|  |
| --- |
| 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 № 1**

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

Москва 2025 г.

**Теоретическое введение:**

Apache Cassandra – открытая, распределенная, децентрализованная, эластично масштабируемая, высокодоступная, отказоустойчивая, строковая, допускающая настройку согласованности база данных, дизайн распределенности которой основан на Amazon Dynamo, а модель данных - на Google Bigtable. Разработана для социальной сети, в настоящее время используется на некоторых наиболее популярных веб-сайтах.

**Модель данных**

В терминологии кассандры приложение работает с пространством ключей (keyspace), что соответствует понятию схемы базы данных (database schema) в реляционной модели. В этом пространстве ключей могут находиться несколько колоночных семейств (column family), что соответствует понятию реляционной таблицы. В свою очередь, колоночные семейства содержат колонки (column), которые объединяются при помощи ключа (row key) в записи (row). Колонка состоит из трех частей: имени (column name), метки времени (timestamp) и значения (value). Колонки в пределах записи упорядочены. В отличие от реляционной БД, никаких ограничений на то, чтобы записи (а в терминах БД это строки) содержали колонки с такими же именами, как и в других записях — нет. Также в последних версиях кассандры появилась возможность выполнять запросы определения и изменения данных (DDL, DML) при помощи языка CQL.

**Особенности**

Запись в кассандру работает с большей скоростью, чем чтение. Это меняет подход, который применяется при проектировании. Если рассматривать кассандру с точки зрения проектирования модели данных, то проще представить колоночное семейство не как таблицу, а как материализованное представление (materialized view) — структуру, которая представляет данные некоторого сложного запроса, но хранит их на диске. Вместо того, чтобы пытаться как-либо скомпоновать данные при помощи запросов, лучше постараться сохранить в коночное семейство все, что может понадобиться для этого запроса. То есть, подходить необходимо не со стороны отношений между сущностями или связями между объектами, а со стороны запросов: какие поля требуются выбрать; в каком порядке должны идти записи; какие данные, связанные с основными, должны запрашиваться совместно — всё это должно уже быть сохранено в колоночное семейство. Количество колонок в записи ограничено теоретически 2 миллиардами.

