Гусев Виталий Евгеньевич

09-335

Отчёт по курсу «Нереляционные БД»

Оглавление

[1. Описание реализации базы данных проекта 3](#_Toc185699763)

[1.1. Реализация реляционной модели на PostgreSQL 4](#_Toc185699764)

[1.2. Реализация объектно-реляционной модели на PostgreSQL 5](#_Toc185699765)

[1.3. Реализация нереляционной модели на Neo4j 6](#_Toc185699766)

[1.4. Реализация нереляционной модели с помощью Redis 7](#_Toc185699767)

[1.5. Реализация нереляционной модели с помощью MongoDB. 8](#_Toc185699768)

[1.6. Реализация нереляционной модели с помощью Cassandra 10](#_Toc185699769)

[2. Сравнение реляционного и нереляционных вариантов проекта 13](#_Toc185699770)

[2.1. Удобство установки 13](#_Toc185699771)

[2.2. Удобство интерфейса 13](#_Toc185699772)

[2.3. Сложность получения данных 14](#_Toc185699773)

[2.4. Сложность добавления данных 15](#_Toc185699774)

[2.5. Сложность удаления данных 16](#_Toc185699775)

[2.6. Сложность обновления данных 16](#_Toc185699776)

# 1. Описание реализации базы данных проекта

Для реализации проекта на различных СУБД была выбрана тема «Мессенджер». Данный проект состоит из таблиц:

1. Тип канала (ChannelType):

- Название,

2. Канал (Channel):

- Название,

- «Типа канала»,

3. Чат (Chat):

- Название,

- Дата создания,

- «Канал»,

4. Sender (Отправитель):

- Имя,

- Дата создания,

5. Сообщение чата (ChatMessage):

- Текст сообщения,

- Дата создания,

- «Чат»,

- «Отправитель».

## 1.1. Реализация реляционной модели на PostgreSQL

Была реализована реляционная модель на PostgreSQL, состоящая из таблиц, связанных по ключу. Все таблицы имеют внутренний ключ по полю id.

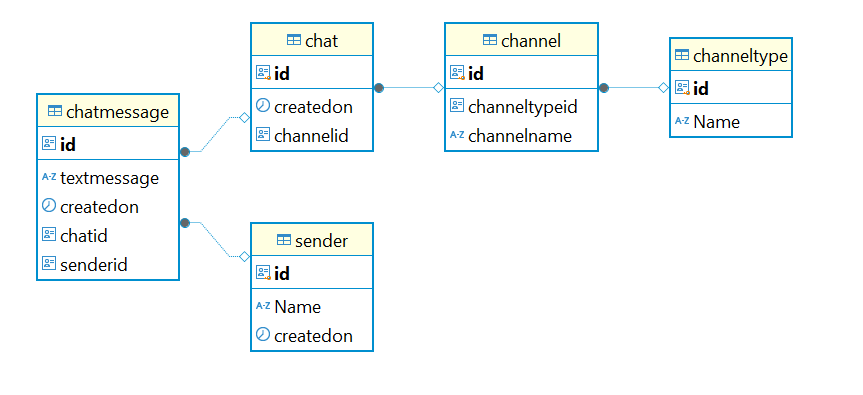


Таблица «Сообщение чата» (ChatMessage) имеет внешние ключи, связанные с таблицами «Отправитель» (Sender) через поле senderid и «Чат» (Chat) через chatid.

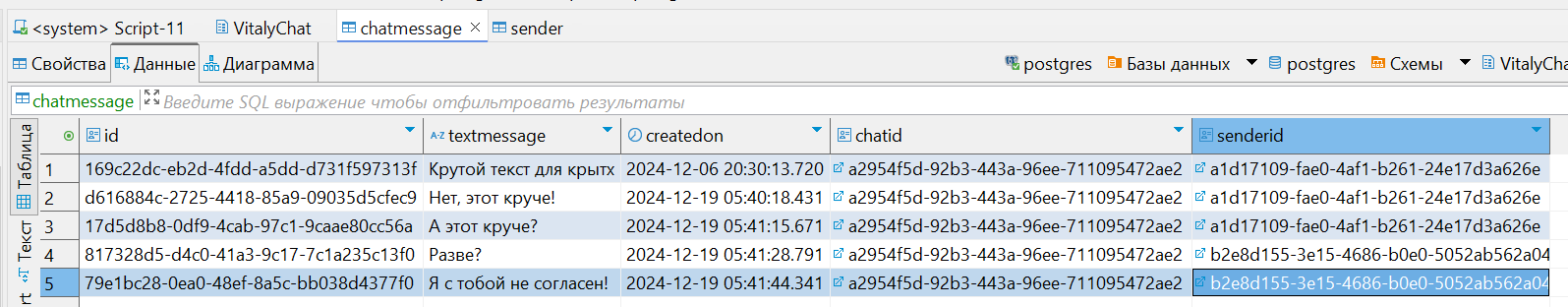


Таблица «Отправитель» (Sender).

