 **НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ЯДЕРНЫЙ УНИВЕРСИТЕТ**

**«МИФИ»**

**ѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕѕ**

#### Институт интеллектуальных кибернетических систем

#### Кафедра № 22

**Курсовая работа по моделированию предметных сред**

**и проектированию баз данных**

*«Онлайн сервис для прямых трансляций»*

|  |  |
| --- | --- |
| Выполнил | Синкевич Г.С. |
| Группа | Б21-504 |
| Вариант | *Онлайн сервис для прямых трансляций* |
| Преподаватель | Тихомирова Д.В. |
| Проверяющий |  |
| Оценка |  |

#### Москва 2024

#### Содержание

[1. Описание предметной области 3](#_Toc166938422)

[**1.1.** **Формулировка задания** 3](#_Toc166938423)

[**1.2.** **Конкретизация предметной области** 3](#_Toc166938424)

[**1.3.** **Пользователи системы** 3](#_Toc166938425)

[**1.4.** **Сроки хранения информации** 4](#_Toc166938426)

[**1.5.** **События, изменяющие состояние Базы Данных** 4](#_Toc166938427)

[**1.6.** **Основные запросы к Базе Данных (на естественном языке)** 4](#_Toc166938428)

[2. Концептуально-информационная модель предметной области 5](#_Toc166938429)

[**2.1.** **ER-диаграмма модели** 5](#_Toc166938430)

[**2.2.** **Оценка мощностных характеристик сущностей и связей** 5](#_Toc166938431)

[3. Концептуальное проектирование 6](#_Toc166938432)

[**3.1.** **Принятые проектные соглашения** 6](#_Toc166938433)

[**3.2.** **Обоснование выбора модели базы данных** 6](#_Toc166938434)

[**3.3.** **Используемые в системе кодификаторы** 7](#_Toc166938435)

[**3.4.** **Концептуальная модель базы данных** 7](#_Toc166938436)

[4. Логическое проектирование 8](#_Toc166938437)

[**4.1.** **ER-диаграмма Базы Данных** 8](#_Toc166938438)

[**4.2. Схемы отношений Базы Данных** 9](#_Toc166938439)

[**4.3.** **Схема реляционной Базы Данных** 9](#_Toc166938440)

[**4.4.** **Схемы основных запросов на реляционной алгебре** 9](#_Toc166938441)

[5. Физическое проектирование 10](#_Toc166938442)

[**5.1. Обоснование выбора конкретной СУБД** 10](#_Toc166938443)

[**5.2. Создание Базы Данных** 10](#_Toc166938444)

[**5.3.** **Создание Таблиц** 11](#_Toc166938445)

[**5.4.** **ETL-процессы загрузки Базы Данных** 13](#_Toc166938446)

[**5.5.** **Запросы в терминах SQL** 15](#_Toc166938447)

[**5.6.** **Оценка размеров Базы данных и каждого из файлов** 17](#_Toc166938448)

[6. Приложение. Отчеты 18](#_Toc166938449)

# **Описание предметной области**

### **Формулировка задания**

Спроектировать базу данных платформы, позволяющей вести прямые трансляции, общаться в чатах, просматривать киберспортивных турниров. База данных должна отображать информацию о стримерах (люди, которые ведут прямые трансляции), зрителях, категориях, тегах и чатах.

### **Конкретизация предметной области**

Twitch - видеостриминговый сервис, специализирующийся на тематике компьютерных игр, в том числе трансляциях геймплея и киберспортивных турниров. Видео на платформе Twitch можно просматривать как в реальном времени, так и по запросу.

Видеостриминговый сервис должен предоставлять следующие возможности:

* Ведение прямых трансляций, которые могут просматривать каждый пользователь.
* Детальная настройка чата, позволяющая настраивать, кто может писать в чат (подписчики, платные подписчики или авторизованные пользователи), минимальное время между сообщениями от одного пользователя.
* Просмотра записей стримов с сохранением хронологии сообщений в чате.
* Подписываться на стримеров бесплатно и платно.
* Выбор категорий и тегов, по которым пользователи могут выбирать интересные для себя стримы.
* Модерирование чата.

Пользователь должен иметь возможности:

* Просматривать прямые трансляции.
* Писать в чат прямых трансляций, если это позволяют правила чата.
* Искать по категориям и тегам подходящие ему трансляции.
* Просматривать записи трансляции с хронологией чата.
* Подписываться платно или бесплатно на стримеров.

Стример должен иметь возможности:

* Вести прямые трансляции.
* Устанавливать правила чата.
* Смотреть историю сообщений пользователей.

Модератор должен иметь возможности:

* Смотреть историю сообщений пользователей.
* Блокировать зрителей за неподобающее поведение.

Администратор должен иметь возможности:

* Смотреть историю сообщений пользователей.
* Блокировать пользователей за неподобающее поведение.

### **Пользователи системы**

Всего есть 4 основных типов пользователей:

* Стример – человек, ведущий прямые трансляции.
* Зритель – человек, просматривающий прямые трансляции или их записи.
* Администратор – человек, который проверяет контент стримеров на нарушение правил.
* Модератор – человек, фильтрующий сообщения в чат, чтобы избежать запрещенных тем и оскорблений.

Так же зрители подразделяются по типу аккаунта:

* Подтвержденный аккаунт – был подтвержден номером телефона.
* Неподтвержденный аккаунт – не был подтвержден номером телефона.

Есть так же подразделение зрителей на прямой трансляции:

* Платный подписчик – зритель, который купил платную подписку, которая дает некоторые бонусы, которые зависят от стримера.
* Подписчик – зритель, который нажал кнопку «отслеживать».
* Посторонний зритель – зритель, который не оформил ни одну из выше перечисленных подписок.

### **Сроки хранения информации**

Записи прямых трансляций обычно хранятся 2 месяца, если их автор не указал другого. Вместе с записями хранятся сообщения чата. При удалении записи трансляции удаляются и сообщения в чате.

Вся остальная информация хранится безгранично, пока не поступит запрос на удаление от владельца данных.

### **События, изменяющие состояние Базы Данных**

Основные события в предметной области, изменяющие хранимые в БД сущности:

* Регистрация нового пользователя.
* Подписание зрителем на какого-либо стримера.
* Выдача новых прав пользователю (модератор или администратор).
* Добавление новых категорий и тегов.

