|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

Институт информационных технологий (ИИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Распределенные системы управления базами данных»

**Практическое занятие № 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО-11-22, Гришин Андрей Валерьевич* | (подпись) | |
| Преподаватель | *Красников Степан Альбертович* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

* 1. Особенности при создании таблиц Cassandra

ОТСУТСТВИЕ JOIN

В Cassandra нет опции join. Чаще всего, это ограничение обходится с помощью денормализации данных по дополнительным таблицам.

ОТСУТСТВИЕ ВНЕШНИХ КЛЮЧЕЙ

В Cassandra нет способа связывать сущности из разных таблиц между собой, как например, это сделано в реляционных базах с помощью внешних ключей. Как следствие отсутствуют такие операции как каскадное удаление. ДЕНОРМАЛИЗАЦИЯ

Сама по себе денормализация обладает одним достоинством, в отличие от обычного подход. Ее удобно использовать для хранения исторических данных, которые ни при каких обстоятельствах не должны изменяться. В качестве примера можно привести хранение покупок клиента со ссылками на товар. Т.к. цена товара меняется с течением времени, чтобы получить стоимость каждого товара на момент приобретения, нужно отдельно хранить цену товара. При использовании подхода с денормализацией, можно в таблице с покупками хранить полностью объект товара. При этом текущий товар можно изменять любым образом и даже удалять и это никак не повлияет на историю покупок клиента.

ЗАПРОС ПЕРВИЧЕН

При проектировании, нужно учитывать, что запрос должен получить за один раз все данные из одной таблицы (т.к. нет join). Поэтому сначала проектируются все возможные запросы, а затем под них создаются таблицы. Таким образом в Cassandra создание структуры БД начинается с определения запросов.

ОПТИМАЛЬНОЕ ХРАНЕНИЕ

В реляционных базах редко можно встретить рекомендации к структуре БД для оптимального хранения и чтения данных. Чаще всего никто не заботится об этом. В распределенных системах дело обстоит иначе, т.к. данные располагаются на нескольких узлах, более высокую производительно будет демонстрировать тот запрос, который отдает данные с одной ноды. Таким образом, желательно располагать данные так, чтобы данные возвращались из одной ноды.

Практическая часть:

*Листинг 1 – docker-compose.yml*

|  |
| --- |
| services:  cassandra-seed-Grishin:  image: cassandra  container\_name: cassandra-seed-Grishin  environment:  CASSANDRA\_CLUSTER\_NAME: "GrishinAV"  CASSANDRA\_ENDPOINT\_SNITCH: "GossipingPropertyFileSnitch"  CASSANDRA\_DC: "datacenter1"  AUTO\_BOOTSTRAP: "false"  volumes:  - cassandra\_data\_seed:/var/lib/cassandra  ports:  - "19040:9042"  cassandra-node-Grishin\_1:  image: cassandra  container\_name: cassandra-node-Grishin\_1  environment:  CASSANDRA\_CLUSTER\_NAME: "GrishinAV"  CASSANDRA\_ENDPOINT\_SNITCH: "GossipingPropertyFileSnitch"  CASSANDRA\_DC: "datacenter1"  CASSANDRA\_SEEDS: "cassandra-seed-Grishin"  volumes:  - cassandra\_data\_node\_1:/var/lib/cassandra  ports:  - "19041:9042"  depends\_on:  - cassandra-seed-Grishin  cassandra-node-Grishin\_2:  image: cassandra  container\_name: cassandra-node-Grishin\_2  environment:  CASSANDRA\_CLUSTER\_NAME: "GrishinAV"  CASSANDRA\_ENDPOINT\_SNITCH: "GossipingPropertyFileSnitch"  CASSANDRA\_DC: "datacenter2"  CASSANDRA\_SEEDS: "cassandra-seed-Grishin"  volumes:  - cassandra\_data\_node\_2:/var/lib/cassandra  ports:  - "19042:9042"  depends\_on:  - cassandra-seed-Grishin  volumes:  cassandra\_data\_seed:  cassandra\_data\_node\_1:  cassandra\_data\_node\_2: |

