004 import os
005 import sys
006 # === НАСТРОЙКИ ===
007 PG_CONFIG = {
008     'dbname': os.getenv('PG_DB', 'repbase'),
009     'user': os.getenv('PG_USER', 'postgres'),
010     'password': os.getenv('PG_PASSWORD', ''),
011     'host': os.getenv('PG_HOST', 'localhost'),
012     'port': os.getenv('PG_PORT', 5432)
013 }
014 SCHEMA = os.getenv('PG_SCHEMA', 'main')
015 OUTPUT_DIR = '/output'
016 COMPOSITE_TABLES = ['equipment_booking', 'service_booking']
017 os.makedirs(OUTPUT_DIR, exist_ok=True)
018 def log(msg):
019     print(msg)
020     sys.stdout.flush()
021 def get_tables(conn):
022     cur = conn.cursor()
023     cur.execute(f"""
024         SELECT table_name
025         FROM information_schema.tables
026         WHERE table_schema = %s AND table_type = 'BASE TABLE'
027         ORDER BY table_name
028     """, (SCHEMA,))
029     return [row[0] for row in cur.fetchall()]
030 def get_columns(conn, table_name):
031     cur = conn.cursor()
032     cur.execute(f"""
033         SELECT column_name
034         FROM information_schema.columns
035         WHERE table_schema = %s AND table_name = %s
036         ORDER BY ordinal_position
037     """, (SCHEMA, table_name))
038     return [row[0] for row in cur.fetchall()]
039 def get_primary_key_columns(conn, table_name):
040     cur = conn.cursor()
041     cur.execute(f"""
042         SELECT kcu.column_name
043         FROM information_schema.table_constraints tc
044         JOIN information_schema.key_column_usage kcu
045           ON tc.constraint_name = kcu.constraint_name
046         WHERE tc.constraint_type = 'PRIMARY KEY'
047           AND tc.table_schema = %s AND tc.table_name = %s
048         ORDER BY kcu.ordinal_position
049     """, (SCHEMA, table_name))
050     return [row[0] for row in cur.fetchall()]
051 def convert_table(conn, table_name):
052     log(f"Конвертируем: {table_name}")
053     columns = get_columns(conn, table_name)
054     pk_cols = get_primary_key_columns(conn, table_name)
055     # Если нет PK → используем все колонки (для equipment_booking,
service_booking)
```

```
    if not pk_cols:
        pk_cols = ['id_equipment', 'id_booking'] if 'equipment' in
table_name else ['id_service', 'id_booking']
        log(f"  → Нет PK, используем составной ключ: {pk_cols}")
    db_path = os.path.join(OUTPUT_DIR, f"{table_name}.db")
    bdb = db.DB()
    bdb.open(db_path, None, db.DB_BTREE, db.DB_CREATE)
    cur = conn.cursor()
    cols_str = ', '.join(columns)
    cur.execute(f"SELECT {cols_str} FROM {SCHEMA}.{table_name}")
    rows = cur.fetchall()
    for row in rows:
        row_dict = dict(zip(columns, row))
        # Формируем ключ
        if len(pk_cols) == 1:
            key = str(row_dict[pk_cols[0]]).encode('utf-8')
        else:
            key = '_'.join(str(row_dict[col]) for col in
pk_cols).encode('utf-8')
        # JSON без ключевых полей
        value_dict = {k: v for k, v in row_dict.items() if k not in
pk_cols}
        value = json.dumps(value_dict, ensure_ascii=False,
default=str).encode('utf-8')
        bdb.put(key, value)
    bdb.close()
    log(f"  → {db_path} ({len(rows)} записей)")
def main():
    log("Запуск конвертера PostgreSQL (схема main) → BerkeleyDB")
    try:
        conn = psycopg2.connect(**PG_CONFIG)
        conn.cursor().execute(f"SET search_path TO {SCHEMA}")
        log(f"Подключено к БД: {PG_CONFIG['dbname']}, схема:
{SCHEMA}")
    except Exception as e:
        log(f"Ошибка подключения: {e}")
        return
    try:
        tables = get_tables(conn)
        log(f"Найдено таблиц: {tables}")
        for table in tables:
            convert_table(conn, table)
        log("Конвертация завершена!")
        log(f"Файлы сохранены в: {OUTPUT_DIR}")
    except Exception as e:
        log(f"Ошибка: {e}")
    finally:
        conn.close()
if __name__ == "__main__":
    main()
```

Файл view.bat:

```
@echo off
chcp 65001 >nul
echo.
echo ╔═══════════════════════════════════════════════════════╗
echo ║              ПРОСМОТР ВСЕХ BERKELEYDB БАЗ             ║
echo ╚═══════════════════════════════════════════════════════╝
echo.
:: ВКЛЮЧАЕМ ЗАДЕРЖКУ ПЕРЕМЕННЫХ
setlocal enabledelayedexpansion
:: Автоопределение папки проекта
set "PROJECT_DIR=%~dp0"
```

```
012 set "DB_DIR=%PROJECT_DIR%berkeley_dbs"
013 set "IMAGE=lab2-converter:latest"
014 :: Проверка папки
015 if not exist "%DB_DIR%" (
016     echo [ОШИБКА] Папка %DB_DIR% не найдена!
017     echo Создайте её и запустите: docker compose up --build
018     pause
019     exit /b 1
020 )
021 :: Список таблиц
022 set "TABLES=booking equipment equipment_booking rehearsal_points
rooms service service_booking staff users"
023 echo [ИНФО] Образ: %IMAGE%
024 echo [ИНФО] Папка: %DB_DIR%
025 echo.
026 :: Перебор таблиц
027 for %%t in (%TABLES%) do (
028     set "DB_FILE=%DB_DIR%\%%t.db"
029     if exist "!DB_FILE!" (
030         echo.
031         echo
═══════════════════════════════════════════════════════════
032         echo                %%t.db
033         echo
═══════════════════════════════════════════════════════════
034         docker run --rm -v "%DB_DIR%:/output" %IMAGE% db_dump -p
/output/%%t.db
035         if errorlevel 1 (
036             echo [ОШИБКА] Не удалось прочитать %%t.db
037         )
038     ) else (
039         echo.
040         echo [ПРЕДУПРЕЖДЕНИЕ] Файл %%t.db НЕ НАЙДЕН
041     )
042    echo.
043 )
044 echo ═══════════════════════════════════════════════════════════
045 echo                     ГОТОВО! Все таблицы выведены.
046 echo ═══════════════════════════════════════════════════════════
047 echo.
048 pause
049 endlocal
```