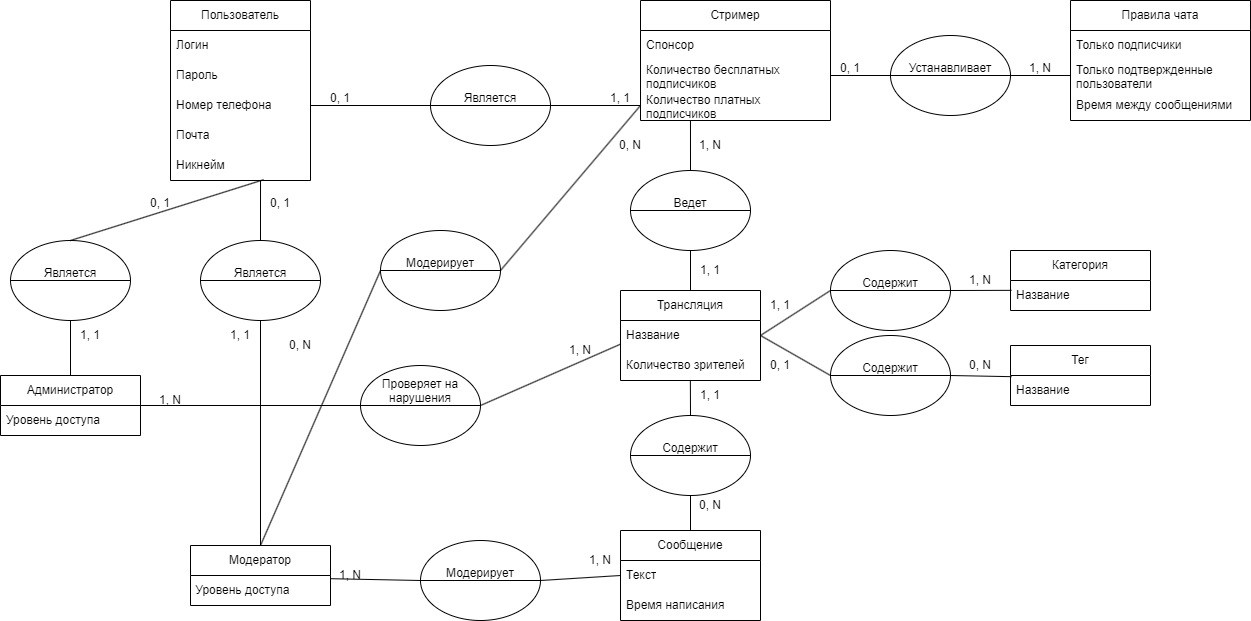
### **Основные запросы к Базе Данных (на естественном языке)**

Основные запросы к БД в разрезе управления БД:

* Получение всех подписок пользователя (платных и/или бесплатных).
* Получение сообщений в чате для выбранной записи трансляции.
* Получение сообщений в чате от выбранного пользователя для проверки их на нарушение.
* Получение прямых трансляций с определенной категорией или тегами.
* Просмотр заблокированных пользователей на определенных каналах.

# **Концептуально-информационная модель предметной области**

### **ER-диаграмма модели**



### **Оценка мощностных характеристик сущностей и связей**

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность/связь | Мощность | | |
| минимальная | средняя | максимальная |
| Пользователь | 0 | 200.000.000 | 300.000.000 |
| Стример | 0 | 4.000.000 | 8.000.000 |
| Правила чата | 0 | 4.000.000 | 8.000.000 |
| Администратор | 0 | 1.500 | 3.000 |
| Модератор | 0 | 1.000.000 | 2.000.000 |
| Трансляция | 0 | 40.000.000 | 100.000.000 |
| Сообщение | 0 | 4.000.000.000 | 10.000.000.000 |
| Категория | 0 | 5.000 | 8.000 |
| Тег | 0 | 30.000 | 50.000 |
| Является(Пользователь –Стример) | 0 | 4.000.000 | 8.000.000 |
| Является(Пользователь –Модератор) | 0 | 1.000.000 | 2.000.000 |
| Является(Пользователь –Администратор) | 0 | 1.500 | 3.000 |
| Модерирует(Модератор –Стример) | 0 | 1.000.000 | 2.000.000 |
| Устанавливает(Стример- Правила чата) | 0 | 4.000.000 | 8.000.000 |
| Ведет(Стример-Трансляция) | 0 | 40.000.000 | 100.000.000 |
| Содержит(Трансляция-Сообщение) | 0 | 4.000.000.000 | 10.000.000.000 |
| Содержит(Трансляция-Категория) | 0 | 5.000 | 8.000 |
| Содержит(Трансляция-Тег) | 0 | 30.000 | 50.000 |
| Модерирует(Модератор- Сообщения) | 0 | 4.000.000.000 | 10.000.000.000 |
| Проверяет на нарушения(Администратор - Трансляция) | 0 | 40.000.000 | 100.000.000 |

Все представленные мощностные характеристики основываются на следующих соображениях:

* Минимальное количество соответствует состоянию, когда ещё не были заведены учетные записи
* Средние значения указаны из расчёта на «нормальную» работу системы, данные были взяты из отчетности Twitch.
* Максимальные значения указаны для случая работы системы в режиме большой нагрузки. Было предположено, что максимальные значения в 2 или в 2,5 раза больше средних.
* В связах мощность выражается максимальной мощностью одной из сущностей.

Мощность Трансляций было предположено из соображений, что каждый стример проводит 10 трансляций за 2 месяца в среднем.

Мощность Сообщений было предположено из соображений, что на каждой трансляции в среднем было по 100 сообщений в чате.

# **Концептуальное проектирование**

### **Принятые проектные соглашения**

В соответствии с требованиями проектирования реляционных баз данных, мы разработали концептуальную модель базы данных (КМБД) на основе концептуальной модели предметной области (КМПО). Каждому отношению был назначен первичный ключ, который представляет собой генерируемый идентификатор. Основные преимущества использования таких ключей включают уменьшение размера базы данных при использовании их в качестве внешних ключей в реляционной модели данных, обеспечение уникальности и отсутствие необходимости изменения первичного ключа записи в любых сценариях взаимодействия с базой данных.