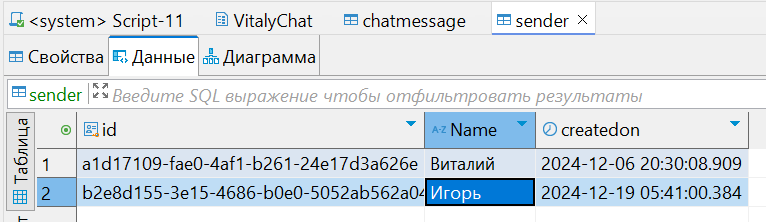


Таблица «Чат» (Chat) имеет внешний ключ на таблицу «Канал» (Channel) через поле channelid.

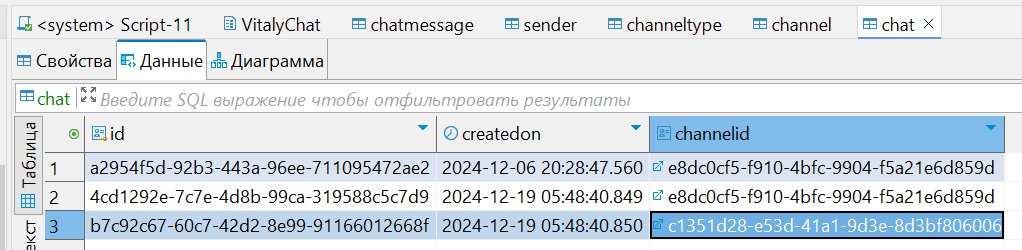
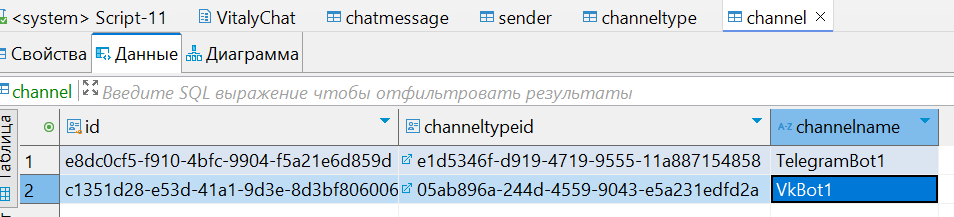
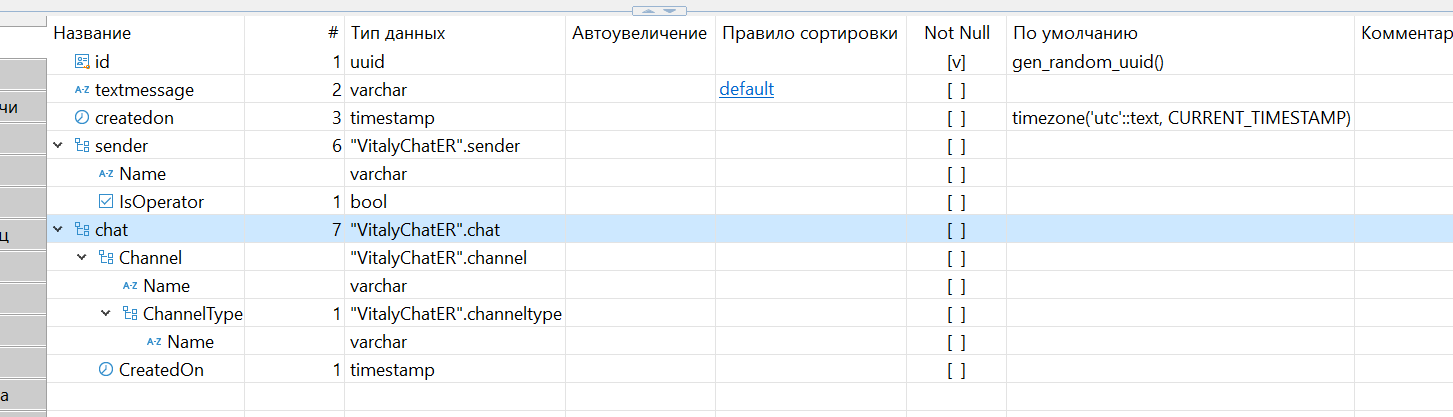