**Практическая часть:**

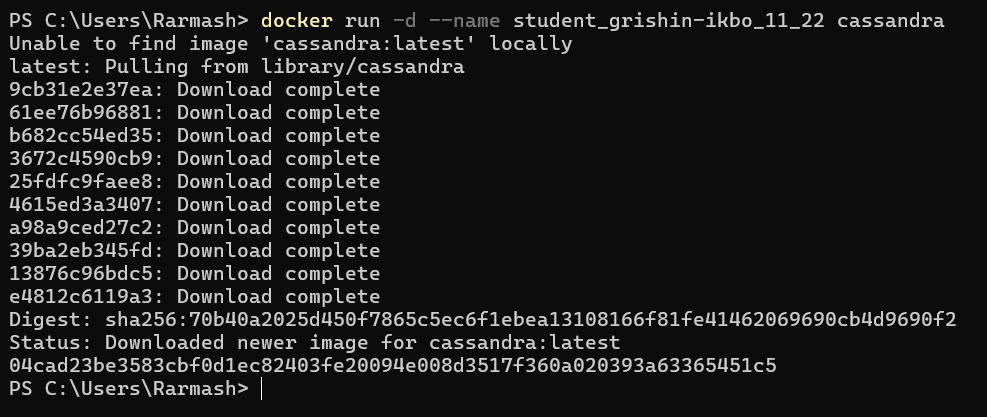


Рисунок 1 - Запуск контейнера с Cassandra

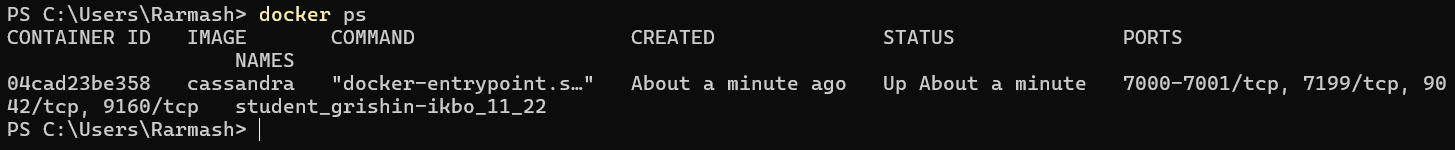


Рисунок 2 - Проверьте работоспособность контейнера

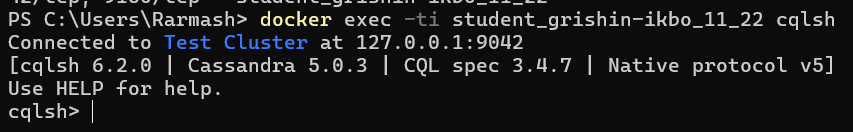


Рисунок 3 - Получение доступа к оболочке

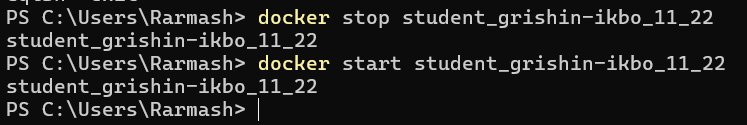


Рисунок 4 - Остановка и перезапуск контейнера

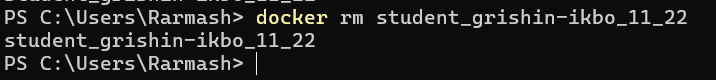


Рисунок 5 - Удаление контейнера

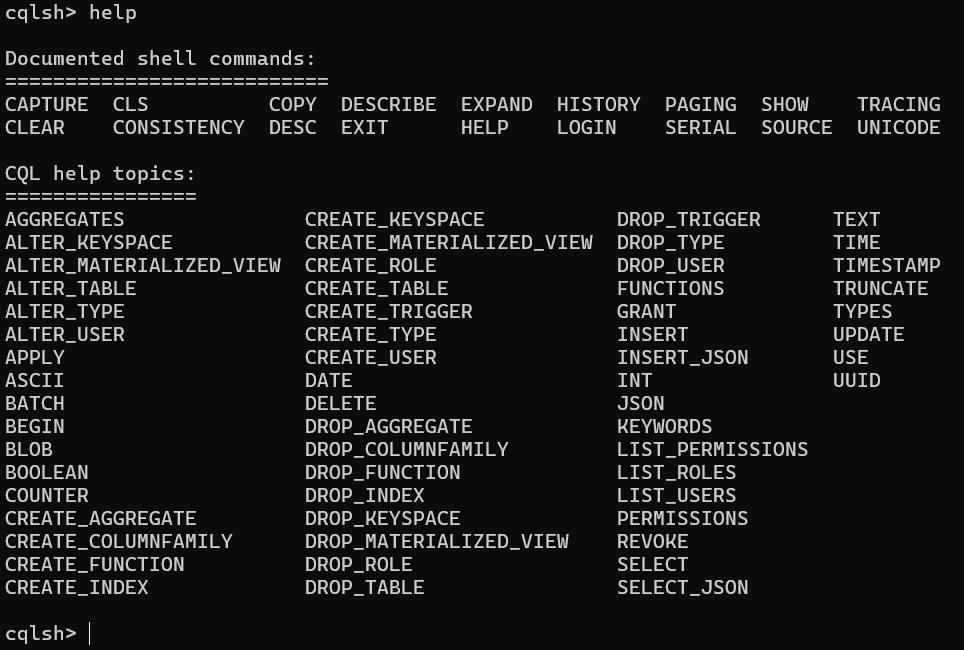


Рисунок 6 - Получения справки по cqlsh

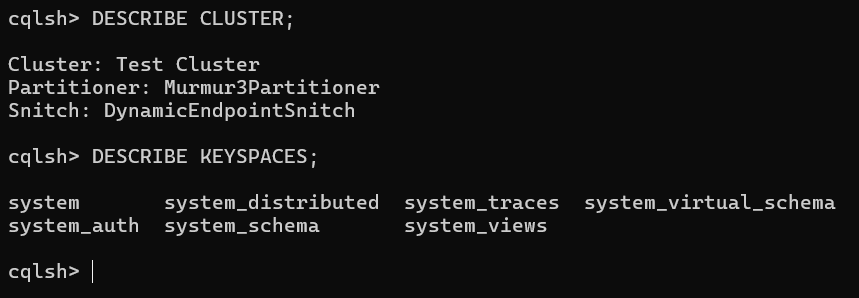


Рисунок 7 - Описание окружения в cqlsh

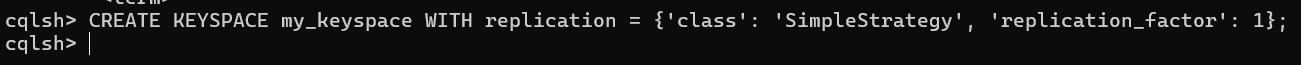


Рисунок 8 - Создание пространства ключей

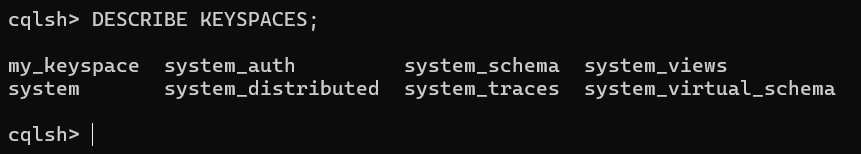


Рисунок 9 – Описание пространств ключей



Рисунок 10 - Переключение на созданное пространство ключей



Рисунок 11 – Создание таблицы

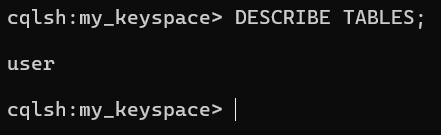


Рисунок 12 – Описание созданной таблицы

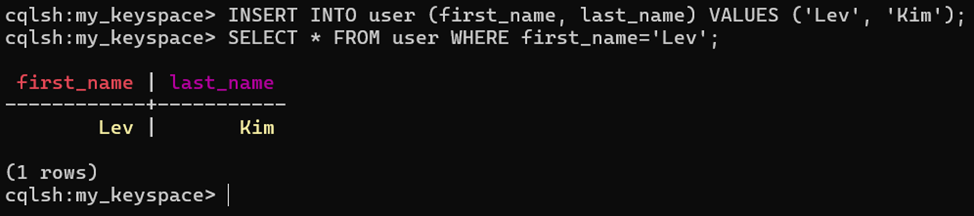


Рисунок 13 – Запись и чтение данных таблицы

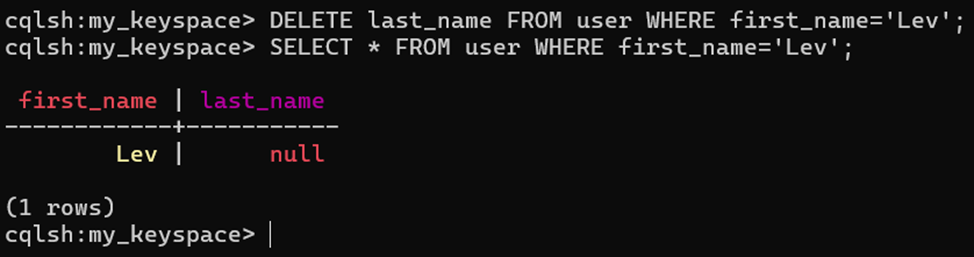


Рисунок 14 – Удаление столбца и проверка данных таблицы

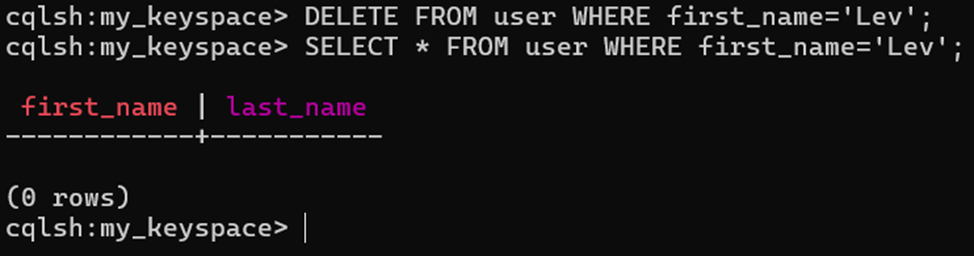


Рисунок 15 – Удаление строки и проверка данных таблицы

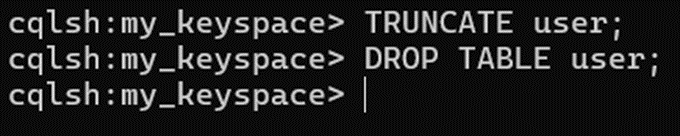


Рисунок 16 – Удаление данных и схемы таблицы

**1. Сходства и различия моделей данных в реляционных СУБД и Cassandra**

**Сходства:**

* **Табличное представление данных:** В обеих системах данные хранятся в таблицах с колонками и строками.
* **Именованные колонки:** Оба типа СУБД оперируют колонками, которые имеют определённые имена и типы данных.