### **Обоснование выбора модели базы данных**

Для разработки данной системы мы приняли решение использовать реляционную модель данных, основываясь на следующих преимуществах:

* Атомарность: Гарантия того, что транзакция будет выполнена либо полностью, либо вообще не будет зафиксирована, обеспечивая целостность операций с базой данных.
* Согласованность: Каждая завершенная транзакция сохраняет целостность базы данных, обеспечивая переход из одного допустимого состояния в другое.
* Изолированность: Во время выполнения транзакции параллельные операции не влияют на ее результат, благодаря уровням изоляции и другим инструментам управления параллельностью.
* Устойчивость: После подтверждения системой выполнения транзакции изменения не подвергаются отмене из-за возможных сбоев.
* Быстродействие: Реляционные базы данных обеспечивают хорошее быстродействие как для чтения, так и для изменения данных, в отличие от некоторых других типов баз данных, где повышенное быстродействие одной операции может привести к снижению быстродействия других операций.

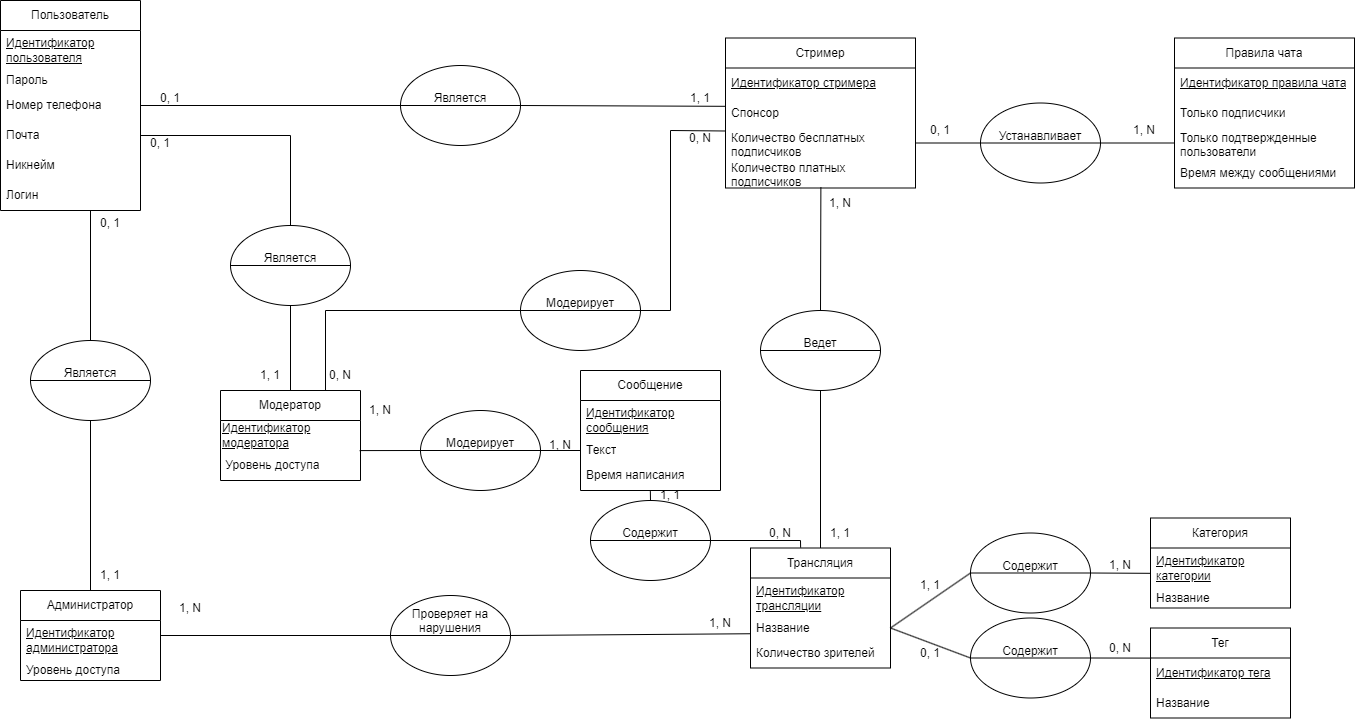
В общем, реляционная модель данных наиболее подходит для этой системы, учитывая описанные преимущества. Например, документоориентированные базы данных лучше подходят для менее структурированных данных, чем те, что используются в нашей системе, а графовые базы данных не подходят из-за непрактичности представления данных в виде графа в данном контексте.

### **Используемые в системе кодификаторы**

В проектируемой системе присутствуют следующие кодификаторы:

|  |  |
| --- | --- |
| Объект и поле | Кодификатор |
| Подписчик | 0-бесплатная подписка  1-платная подписка |
| Модератор.Уровень доступа | 0-модератор у определенного стримера  1-администратор |