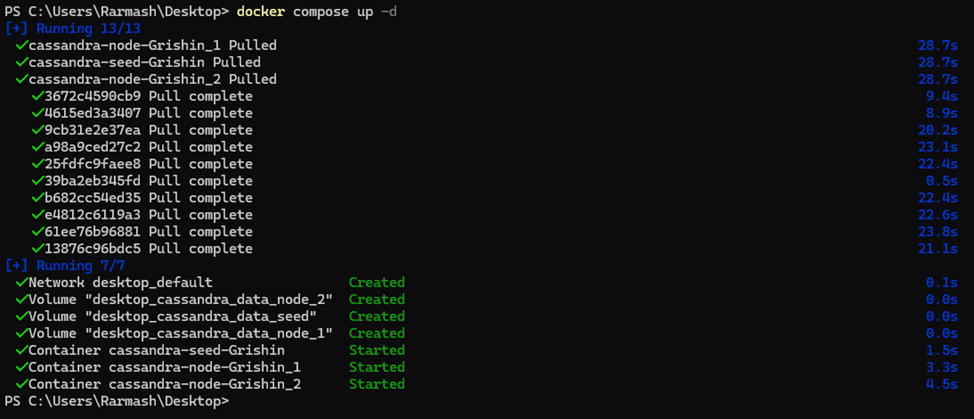


Рисунок 1 – Запуск docker compose для создания кластера

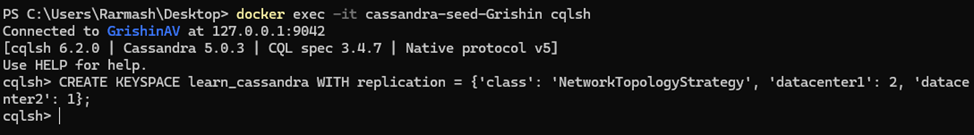


Рисунок 2 – Подключение к контейнеру и создание пространства ключей

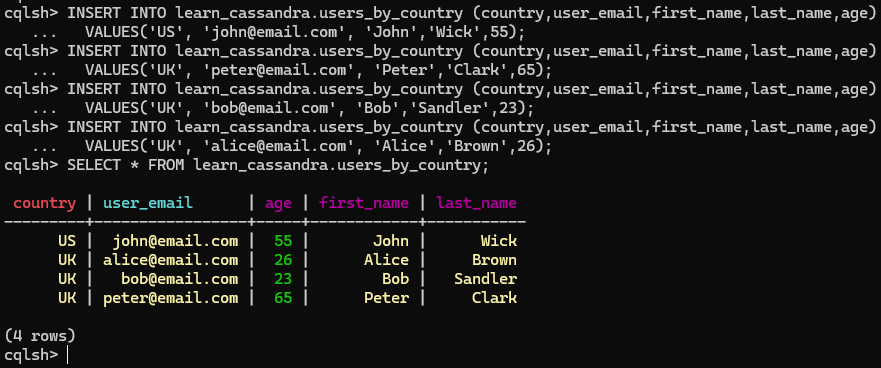
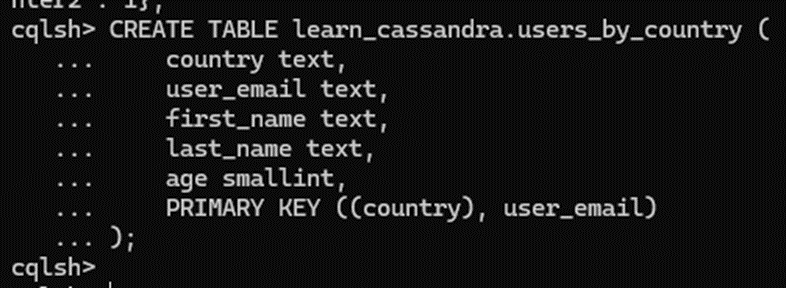


