

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Базы данных

ОТЧЁТ
по лабораторной работе №3
на тему
РАЗРАБОТКА NOSQL БАЗЫ ДАННЫХ И СПЕЦИФИКАЦИЙ
ПРИКЛАДНОЙ ПРОГРАММЫ

Студент

М.С. Патюпин

Преподаватель

С.С. Силич

Минск 2025

СОДЕРЖАНИЕ

Введение.....	3
1 Технические требования.....	4
2 Разработка конвертора.....	5
Заключение	9

ВВЕДЕНИЕ

Данная лабораторная работа посвящена разработке NoSQL базы данных на основе BerkeleyDB и соответствующей прикладной программы, с целью освоения принципов работы с данными в распределенных системах. Актуальность работы обусловлена увеличением объемов данных и необходимостью адаптации приложений к новым формам хранения информации, где традиционные реляционные базы данных не всегда могут обеспечить требуемую производительность и масштабируемость.

В ходе выполнения лабораторной работы нам необходимо:

- 1 Изучить прикладной интерфейс СУБД BerkeleyDB.
- 2 Разработать конвертер для преобразования данных из базы данных PostgreSQL в набор баз данных BerkeleyDB.
- 3 Определить требования к прикладной программе, обеспечивающей эффективную работу с созданной NoSQL базой данных.

Лабораторная работа поможет закрепить теоретические знания о структуре и особенностях работы NoSQL систем, а также предоставить практические навыки в разработке и настройке приложений для работы с новыми форматами хранения данных.

1 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Для выполнения лабораторной работы были сформулированы следующие технические требования, касающиеся создания и интеграции базы данных PostgreSQL с BerkeleyDB. Проект направлен на разработку консольного приложения, которое эффективно осуществляет конвертацию данных между двумя системами хранения.

Технические требования создание баз данных BerkeleyDB:

1 Для каждой таблицы, существующей в базе данных PostgreSQL, необходимо создать соответствующую базу данных BerkeleyDB. Это обеспечит терабайтную прочность и высокую доступность данных, которые невозможно достичь только с помощью PostgreSQL.

2 Следует обратить внимание на правильное именование баз данных BerkeleyDB, чтобы они соответствовали названиям таблиц в PostgreSQL. Например, если существует таблица под названием users, то и база данных BerkeleyDB должна именоваться аналогично, что облегчит дальнейшую интеграцию и понимание структуры данных.

3 В качестве ключей в базах данных BerkeleyDB должны использоваться первичные ключи таблиц PostgreSQL. Это позволит обеспечить уникальность и целостность данных, а также упростит операции поиска и модификации записей.

4 При выборе первичных ключей необходимо убедиться, что они действительно уникальны и не будут изменяться в процессе эксплуатации системы, так как это может привести к ошибкам при обращении к данным в BerkeleyDB.

5 В качестве значений необходимо использовать сериализованные в формате JSON значения всех столбцов записи. Это позволит сохранить структуру данных и легко манипулировать ими в будущем.

Технические требования конвертора:

1 Конвертор должен представлять из себя консольное приложение, которое выполняет конвертацию и завершает свою работу после успешного завершения всех операций. Приложение должно быть простым в использовании, с чётким и понятным интерфейсом для взаимодействия.

2 Необходимо предусмотреть информирование пользователя о ходе выполнения операций, включая отображение статуса конвертации, возможных предупреждений и ошибок. Это поможет в процессе отладки и обеспечит пользователю уверенность в том, что процесс конвертации проходит успешно.

2 РАЗРАБОТКА КОНВЕРТОРА

В таблице 2.1 приведено описание формата хранения данных, конвертированных из таблиц PostgreSQL в базы данных BerkeleyDB. В полях таблицы приведены названия столбцов таблиц PostgreSQL.