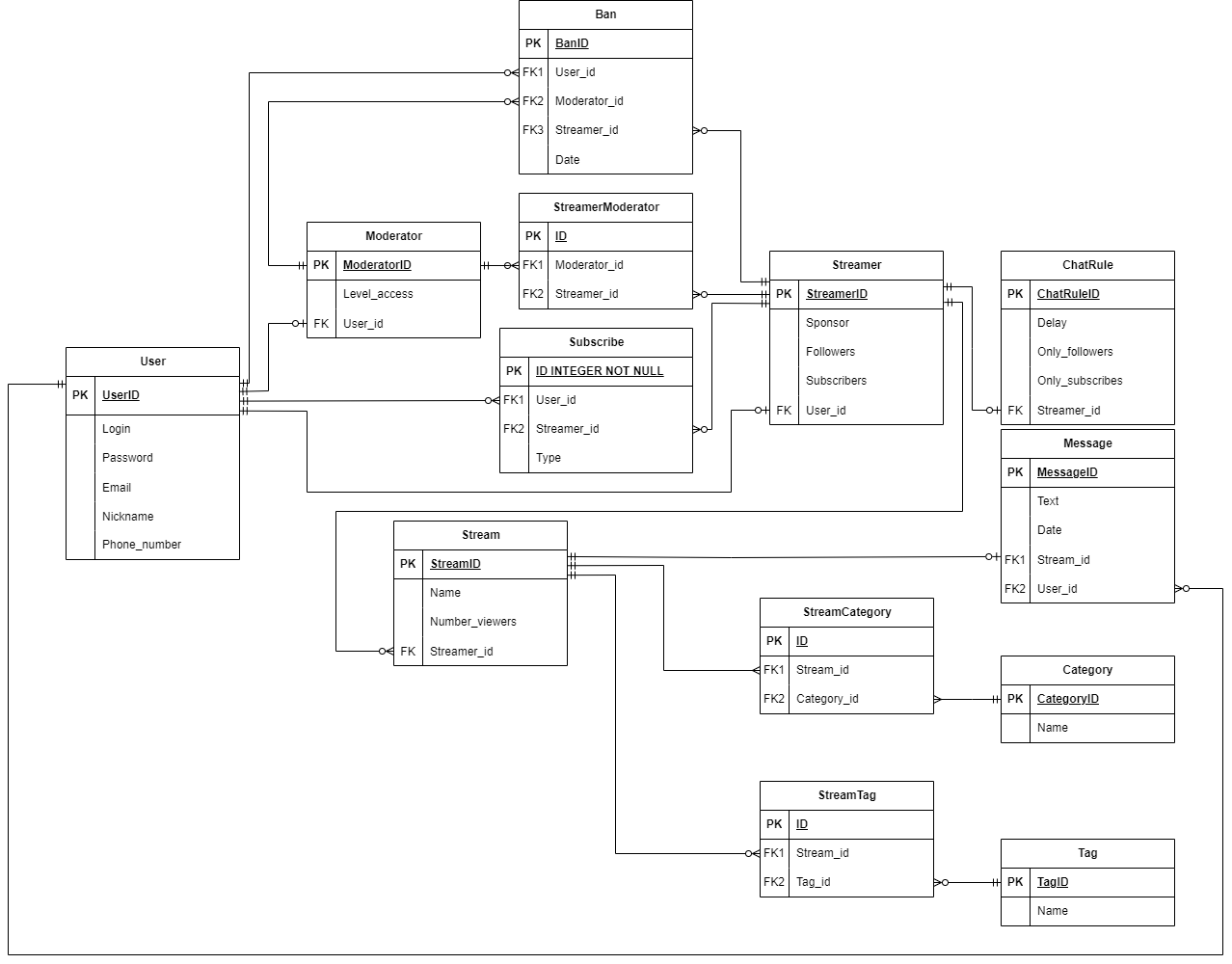
### **Концептуальная модель базы данных**



Начало формы

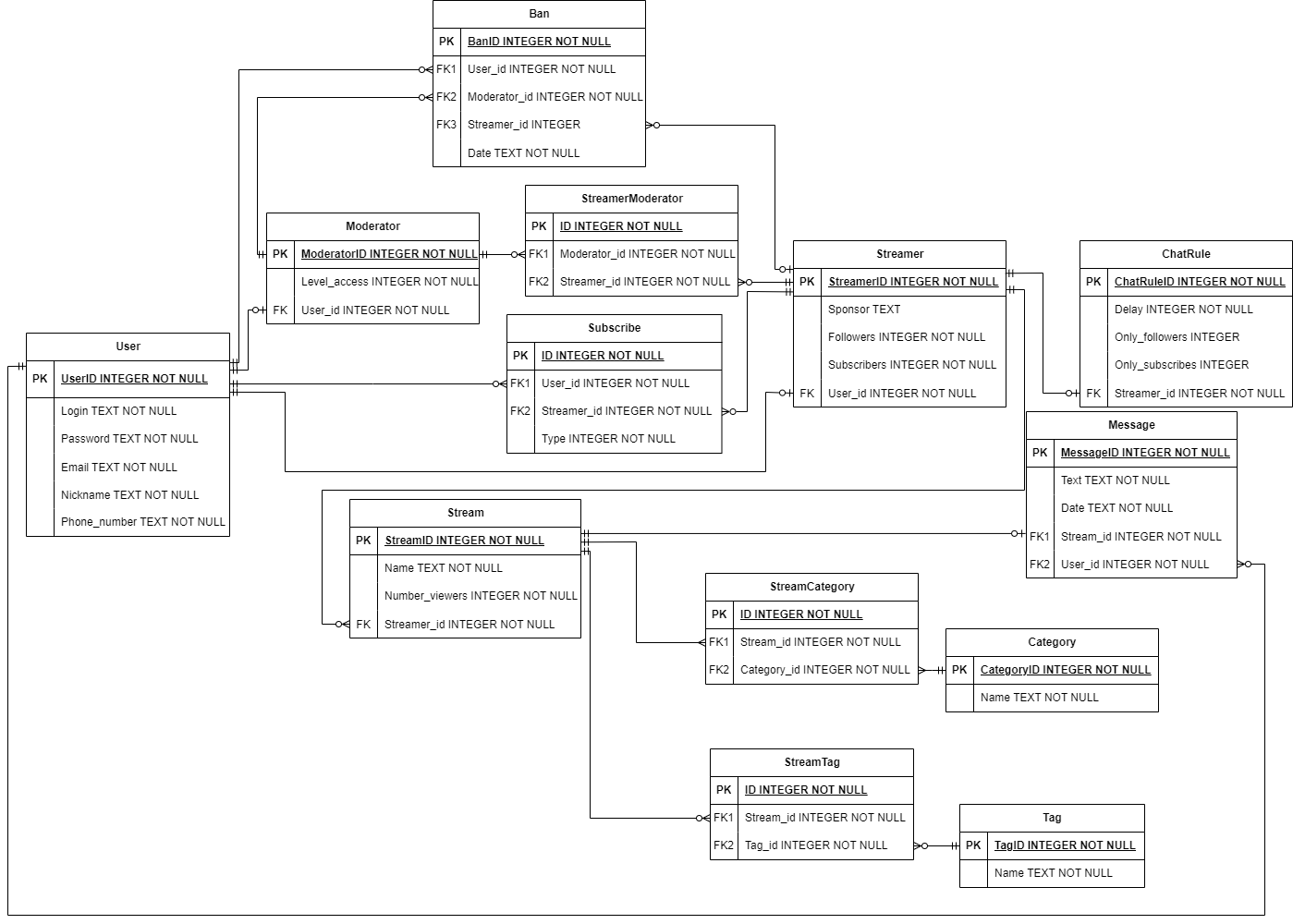
# **Логическое проектирование**

### **ER-диаграмма Базы Данных**



В ходе перехода от концептуальной модели к логической в отношения были добавлены внешние ключи и, в случае связей “многие ко многим” - дополнительные таблицы. Также, для связи пользователя и стримера были добавлены таблицы Subscribe и Ban, которые отображают уровень подписки и дату, когда был забанен пользователь соответственно. Также сущности Администратор и Модератор были объединены в единую таблицу, и разделение идет путем разных Level\_access.