Рисунок 3 – Создание таблицы users\_by\_country с последующим её заполнением

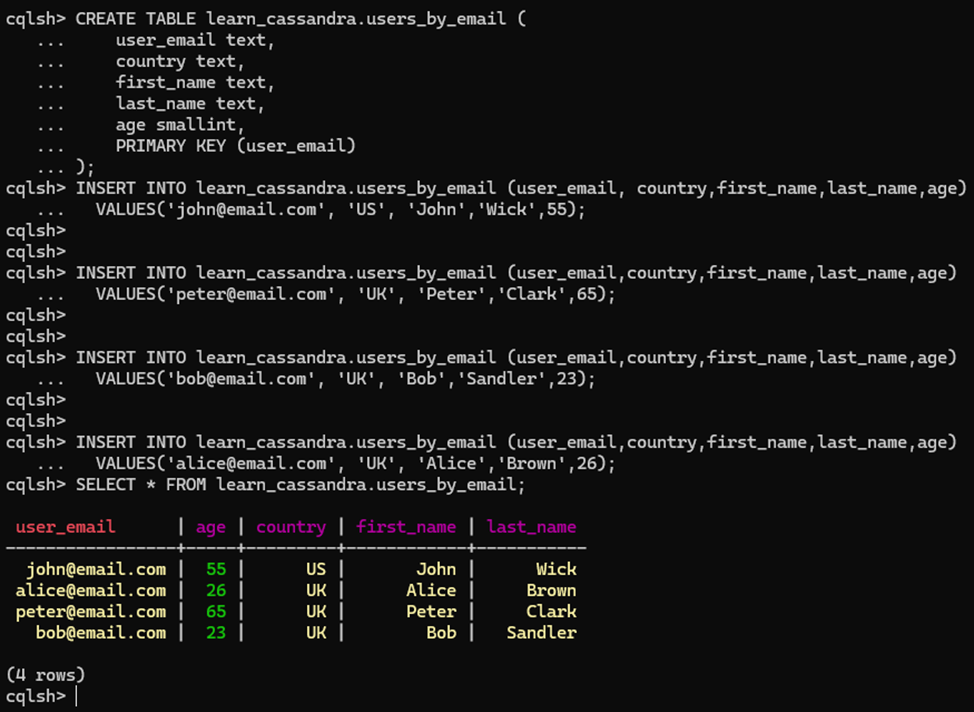
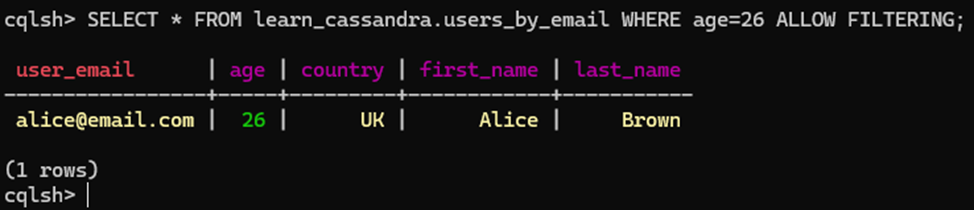


Рисунок 4 – Создание таблицы users\_by\_email с последующим её заполнением

Рисунок 5 – Фильтрация по столбцу, который не является ключом раздела

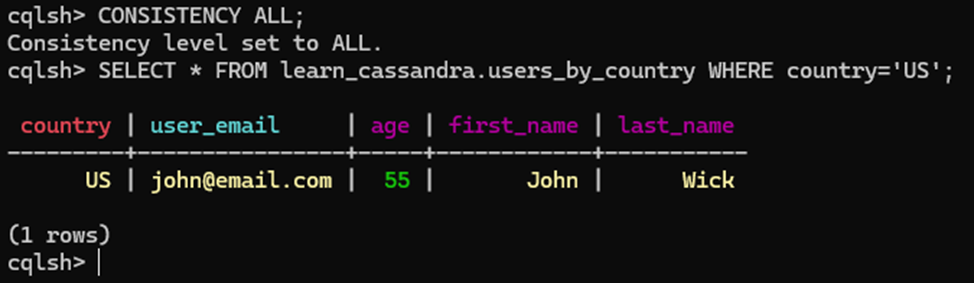


Рисунок 6 – Установление уровня согласованности «ALL» с последующим получением данных

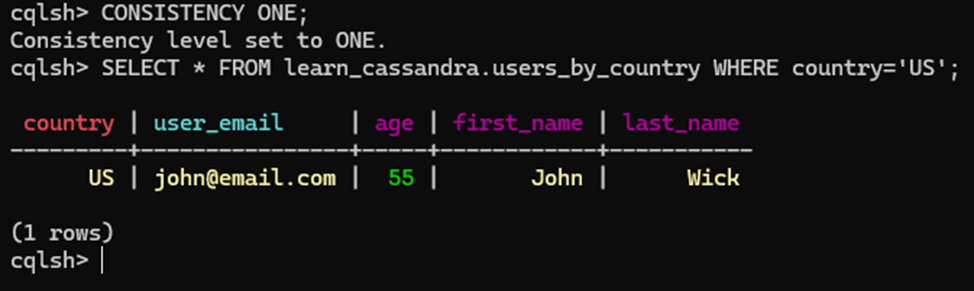


Рисунок 7 – Установление уровня согласованности «ONE» с последующим получением данных