Таблица 2.1 – Формат хранения данных в BerkeleyDB

Таблица PostgreSQL	Ключ в BerkeleyDB	Значение (JSON)
Order	orderid	{"status", "totalcost", "numberofguests", "orderdatetime", "visitorid", "employeeid"}
dish	dishid	{"name", "weight", "nutritionalvalue", "cost", "category"}
dishingredient	dishid_ingredientid	{"dishid", "ingredientid", "quantity"}
employee	employeeid	{"fullname", "phonenumber", "email", "hiredate", "contractenddate"}
employeeposition	employeeid_positionid	{"employeeid", "positionid"}
ingredient	ingredientid	{"name", "unitofmeasurement", "instock", "expirationdate", "supplierid"}
orderdish	orderid_dishid	{"orderid", "dishid", "count"}
position	positionid	{"name", "salary", "workschedule", "responsibilities"}
supplier	supplierid	{"companyname", "contactinformation", "reliabilityrating", "productcategory"}
visitor	visitorid	{"name", "birthdate", "phonenumber", "preferences"}

Конвертор работает по следующему алгоритму:

- 1 Подключение к базе данных PostgreSQL.
- 2 Получение метаданных о таблицах (названия, структура).
- 3 Получение содержимого таблиц.
- 4 Создание соответствующих баз данных BerkeleyDB.
- 5 Заполнение баз данных BerkeleyDB преобразованными данными.
- 6 Закрытие соединений с базами данных.

Код конвертора:

```
import psycopg2
import json
import bsddb3
from psycopg2.extras import RealDictCursor

DB_CONFIG = {
    "dbname": "Liso",
    "user": "postgres",
    "password": "",
    "host": "localhost",
    "port": "5432"
}

TABLES = {
    "Order": "orderid",
    "dish": "dishid",
    "dishingredient": ["dishid", "ingredientid"],
    "employee": "employeeid",
    "employeeposition": ["employeeid", "positionid"],
    "ingredient": "ingredientid",
    "orderdish": ["orderid", "dishid"],
    "position": "positionid",
    "supplier": "supplierid",
    "visitor": "visitorid"
}

def convert_postgres_to_berkeleydb():
    """Основная функция конвертации данных из PostgreSQL в
    BerkeleyDB"""

    try:
        conn = psycopg2.connect(**DB_CONFIG)
        cursor = conn.cursor(cursor_factory=RealDictCursor)

        for table_name, primary_key in TABLES.items():
            print(f"Конвертация таблицы: {table_name}")

            if table_name == "Order":
                cursor.execute(f'SELECT * FROM
"{table_name}"')
            else:
                cursor.execute(f'SELECT * FROM
{table_name}')

            rows = cursor.fetchall()

            db = bsddb3.btopen(f"{table_name}.db", "c")
```

```

        for row in rows:
            row_dict = dict(row)

            if isinstance(primary_key, list):
                key_parts = [str(row_dict[col]) for col
in primary_key]
                key_value = "_".join(key_parts)
            else:
                key_value = str(row_dict[primary_key])

            json_value = json.dumps(row_dict,
ensure_ascii=False, default=str)
            db[key_value.encode('utf-8')] =
json_value.encode('utf-8')

            db.close()
            print(f"Таблица {table_name} конвертирована.
Записей: {len(rows)}")

        cursor.close()
        conn.close()
        print("Конвертация завершена успешно!")

    except Exception as e:
        print(f"Ошибка при конвертации: {str(e)}")

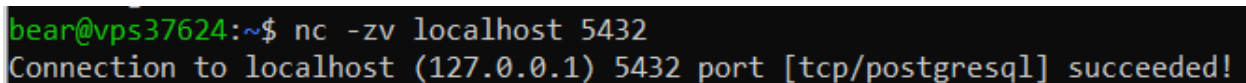
if __name__ == "__main__":
    convert_postgres_to_berkeleydb()

```

Выполнение программы производилось на выделенном сервере под управлением ОС Ubuntu 22.04. Для подключения, которого к локальной бд, был организован проброс портов:

```
ssh -R 5432:localhost:5432 vps
```

Для контроля доступности локальной БД postgresql на сервере, используется команда netstst, рисунок 2.1



```

bear@vps37624:~$ nc -zv localhost 5432
Connection to localhost (127.0.0.1) 5432 port [tcp/postgresql] succeeded!

```

Рисунок 2.1 – Проверка доступности БД

В результате работы программы были созданы файлы .bd для каждой таблицы, рисунок 2.2.