### **4.2. Схемы отношений Базы Данных**



### **Схема реляционной Базы Данных**

|  |  |
| --- | --- |
| R1 | R1(#UserID, Login, Password, Email, Nickname, Phone\_number) |
| R2 | R2(#TagID, Name) |
| R3 | R3(#BanID, User\_id, Moderator\_id, Streamer\_id, Date) |
| R4 | R4(#ModeratorID, Level\_access, User\_id) |
| R5 | R5(#ID, Moderator\_id, Streamer\_id) |
| R6 | R6(#ID, User\_id, Streamer\_id, Type) |
| R7 | R7(#StreamerID, Sponsor, Followers, Subsribers, User\_id) |
| R8 | R8(#ChatRuleID, Delay, Only\_followers, Only\_subscribers, Streamer\_id) |
| R9 | R9(#MessageID, Text, Date, Stream\_id, User\_id) |
| R10 | R10(#StreamID, Name, Number\_viewers, Streamer\_id) |
| R11 | R11(#ID, Stream\_id, Categoty\_id) |
| R12 | R12(#ID, Stream\_id, Tag\_id) |
| R13 | R13(#CategoryID, Name) |

### **Схемы основных запросов на реляционной алгебре**

--- вывести пользователей, которые одновременно являются модераторами и стримерами---

R\_user\_streamer = R1[R1.UserID = R7.User\_id]R7

R\_user\_moderator\_streamer = R\_user\_streamer[R1.UserID = R7.User\_id]R7

R = R\_user\_moderator\_streamer[R1.Nickname]

--- Вывести для пользователя подписки на стримеров, у которых он не был забанен ---

R\_user\_followers = (R6[R6.User\_id = 1])[Streamer\_id]

R\_user\_ban = (R3[R3.User\_id = 1])[Streamer\_id]

R = R\_user\_followers \ R\_user\_ban

--- вывести наиболее активного подписчика для каждого стримера ---

R\_user\_count\_message\_stream = R1[R1.UserID = R9.User\_id]GroupBY(R9.Stream\_id)

R\_streamer\_stream = R7[R7.StreamerID=R10.Streamer\_id]R10

R\_user\_count\_message = R\_user\_count\_message\_stream[R9.Stream\_id=R10.StreamID]R\_streamer\_stream

R\_user\_max\_count\_message = R\_user\_count\_message[max(count)]GroupBy(R10.StreamID)

--- админы, которые забанили больше всего людей у стримеров с >10 стримов ---

R\_streamer = (R10[Count(\*), Streamer\_id]GroupByR10.Streamer\_id)[count > 10]

R\_banned\_users = R3[R3.Streamer\_id=R10.Streamer\_id]R\_streamer

R\_moder = R4[R4.Level\_access = 1]

R = R\_moder[R4.ModeratorID = R3.Moderator\_id]R\_banned\_users

-- вывести топ активных пользователей, которые не были ни разу забанены среди стримеров, проводивших трансляций в категории "Just Chatting" ---

R\_stream\_id\_category = R10[R10.StreamID = R11.Stream\_id]R11

R\_stream\_name\_category = R\_stream\_category[R11.Category\_id = R13.CategoryID]R13

R\_streamer\_just\_chatting = R\_stream\_name\_category[Streamer\_id]

R\_count\_message = R9[count(\*), User\_id]GroupBy(User\_id)

R\_banned\_users = R3[User\_id, Streamer\_id]GroupBY(Streamer\_id, User\_id)

R\_banned\_users\_true = R\_banned\_users[R3.Streamer\_id = Streamer\_id]R\_streamer\_just\_chatting

R\_true\_users = R\_count\_message \ R\_banned\_users\_true

# **Физическое проектирование**

### **5.1. Обоснование выбора конкретной СУБД**

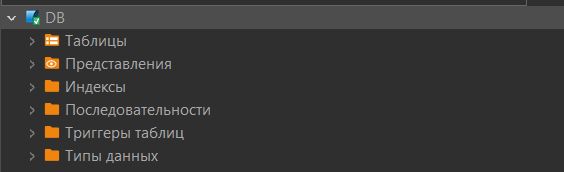
В качестве системы управления базой данных была выбрана СУБД SQLite. КО ее преимуществам относятся:

* Простота: SQLite не требует отдельного сервера или настройки. Она работает как часть вашего приложения.
* Транзакционная безопасность: SQLite поддерживает транзакционность, что обеспечивает целостность данных в случае сбоев или ошибок при работе с базой данных.
* Кроссплатформенность: SQLite поддерживается на множестве платформ, включая Windows, macOS и Linux.
* Быстродействие: SQLite обеспечивает высокую производительность при выполнении запросов и операций с данными. Он способен обрабатывать большие объемы данных быстро и эффективно.
* Поддержка ACID-свойств: SQLite обеспечивает ACID-свойства (атомарность, согласованность, изолированность, долговечность) для обеспечения надежности и целостности данных.

### **5.2. Создание Базы Данных**

База данных была создана с помощью программы DBeaver 24.0.4. DBeaver - это мощный инструмент для управления базами данных и выполнения запросов. Он предоставляет широкие возможности для работы с различными типами баз данных, включая реляционные базы данных (например, MySQL, PostgreSQL, Oracle, SQLite), NoSQL базы данных (например, MongoDB, Cassandra), а также другие форматы хранения данных.