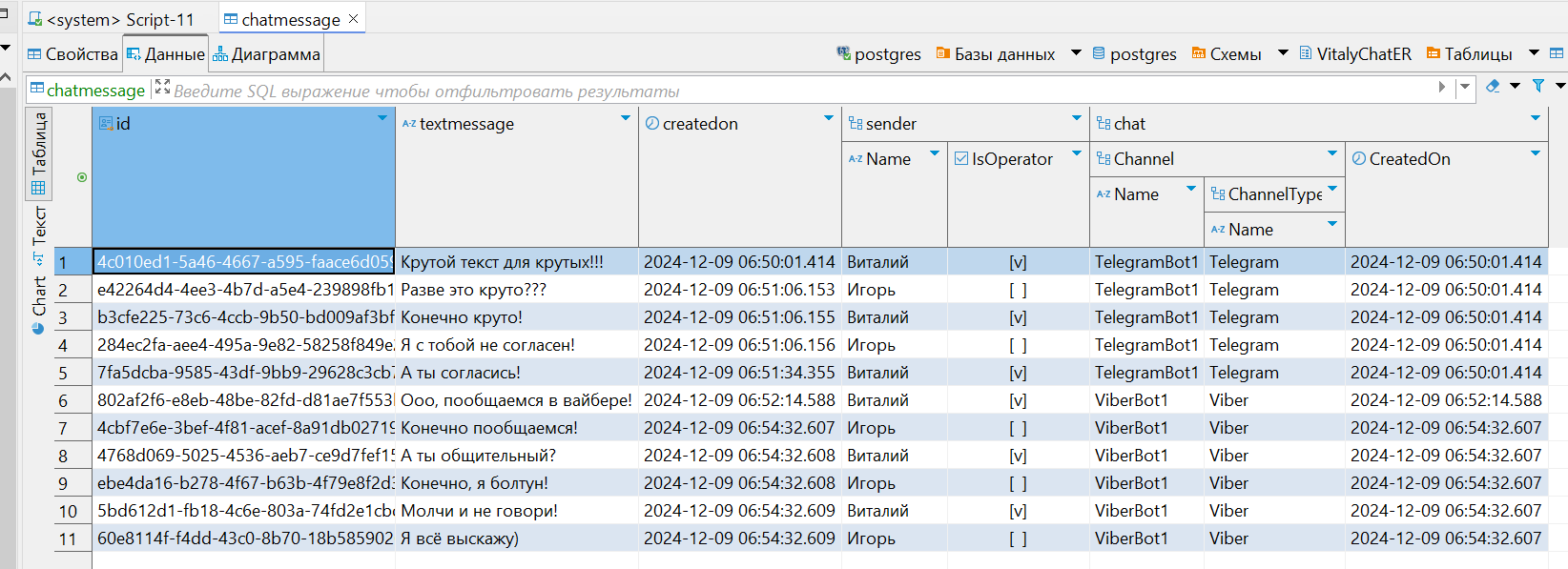
Таблица «Канал» (Channel) имеет внешний ключ на таблицу «Тип канала» (ChannelType) по полю channeltypeid.