Пример данных, содержащихся в файлах приведен на рисунках 2.3, 2.4.

```
(myenv) bear@vps37624:~/db-lab/Lab3$ ll
total 232
drwxrwxr-x 3 bear bear 4096 Nov 20 20:13 ./
drwxrwxr-x 3 bear bear 4096 Nov 20 18:57 ../
-rw-rw-r-- 1 bear bear 2652 Nov 20 19:09 convert.py
-rw-rw-r-- 1 bear bear 1511 Nov 20 19:45 decode.py
-rw-rw-r-- 1 bear bear 870 Nov 20 20:01 decode_test.py
-rw-rw-r-- 1 bear bear 28672 Nov 20 19:09 dish.db
-rw-rw-r-- 1 bear bear 24576 Nov 20 19:09 dishingredient.db
-rw-rw-r-- 1 bear bear 20480 Nov 20 19:09 employee.db
-rw-rw-r-- 1 bear bear 8192 Nov 20 19:09 employeeposition.db
-rw-rw-r-- 1 bear bear 24576 Nov 20 19:09 ingredient.db
drwxrwxr-x 5 bear bear 4096 Nov 20 18:57 myenv/
-rw-rw-r-- 1 bear bear 16384 Nov 20 19:09 Order.db
-rw-rw-r-- 1 bear bear 20480 Nov 20 19:09 orderdish.db
-rw-rw-r-- 1 bear bear 16384 Nov 20 19:09 position.db
-rw-rw-r-- 1 bear bear 20480 Nov 20 19:09 supplier.db
-rw-rw-r-- 1 bear bear 30 Nov 20 20:07 test_russian.txt
-rw-rw-r-- 1 bear bear 65 Nov 20 19:25 test.txt
-rw-rw-r-- 1 bear bear 56 Nov 20 20:13 t.txt
-rw-rw-r-- 1 bear bear 20480 Nov 20 19:09 visitor.db
(myenv) bear@vps37624:~/db-lab/Lab3$
```

Рисунок 2.2 – Результат работы программы

```
(myenv) bear@vps37624:~/db-lab/Lab3$ db_dump -p orderdish.db | head -15
VERSION=3
format=print
type=btree
db_pagesize=4096
HEADER=END
10_177
{"orderid": 10, "dishid": 177, "count": 3}
10_182
{"orderid": 10, "dishid": 182, "count": 5}
10_183
{"orderid": 10, "dishid": 183, "count": 1}
10_184
{"orderid": 10, "dishid": 184, "count": 1}
10_185
{"orderid": 10, "dishid": 185, "count": 1}
```

Рисунок 2.3 – Часть данных файла orderdish.db

```
(myenv) bear@vps37624:~/db-lab/Lab3$ db_dump -p employeeposition.db | head -15
VERSION=3
format=print
type=btree
db_pagesize=4096
HEADER=END
10_8
{"employeeid": 10, "positionid": 8}
10_9
{"employeeid": 10, "positionid": 9}
11_25
{"employeeid": 11, "positionid": 25}
11_26
{"employeeid": 11, "positionid": 26}
12_11
{"employeeid": 12, "positionid": 11}
(myenv) bear@vps37624:~/db-lab/Lab3$
```

Рисунок 2.4 – Часть данных файла employeeposition.db

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы №3 были успешно реализованы все поставленные задачи, что позволило получить глубокое понимание принципов работы с NoSQL базами данных, в частности с BerkeleyDB. Реализация конвертера для передачи данных из PostgreSQL в BerkeleyDB продемонстрировала эффективность процесса транзакций и гибкость хранения информации в новом формате.

Практическая часть работы включала детальное изучение структуры хранения данных и разработку алгоритмов для их конвертации, что открыло новые горизонты в обработке секторов хранения информации. Успешное завершение проекта позволило убедиться в преимуществах NoSQL решений, таких как возможность работы с большим объемом данных и адаптивность к изменениям в структуре данных.

Результаты лабораторной работы стали важным шагом в освоении современных технологий разработки приложений и управления данными. Полученные навыки будут полезны при проектировании и оптимизации приложений, требующих работы с распределенными системами хранения данных. Заключение выполнено, все цели достигнуты, и выполненная работа подтверждает как усвоение теоретических основ, так и развитие практических навыков работы с NoSQL базами данных.