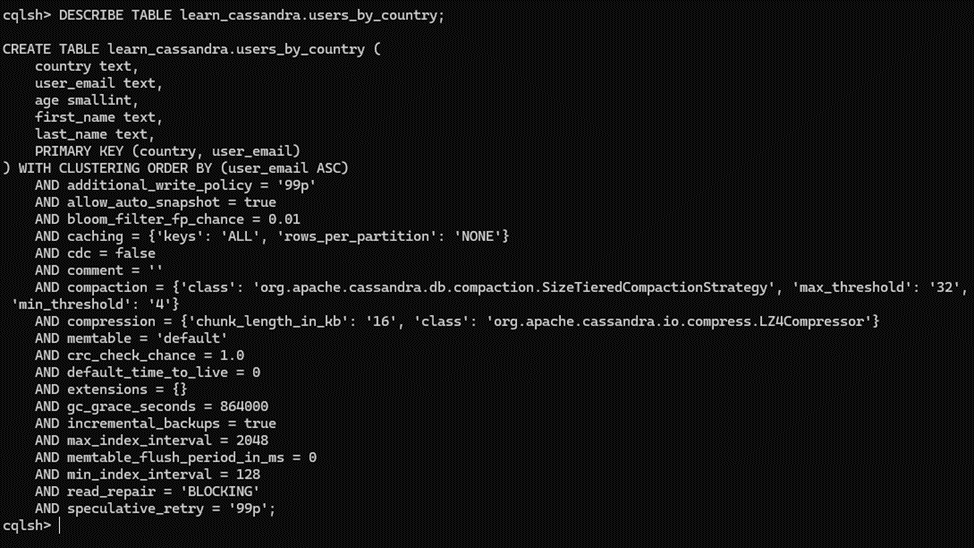


Рисунок 8 – Определение стратегий для таблицы users\_by\_country

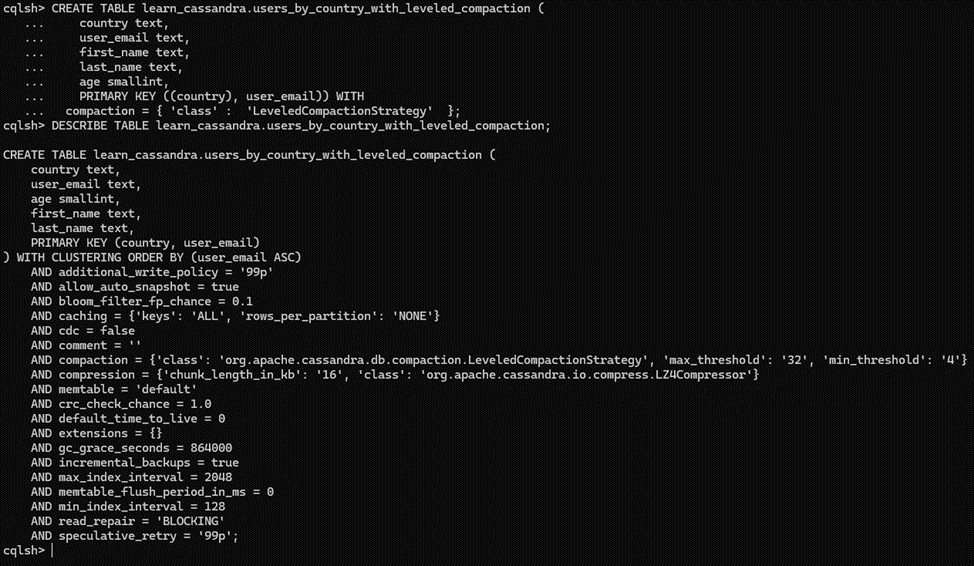
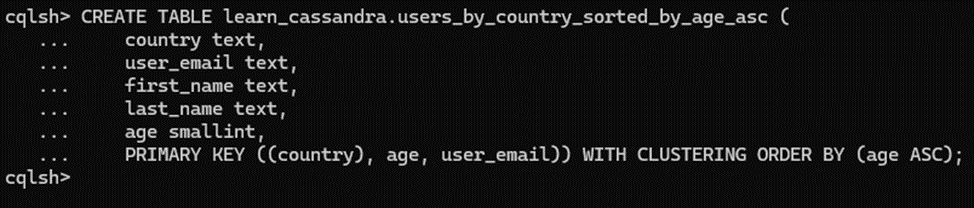