## 1.2. Реализация объектно-реляционной модели на PostgreSQL

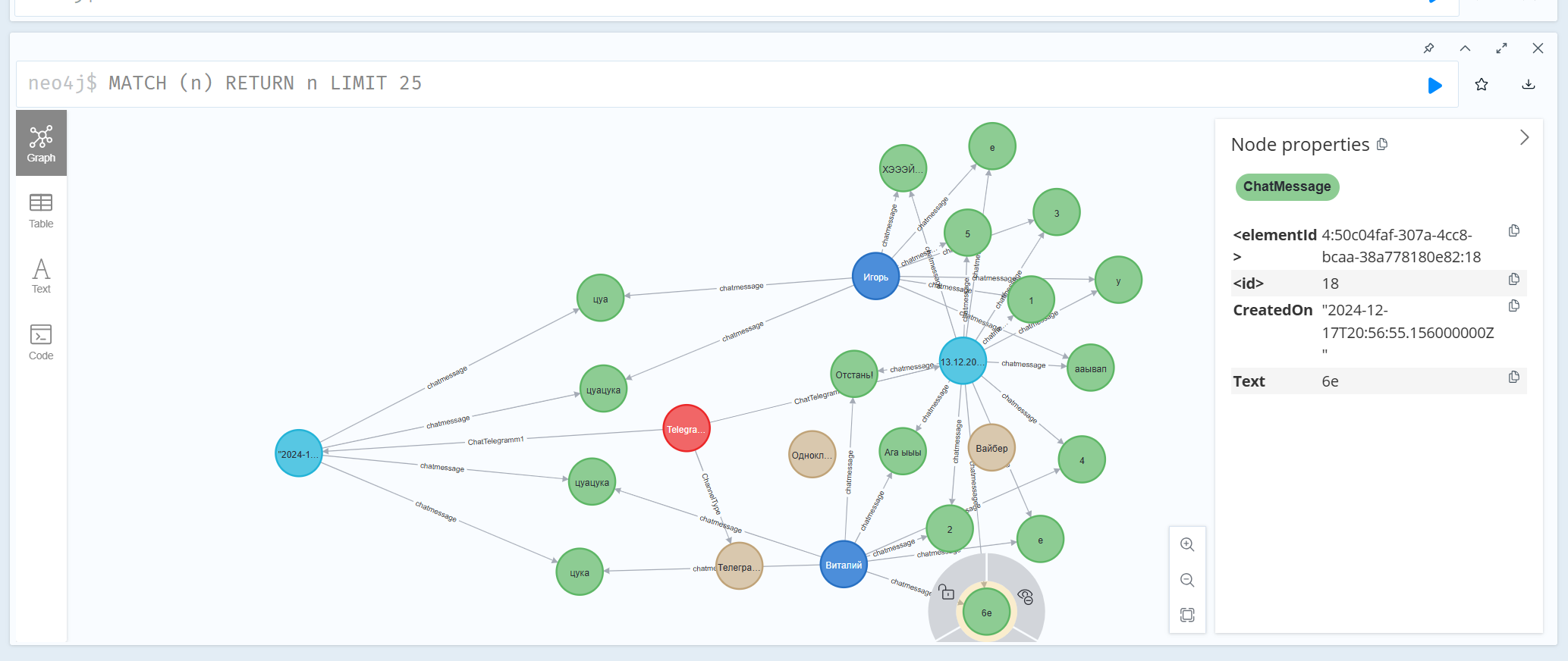
Была реализована объектно-реляционная модель на PostgreSQL, состоящая из одной таблицы «Сообщение чата» (ChatMessage) и типов данных, которые в ней содержатся: «Отправитель» (Sender) и «Чат» (Chat). В типе данных «Чат» (Chat) содержится тип данных «Канал» (Channel), а в типе данных «Канал» (Channel) содержится тип данных «Тип канала» (ChannelType).



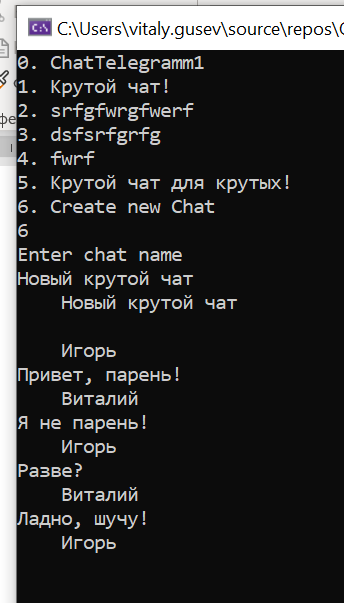


## 1.3. Реализация нереляционной модели на Neo4j

Была реализована нереляционная графовая модель на Neo4j. Проект состоит из узлов «Канал» (Channel) имеющие рёбра в узлы «Тип канала» (ChannelType) и в узлы «Чат» (Chat). Узлы «Чат» (Chat) и «Отправитель» (Sender) имеют рёбра в узлы «Сообщение чата» (ChatMessage).



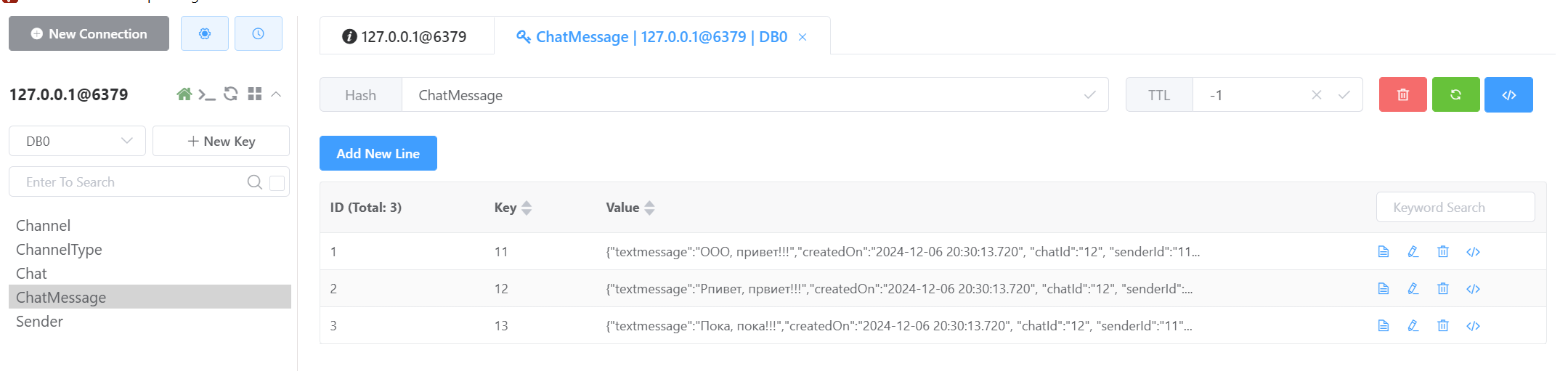
Для данного проекта было реализовано консольное приложение на языке программирования C# на .NET8, которое позволяет в списке чатов выбрать чат, либо создать новый. Если создать новый чат, то создаётся новый узел чата, к которому создаются новые узлы с сообщениями при отправке. Если выбрать чат из созданных, то отобразится история чата и можно в нём продолжить диалог.