* **Индексы:** Обе системы поддерживают индексацию для ускорения поиска данных.

**Различия:**

* **Структура данных:**
  + В **реляционных СУБД** (например, PostgreSQL, MySQL) модель строится на основе **строго нормализованных таблиц** с отношениями (Foreign Keys).
  + В **Cassandra** модель **денормализована** — данные часто дублируются, чтобы оптимизировать запросы.
* **Первичный ключ:**
  + В реляционных базах он состоит из одного или нескольких атрибутов и часто связан с автоинкрементом.
  + В Cassandra первичный ключ всегда включает **партиционный ключ** (Partition Key), который определяет, на каком узле хранится строка.
* **Запросы и оптимизация:**
  + Реляционные СУБД используют **SQL** с мощными возможностями JOIN'ов и сложных агрегаций.
  + В Cassandra используется **CQL (Cassandra Query Language)**, который похож на SQL, но не поддерживает JOIN и сложные вложенные запросы.
* **Масштабируемость:**
  + Реляционные базы **масштабируются вертикально** (добавлением ресурсов на сервер).
  + Cassandra **масштабируется горизонтально**, автоматически распределяя данные между узлами.
* **Последовательность транзакций:**
  + Реляционные базы обеспечивают **ACID**-свойства (атомарность, согласованность, изолированность, надёжность).
  + Cassandra следует модели **BASE** (Basic Availability, Soft state, Eventual consistency), где согласованность может быть слабее в пользу доступности и масштабируемости.

**2. Для каких задач может использоваться Cassandra?**

Cassandra подходит для высоконагруженных распределённых систем, требующих высокой доступности и горизонтального масштабирования. Примеры задач:

* **Системы хранения логов и событий** (например, мониторинг серверов, аналитика трафика).
* **Системы рекомендаций** (например, персонализация контента, рекомендации товаров).
* **IoT (Интернет вещей)** — обработка огромного потока данных с датчиков.
* **Социальные сети** — хранение лайков, комментариев, подписок (например, Twitter/X, Instagram).
* **Мессенджеры и чаты** — быстрый доступ к истории сообщений.
* **Финансовые системы** — хранение транзакций с высокой скоростью записи (но не критичных к строгой согласованности).

Cassandra отлично справляется с задачами, где важны скорость, отказоустойчивость и масштабируемость, но плохо подходит для сложных реляционных запросов с JOIN'ами.

**Список использованных источников и литературы:**

1. Стин ван М., Таненбаум Э. С. Распределенные системы / пер. с англ. В. А. Яроцкого. – М.: ДМК Пресс, 2021. – 584 с.: ил.
2. Ёcy М. Т., Вальдуриес П. Принципы организации распределенных баз данных / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2021. – 672 с.: ил.
3. Петров Алекс. Распределенные данные. Алгоритмы работы современных систем хранения информации. — СПб.: Питер, 2021. — 336 с.: ил. — (Серия «Бестселлеры O’Reilly»).
4. Соколинский Л. Б. Параллельные системы баз данных: Учебное пособие / Предисл.: В. А. Садовничий. – М.: Издательство Московского университета, 2013. – 184 с., илл. – (Серия «Суперкомпьютерное образование»).
5. Григорьев Ю. А., Плутенко А. Д., Плужников В. Л., Ермаков Е. Ю., Цвященко Е. В., Пролетарская В. А. Теория и практика анализа параллельных систем баз данных. - Владивосток: Дальнаука; 2015. - 336 с.