Рисунок 9 – Создание таблицы с указанием стратегии сжатия и проверка результата



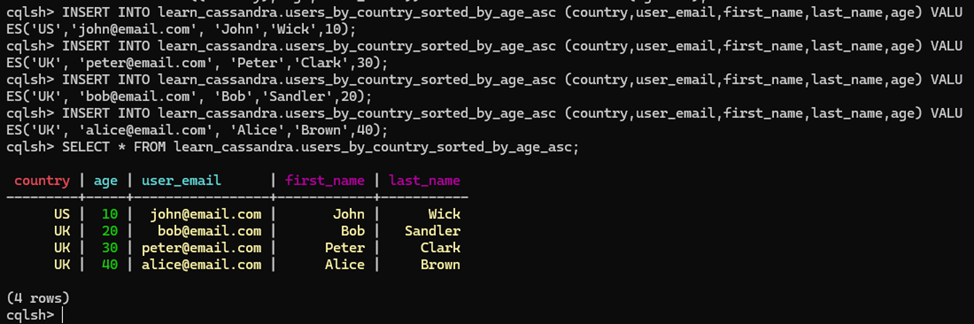


Рисунок 10 – Создание таблицы с определением столбца кластеризации для сортировки данных с последующей вставкой информации

**Заключение:**

В ходе работы были рассмотрены ключевые особенности работы с базой данных Cassandra, включая отсутствие операций JOIN и внешних ключей, необходимость денормализации данных, а также важность проектирования структуры таблиц исходя из запросов. Было отмечено, что оптимальное хранение данных в распределённой среде требует учитывать их расположение на узлах для повышения производительности.

Практическая часть включала в себя создание кластера из трёх узлов с использованием Docker Compose, изучение работы с ключами в Cassandra и анализ подходов к обеспечению согласованности данных. В результате были получены знания и навыки, необходимые для проектирования и развертывания распределённых баз данных на основе Cassandra, что позволит эффективно управлять данными в системах с высокой нагрузкой.

**Вопросы для самопроверки:**

1. Какой Snitch вы выбрали при создании docker-compose и почему? Какие варианты Snitch для Cassandra бывают?

При настройке кластера Cassandra в Docker был выбран GossipingPropertyFileSnitch, так как он обеспечивает автоматическое распространение информации о топологии через gossip-протокол, поддерживает работу с несколькими датацентрами (datacenter1 и datacenter2) и позволяет гибко настраивать DC/рэки через переменные окружения, что критично для динамических окружений с контейнерами. Этот Snitch минимизирует ручное вмешательство в конфигурацию и соответствует рекомендациям для продакшен-сред.

В Apache Cassandra доступны различные варианты Snitch, включая **SimpleSnitch**, **RackInferringSnitch**, **GossipingPropertyFileSnitch** и другие. **GossipingPropertyFileSnitch** является наиболее рекомендуемым, так как использует информацию о датацентрах и стойках из cassandra-rackdc.properties и учитывает динамические изменения кластера. **SimpleSnitch** подходит для одноцентровых развертываний, а **RackInferringSnitch** автоматически определяет стойки и датацентры по IP-адресам.

2. Укажите формулу уровней согласованности, которая обеспечивает высокую согласованность данных:

Формула уровней согласованности в Apache Cassandra определяется следующим образом: **R+W>N**

где:

* R — уровень согласованности для операций чтения (количество узлов, с которых должно быть получено подтверждение чтения).
* W — уровень согласованности для операций записи (количество узлов, которые должны подтвердить запись).
* N — фактор репликации (количество узлов, хранящих копию данных).

Для обеспечения высокой согласованности данных обычно выбирают кворумную стратегию: **R=W=N/2+1** что гарантирует пересечение множеств узлов при операциях чтения и записи, обеспечивая актуальность данных.