## 1.4. Реализация нереляционной модели с помощью Redis

Была реализована нереляционная модель «ключ-значение» с помощью Redis. Были реализованы ключи через хэши. Были реализованы ключи: «Канал» (Channel), «Тип канала» (ChannelType), «Чат» (Chat), «Сообщение чата» (ChatMessage), «Отправитель» (Sender).

Все созданные ключи имеют идентификаторы для связи с другими ключами, содержащиеся в ключах (столбец key) записей. В значениях (столбец value) записей содержатся информация об объекте в формате json.



## 1.5. Реализация нереляционной модели с помощью MongoDB.

Была реализована нереляционная документная модель с помощью MongoDB. Были созданы коллекции: «Чат» (Chat), «Сообщение чата» (ChatMessage) и «Отправитель» (Sender).

Записи в коллекции «Сообщение чата» (ChatMessage) имеют структуру из текста сообщения, даты создания, вложенного объекта Отправитель с полями имя и признак, что он оператор. Также внутри есть объект Чат, который имеет объект Канал с именем канала, внутри канала есть тип канала. Пример в формате json:

{

"\_id": {

"$oid": "675695fadb5eae3ce45aa96f"

},

"textMessage": "Крутой текст для крутых!!!",

"CreatedOn": "2024-12-09 06:50:01.414",

"Sender": {

"Name": "Виталий",

"IsOperator": true

},

"Chat": {

"Channel": {

"Name": "TelegramBot1",

"ChannelType": {

"Name": "Telegram"

}

},

"CreatedOn": "2024-12-09 06:50:01.414"

}

}

Записи в коллекции «Чат» (Chat) имеет объект Канал, который имеет название и тип канала. Пример в формате json:

{

"\_id": {

"$oid": "6757c614b88c895cf55b1159"

},

"Channel": {

"Name": "TelegramBot1",

"ChannelType": {

"Name": "Telegram"

}

},

"CreatedOn": "2024-12-09 06:50:01.414"

}

Записи в коллекции «Отправитель» (Sender) имеют поле название и признак, что он оператор. Пример в формате json:

{

"\_id": {

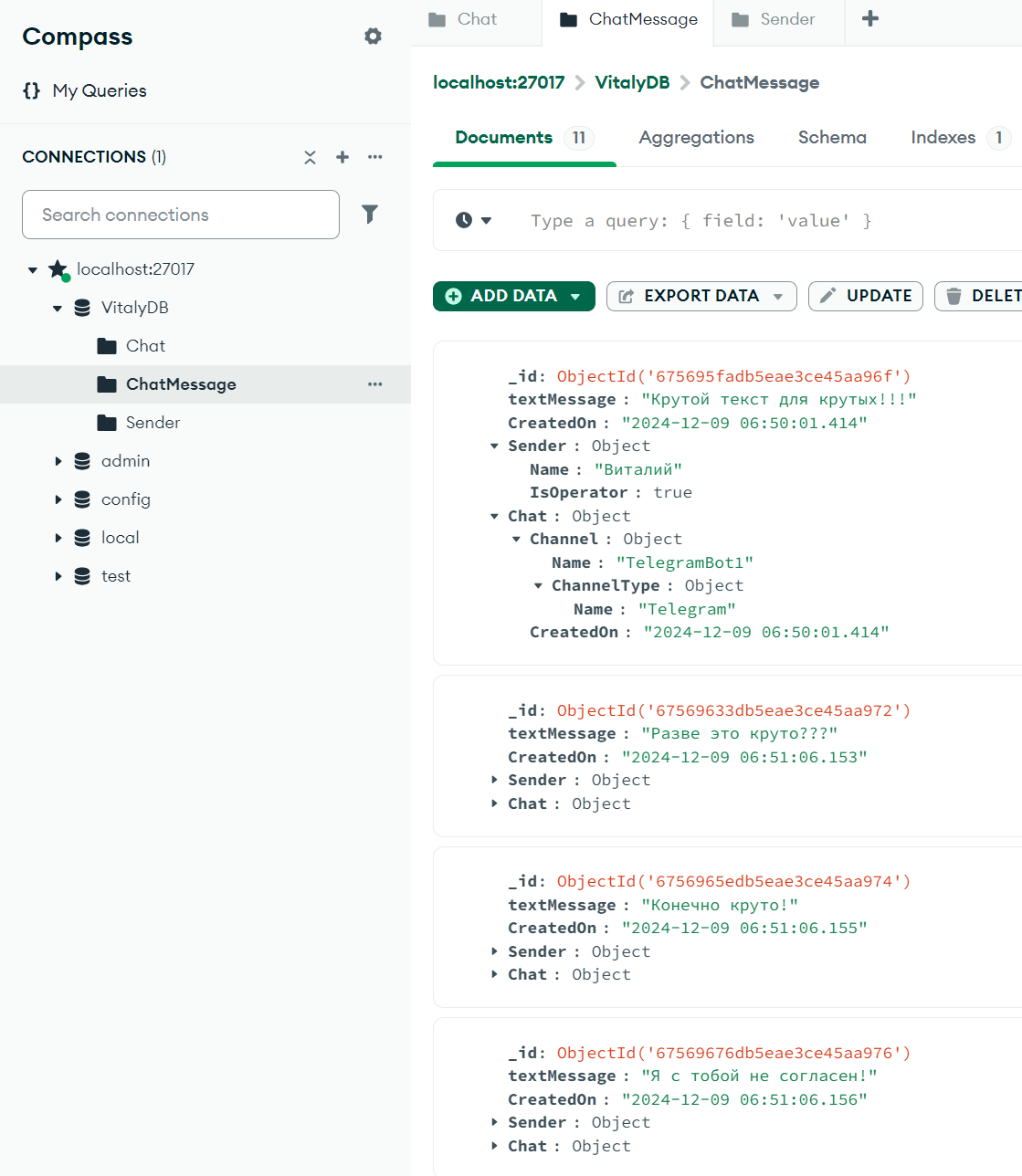
"$oid": "6757c5c3b88c895cf55b1151"

},

"Name": "Виталий",

"IsOperator": true

}



## 1.6. Реализация нереляционной модели с помощью Cassandra

Была реализована нереляционная модель данных «семейство столбцов» с помощью Cassandra.

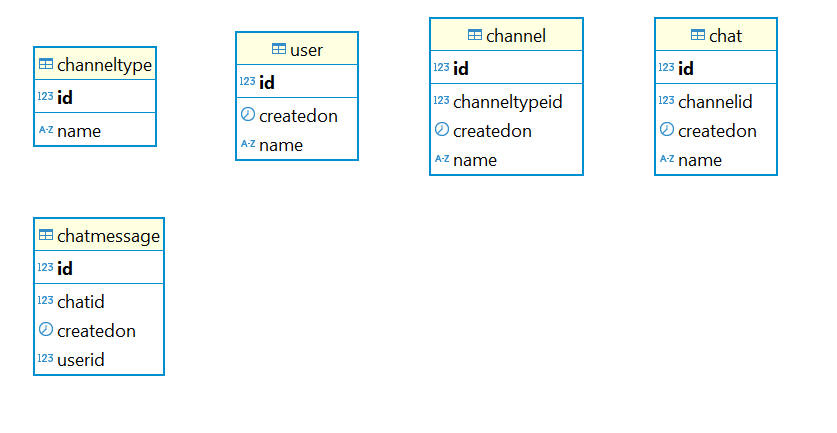


Таблица «Сообщение чата» (ChatMessage) имеет идентификаторы, связанные с таблицами «Отправитель» (Sender) через поле senderid и «Чат» (Chat) через chatid.

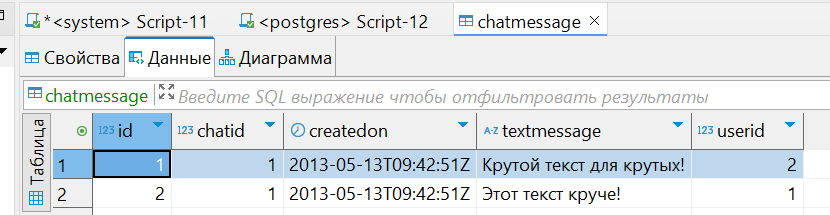


Таблица «Отправитель» (Sender).