Вот так выглядит созданная база данных:



### **Создание Таблиц**

Ниже приведены запросы для создания таблиц в базе данных:

**CREATE** **TABLE** **IF** **NOT** **EXISTS** User

(

UserID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Login TEXT **NOT** **NULL**,

Password TEXT **NOT** **NULL**,

Email TEXT **NOT** **NULL** **CHECK**(Email GLOB **'\*@\*[.]\*'**),

Nickname TEXT **NOT** **NULL**,

Phone\_number TEXT **NOT** **NULL**

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Moderator

(

ModeratorID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Level\_access **INTEGER** **NOT** **NULL**,

User\_id **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (User\_id) **REFERENCES** **User**(UserID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Streamer

(

StreamerID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Sponsor TEXT,

Followers **INTEGER** **NOT** **NULL**,

Subscribers **INTEGER** **NOT** **NULL**,

User\_id **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (User\_id) **REFERENCES** **User**(UserID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Ban

(

BanID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

**Date** **INTEGER** **NOT** **NULL**,

Moderator\_id **INTEGER** **NOT** **NULL**,

Streamer\_id **INTEGER**,

User\_id **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (User\_id) **REFERENCES** User(UserID),

**FOREIGN** **KEY** (Streamer\_id) **REFERENCES** Streamer(StreamerID),

**FOREIGN** **KEY** (Moderator\_id) **REFERENCES** Moderator(ModeratorID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** StreamerModerator

(

ID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Moderator\_id **INTEGER** **NOT** **NULL**,

Streamer\_id **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (Streamer\_id) **REFERENCES** Streamer(StreamerID),

**FOREIGN** **KEY** (Moderator\_id) **REFERENCES** Moderator(ModeratorID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Subscribe

(

ID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

User\_id **INTEGER** **NOT** **NULL**,

Streamer\_id **INTEGER** **NOT** **NULL**,

**Type** **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (Streamer\_id) **REFERENCES** Streamer(StreamerID),

**FOREIGN** **KEY** (User\_id) **REFERENCES** **User**(UserID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Stream

(

StreamID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Name TEXT **NOT** **NULL**,

Streamer\_id **INTEGER** **NOT** **NULL**,

Number\_viewers **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (Streamer\_id) **REFERENCES** Streamer(StreamerID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** ChatRule

(

ChatRuleID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Delay **INTEGER** **NOT** **NULL**,

Streamer\_id **INTEGER** **NOT** **NULL**,

Only\_followers **INTEGER**,

Only\_subscribers **INTEGER**,

**FOREIGN** **KEY** (Streamer\_id) **REFERENCES** Streamer(StreamerID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Message

(

MessageID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

User\_id **INTEGER** **NOT** **NULL**,

Stream\_id **INTEGER** **NOT** **NULL**,

**Date** **INTEGER** **NOT** **NULL**,

Text text **NOT** **NULL**,

**FOREIGN** **KEY** (Stream\_id) **REFERENCES** Stream(StreamID),

**FOREIGN** **KEY** (User\_id) **REFERENCES** **User**(UserID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Category

(

CategoryID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Name TEXT **NOT** **NULL**

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** Tag

(

TagID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Name TEXT **NOT** **NULL**

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** StreamCategory

(

ID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Category\_id **INTEGER** **NOT** **NULL**,

Stream\_id **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (Stream\_id) **REFERENCES** Stream(StreamID),

**FOREIGN** **KEY** (Category\_id) **REFERENCES** Category(CategoryID)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** StreamTag

(

ID **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,

Tag\_id **INTEGER** **NOT** **NULL**,

Stream\_id **INTEGER** **NOT** **NULL**,

**FOREIGN** **KEY** (Stream\_id) **REFERENCES** Stream(StreamID),

**FOREIGN** **KEY** (Tag\_id) **REFERENCES** Tag(TagID)

);

**CREATE** **TRIGGER** InsertSubscribe

**AFTER** **INSERT** **ON** Subscribe

**BEGIN**

**UPDATE** Streamer **SET** Subscribers = Subscribers + **NEW**.**Type**, Followers = Followers + ((**NEW**.**TYPE** - 1) \* (-1)) **WHERE** StreamerID = **NEW**.Streamer\_id;

**END**;

**CREATE** **TRIGGER** DeleteSubscribe

**AFTER** **DELETE** **ON** Subscribe

**BEGIN**

**UPDATE** Streamer **SET** Subscribers = Subscribers - **OLD**.**Type**, Followers = Followers - ((**OLD**.**TYPE** - 1) \* (-1)) **WHERE** StreamerID = **OLD**.Streamer\_id;

**END**;

### 

Созданная база данных

### **ETL-процессы загрузки Базы Данных**

#### Проектирование ETL-процессов

Для заполнения данных необходимо узнать информацию о пользователях, стримерах, прямых трансляциях. Хоть у TWITCH есть своя API, но она не дает возможность получить данные о пользователе такие как почта, логин, пароль и номер телефона. Но с помощью API можно получить названия категорий и тегов.

#### Заполнение таблиц

В процессе разработки для заполнения таблиц использовался язык Pyhton с библиотеками sqlite3 (для взаимодействия с бд), nickname\_generator (для генерации никнеймов), secmail (для генерации электронных почт), phone\_gen (для генерации номеров телефонов), password\_generator (для генерации паролей). Код для заполнения таблиц можно найти в репозитории [https://github.com/LennyLenS/6SEM\_DB](https://github.com/???), а результаты заполнения (первые несколько строк заполненных таблиц) приведены ниже.

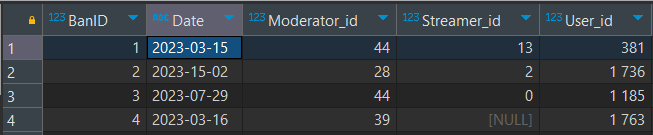


Таблица Ban

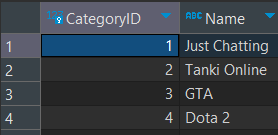


Таблица Category

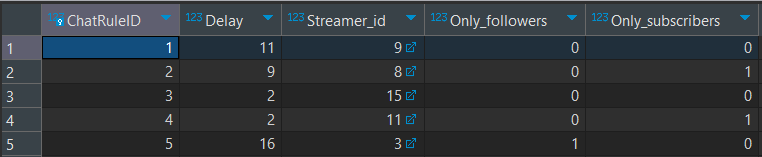


Таблица ChatRule

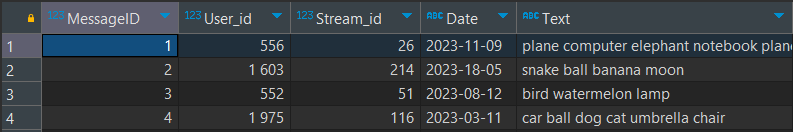


Таблица Message

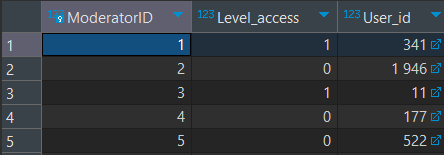


Таблица Moderator

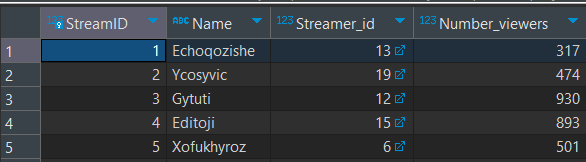


Таблица Stream

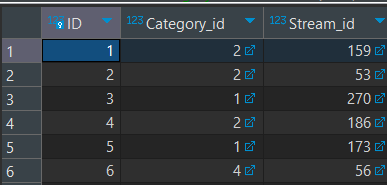


Таблица StreamCategory

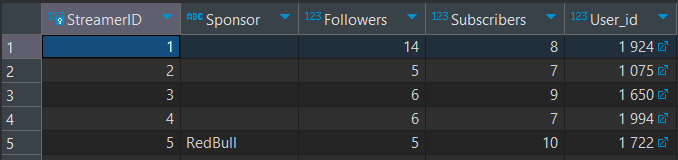


Таблица Streamer

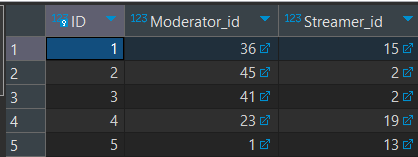


Таблица StreamerModerator



Таблица StreamTag

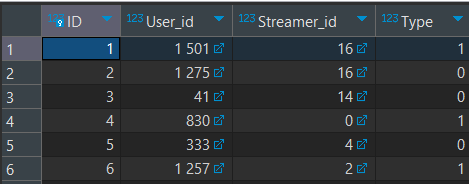


Таблица Subscribe

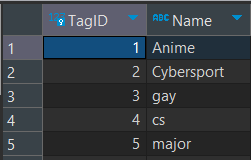


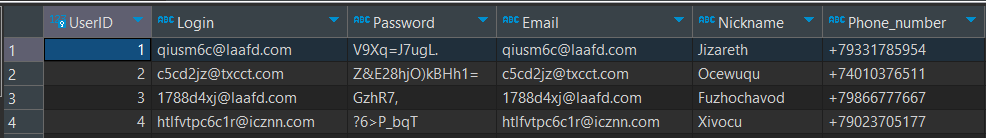
Таблица Tag 

Таблица User

### **Запросы в терминах SQL**

--- Вывести пользователей, которые одновременно являются модераторами и стримерами---

**SELECT** *u*.\* **FROM** Moderator *m*

**JOIN** Streamer *s* **ON** *m*.User\_id = *s*.User\_id

**JOIN** **User** *u* **ON** *u*.***UserID*** = *m*.User\_id;



--- Вывести для пользователя подписки на стримеров, у которых они не были забанены ---

**SELECT** *u*.***UserID***, *s*.Streamer\_id, s.**Type** **FROM** **User** *u*

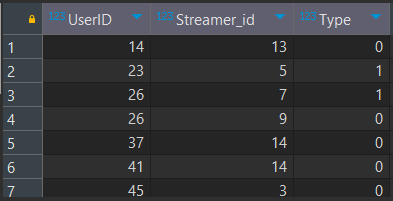
**JOIN** Subscribe *s* **ON** *s*.User\_id = *u*.***UserID***

**EXCEPT**

**SELECT** *u*.***UserID***, *b*.Streamer\_id, s.**Type** **FROM** **User** *u*

**JOIN** Subscribe *s* **ON** *s*.User\_id = *u*.***UserID***

**JOIN** Ban *b* **ON** *b*.User\_id = *u*.***UserID*** **AND** *b*.Streamer\_id = *s*.Streamer\_id;



--- Вывести наиболее активного пользователя для каждого стримера ---

**WITH** *user\_count\_message* **as** (**SELECT** *u*.UserID, *s2*.StreamerID, **Count**(\*) **as** **count** **FROM** "User" *u*

**JOIN** Message *m* **ON** *m*.User\_id = *u*.UserID

**JOIN** Stream *s* **ON** *s*.StreamID = *m*.Stream\_id

**JOIN** Streamer *s2* **ON** *s2*.StreamerID = *s*.Streamer\_id

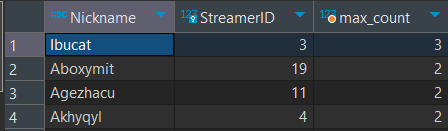
**GROUP** **BY** *u*.UserID, *s2*.StreamerID)

**SELECT** *u*.Nickname, *ucm*.StreamerID, **MAX**(**count**) **as** *max\_count* **FROM** *user\_count\_message* *ucm*

**JOIN** "User" *u* **ON** *u*.UserID = *ucm*.UserID

**GROUP** **BY** *u*.Nickname, *ucm*.StreamerID

**ORDER** **BY** ***max\_Count*** **DESC**;



--- Модераторы, которые забанили больше всего людей у стримеров с >10 стримов ---

**WITH** *stream\_count* **as** (

**SELECT** *s2*.StreamerID, **COUNT**(\*) **as** **count** **FROM** Streamer *s2*

**JOIN** Stream *srt* **ON** *srt*.Streamer\_id = *s2*.StreamerID

**GROUP** **BY** *s2*.StreamerID

**HAVING** **count** > 10

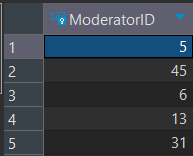
)

**SELECT** ModeratorID **FROM** Moderator *m*

**JOIN** Ban *b* **ON** *b*.Moderator\_id = *m*.ModeratorID

**JOIN** *stream\_count* *sc* **ON** *sc*.StreamerID = *b*.Streamer\_id

**WHERE** Level\_access = 0;



-- Вывести топ 5 активных пользователей, которые не были ни разу забанены среди стримеров, проводивших трансляций в категории "Just Chatting" ---