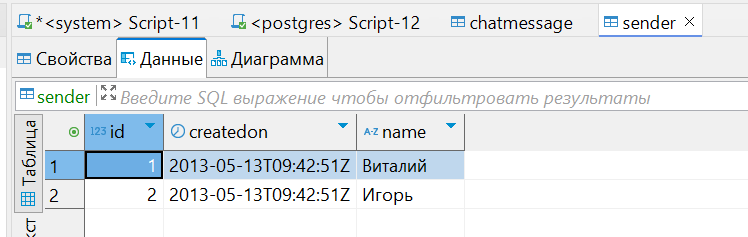


Таблица «Чат» (Chat) имеет идентификатор на таблицу «Канал» (Channel) через поле channelid.

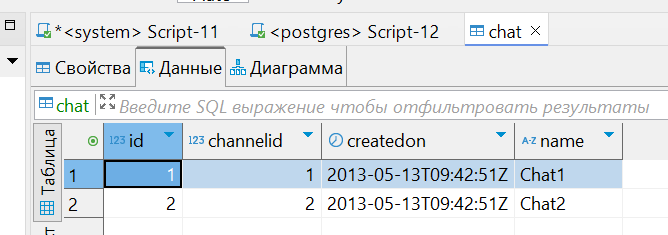


Таблица «Канал» (Channel) имеет идентификатор на таблицу «Тип канала» (ChannelType) по полю channeltypeid.

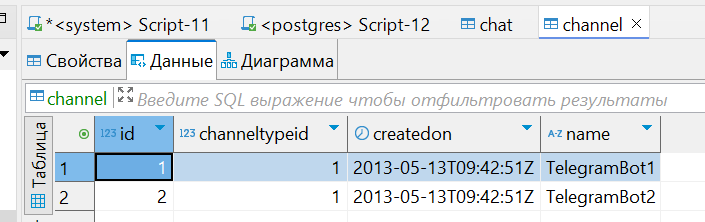
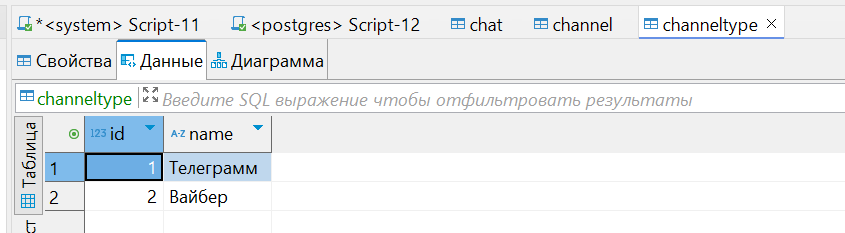


Таблица «Тип канала» (ChannelType).



# 2. Сравнение реляционного и нереляционных вариантов проекта

В результате выполненного проекта на различных СУБД были сделаны выводы, чтобы сравнить их различные критерии.

## 2.1. Удобство установки

PostgreSQL был прост в скачивании и установки.

Neo4j на сегодняшний день имеет трудности в скачивании на территории Российской Федерации и после установки через ВПН имеет ограниченное время работы на ПК, далее после переустановки прекращает работать. Для установки данного СУБД был скачан образ и создан контейнер через Docker, где запускается без проблем и трудностей.

Redis на сегодняшний день имеет трудности в скачивании на территории Российской Федерации. Чтобы скачать и работать с Redis, была скачена установка с GitHub. Установка была без проблем.

MongoDB на сегодняшний день имеет трудности в скачивании на территории Российской Федерации. Если скачать установщик через ВПН, то установка проходит без трудностей.

Cassandra на сегодняшний день имеет очень сложную и непонятную установку, поэтому была использована старая версия 3.0.4, которая устанавливается без проблем.

В результате сравнения PostgreSQL оказался самым простым в скачивании и установке.

## 2.2. Удобство интерфейса

Для работы с PostgreSQL была использована бесплатная версия DBeaver. Данное приложение позволяет легко подключаться к БД PostgreSQL. Имеются встроенные возможности создания таблиц, колонок и записей, просмотр диаграмм, написание запросов и т.д.

Для работы с Neo4j было использовано встроенное браузерное приложение, через которое можно легко подключиться, просматривать графы и внутренности узлов, писать запросы.

Для работы с Redis было использовано приложение «Another Redis Desktop Manager», в котором можно подключиться к БД, создавать ключи, писать запросы и смотреть подробную информацию по ключам.