**WITH** *right\_user* **as** (

**SELECT** *b*.User\_id, *b*.Streamer\_id **FROM** Ban *b*

**EXCEPT**

**SELECT** *b*.User\_id, *b*.Streamer\_id **FROM** Stream *s*

**JOIN** StreamCategory *sc* **ON** *sc*.Stream\_id = *s*.StreamID

**JOIN** Category *c* **ON** *c*.CategoryID = *sc*.Category\_id

**JOIN** Streamer *str* **ON** *str*.StreamerID = *s*.Streamer\_id

**JOIN** Ban *b* **ON** *b*.Streamer\_id = *str*.StreamerID

**WHERE** *c*.Name = **'Just Chatting'**),

*user\_count\_message* **as** (**SELECT** *u*.UserID, *s2*.StreamerID, **Count**(\*) **as** **count** **FROM** "User" *u*

**JOIN** Message *m* **ON** *m*.User\_id = *u*.UserID

**JOIN** Stream *s* **ON** *s*.StreamID = *m*.Stream\_id

**JOIN** Streamer *s2* **ON** *s2*.StreamerID = *s*.Streamer\_id

**GROUP** **BY** *u*.UserID, *s2*.StreamerID)

**SELECT** *u*.Nickname, *ucm*.StreamerID, **MAX**(**count**) **as** *max\_count* **FROM** *user\_count\_message* *ucm*

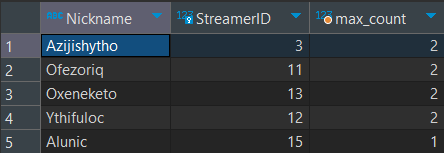
**JOIN** "User" *u* **ON** *u*.UserID = *ucm*.UserID

**JOIN** *right\_user* *ru* **ON** *ru*.User\_id = *u*.UserID

**GROUP** **BY** *u*.Nickname, *ucm*.StreamerID

**ORDER** **BY** ***max\_Count*** **DESC**

**LIMIT** 5;



### **Оценка размеров Базы данных и каждого из файлов**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Отношение | Атрибут | Тип данных | Размер, байт | Среднее количество | Объем, байт |
| User | UserID | INTEGER | 4 | 200.000.00 | 800.000.000 |
| Login | TEXT | 4-100 | 200.000.00 | 800.000.000-20.000.000.000 |
| Password | TEXT | 4-100 | 200.000.00 | 800.000.000-20.000.000.000 |
| Email | TEXT | 4-100 | 200.000.00 | 800.000.000-20.000.000.000 |
| Nickname | TEXT | 4-100 | 200.000.00 | 800.000.000-20.000.000.000 |
| Phone\_number | TEXT | 4-100 | 200.000.00 | 800.000.000-20.000.000.000 |
| Ban | BanID | INTEGER | 4 | 400.000 | 1.600.000 |
| User\_id | INTEGER | 4 | 400.000 | 1.600.000 |
| Moderator\_id | INTEGER | 4 | 400.000 | 1.600.000 |
| Streamer\_id | INTEGER | 4 | 400.000 | 1.600.000 |
| Date | TEXT | 4-100 | 400.000 | 1.600.000-40.000.000 |
| Moderator | ModeratorID | INTEGER | 4 | 1.000.000 | 4.000.000 |
| Level\_access | INTEGER | 4 | 1.000.000 | 4.000.000 |
| User\_id | INTEGER | 4 | 1.000.000 | 4.000.000 |
| StreamerModerator | ID | INTEGER | 4 | 2.000.000 | 8.000.000 |
| Moderator\_id | INTEGER | 4 | 2.000.000 | 8.000.000 |
| Streamer\_id | INTEGER | 4 | 2.000.000 | 8.000.000 |
| Subscribe | ID | INTEGER | 4 | 600.000.000 | 24.000.000.000 |
| User\_id | INTEGER | 4 | 600.000.000 | 24.000.000.000 |
| Streamer\_id | INTEGER | 4 | 600.000.000 | 24.000.000.000 |
| Type | INTEGER | 4 | 600.000.000 | 24.000.000.000 |
| Streamer | StreamerID | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Sponsor | TEXT | 4-500 | 4.000.000 | 16.000.000-2.000.000.000 |
| Followers | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Subscribers | INTEGER | 4 | 4.000.000 | 16.000.000 |
| User\_id | INTEGER | 4 | 4.000.000 | 16.000.000 |
| ChatRule | ChatRuleID | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Delay | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Only\_followers | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Only\_subscribes | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Streamer\_id | INTEGER | 4 | 4.000.000 | 16.000.000 |
| Message | MessageID | INTEGER | 4 | 4.000.000.000 | 16.000.000.000 |
| Text | TEXT | 4-400 | 4.000.000.000 | 16.000.000.000-1.600.000.000.000 |
| Date | TEXT | 4-100 | 4.000.000.000 | 16.000.000.000-400.000.000.000 |
| Stream\_id | INTEGER | 4 | 4.000.000.000 | 16.000.000.000 |
| User\_id | INTEGER | 4 | 4.000.000.000 | 16.000.000.000 |
| Stream | StreamID | INTEGER | 4 | 40.000.000 | 160.000.000 |
| Name | TEXT | 4-400 | 40.000.000 | 160.000.000-16.000.000.000 |
| Number\_viewers | INTEGER | 4 | 40.000.000 | 160.000.000 |
| Streamer\_id | INTEGER | 4 | 40.000.000 | 160.000.000 |
| StreamCategory | ID | INTEGER | 4 | 5.000 | 20.000 |
| Stream\_id | INTEGER | 4 | 5.000 | 20.000 |
| Category\_id | INTEGER | 4 | 5.000 | 20.000 |
| StreamTag | ID | INTEGER | 4 | 30.000 | 120.000 |
| Stream\_id | INTEGER | 4 | 30.000 | 120.000 |
| Tag\_id | INTEGER | 4 | 30.000 | 120.000 |
| Category | CategoryID | INTEGER | 4 | 5.000 | 20.000 |
| Name | TEXT | 4-400 | 5.000 | 20.000-20.000.000 |
| Tag | TagID | INTEGER | 4 | 30.000 | 120.000 |
| Name | TEXT | 4-400 | 30.000 | 120.000-12.000.000 |

Средний размер БД – 169.17 Гб – 2.05 Тб

# **Приложение. Отчеты**