Для работы с MongoDB было использовано приложение «MongoDB Compass», которое позволяет легко подключиться к БД, писать запросы, создавать коллекции, добавлять записи через интерфейс и просматривать данные.

Для работы с Cassandra была использована платная версия DBeaver, так как удобное бесплатное приложение для работы с данной СУБД найдено не было. Были проблемы с написанием запросов в СУБД и не хватает встроенных возможностей для работы с СУБД.

В результате сравнения почти все приложения для работы с СУБД были удобными, кроме платного DBeaver для работы с Cassandra.

## 2.3. Сложность получения данных

Для получения данных в реляционном PostgreSQL использовались запросы, где можно легко по ключу связывать таблицы, использовать представления и функции.

Для получения данных в объектно-реляционном PostgreSQL можно легко вытащить данные, из одной таблицы, которые будут содержать информацию о других объектах, если они в ней созданы. Также можно получать информацию, связывая таблицы по ключам, использовать представления и функции.

Для получения данных в Neo4j были трудности, что имеется множество узлов и их можно связать только если есть прямая связь. Если рёбра отсутствует, то тяжело связать данные для вывода.

Для получения данных в Redis можно обращаться к ключам, но нельзя эти ключи через запрос связать. Можно связать только если работать с данными через стороннее приложение.

Для получения данных через MongoDB можно обращаться к объектам и связывать данные через агрегированные функции, либо через стороннее приложение.

Получение данных через Cassandra можно делать через похожий на обычный select-запрос, но отсутствует возможность связывать данные через join. Поэтому записи можно связывать через стороннее приложение.

В результате сравнения наибольшую удобность получения данных имеет объектно-реляционный PostgreSQL, так как имеет классический select-запрос с возможностью просмотра вложенных объектов.

## 2.4. Сложность добавления данных

Чтобы добавить связанные данные в реляционном PostgreSQL, нужно добавлять данные, имеющие реально существующую запись, иначе будет ошибка добавления записи. При этом нужно сначала создать запись, на которую будет ссылаться новая запись. Данный момент создаёт проблемы добавления новых записей.

Чтобы добавить связанные данные в объектно-реляционном PostgreSQL, можно создать запись, дополняя информацию о вложенном объекте, не вызывая ошибок добавления.

Чтобы добавить связанные данные в Neo4j, можно создавать узлы, указывая необходимые рёбра между узлами сразу, что не создаёт проблем создания новых записей.

Чтобы добавить связанные данные в Redis и MongoDB, можно добавлять записи в формате json, не переживая об ошибках связности. Но может возникнуть ошибка при поиске данных, если в итоге идентификатор записи, на которую ссылается запись, не существует.

Чтобы добавить необходимые записи в Cassandra, можно указывать идентификаторы записей, на которые ссылается запись, не получая явную ошибку добавления. Но может возникнуть ошибка при поиске данных, если в итоге идентификатор записи, на которую ссылается запись, не существует.

В результате сравнения наибольшую удобность добавления записей имеет neo4j, так как можно указывать как идут записи. Также объектно-реляционный PostgreSQL даёт удобство добавления связанных данных, так как в каждой записи могут быть вложенные объекты, которые можно заполнять.

## 2.5. Сложность удаления данных

При удалении записи в реляционном PostgreSQL может возникнуть ошибка удаления, если удалить запись, на которую ссылается какая-нибудь запись. Это вызывает проблемы удаления, так как связи могут быть тяжёлыми.

В объектно-реляционном PostgreSQL, Neo4j, Redis, MongoDB, Cassandra идёт прямое удаление записей без проблем, но все эти БД могут потерять связи со связанными объектами по идентификатору, либо рёбрами в Neo4j.

В результате сравнения нереляционные БД дают наибольшее удобство удаления записей, так как реляционные БД дают проблемы из-за наличий связей.

## 2.6. Сложность обновления данных

При обновлении записи в реляционном PostgreSQL может возникнуть ошибка обновления поля, который является внешним ключом на другую таблицу, либо является внутренним ключом, на которую ссылается другая таблица. Это создаёт безопасность обновления данных, но трудность их обновления. В условиях связности легко обновлять записи по условиям между таблицами.

В нереляционных БД можно легко обновлять идентификаторы, но есть угроза, что связность может быть нарушена. В PostgreSQL легко обновлять вложенные объекты. В Neo4j легко обновить связанные данные, если присутствуют рёбра. В MongoDB, Redis, Cassandra есть трудности обновления связанных данных, нужно использовать стороннее приложение, которое работает с данными БД.