Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

БАЗА ДАННЫХ МОБИЛЬНОГО БАНКИНГА   
БГУИР КП 1-40 01 01 018 ПЗ

Студент: гр. 051004 Пелих Н.А.

Руководитель: асс. Фадеева Е.Е.

Минск 2023

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2023 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Пелиху Никите Андреевичу–––––––––––––––––

1. Тема работы––База данных мобильного банкинга.

2. Срок сдачи студентом законченной работы ––13.12.2023 г.

3. Исходные данные к работе––СУБД MySQL, язык запросов SQL, среда визуального моделирования и проектирования Enterprise Architect.

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Анализ прототипов, литературных источников и формирование требований к проектируемой базе данных;

2. Анализ требований к базе данных и разработка функциональных требований;

3. Инфологическая модель предметной области;

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. "База данных мобильного банкинга", А1, логическая схема БД

6. Консультант по курсовой работе

Фадеева Е.Е.

7. Дата выдачи задания––03.09.2023 г.

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 15.09.2023 – 15 % готовности работы;

разделы 2, 3 к 15.10.2023 – 30 % готовности работы;

разделы 4 к 15.11.2023 – 60 % готовности работы;

раздел 5 к 10.12.2023 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 12.12.2023 – 100 % готовности работы.

Защита курсовой работы с 12.12.2023 по 21.12.2023 г.

РУКОВОДИТЕЛЬ–––––– Е.Е.Фадеева

(подпись)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc153413850)

[1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОЙ БАЗЕ ДАННЫХ 5](#_Toc153413851)

[1.1 Анализ баз данных мобильных банкингов 5](#_Toc153413852)

[1.2 Формирование требований к проектируемому программному средству 9](#_Toc153413853)

[2 АНАЛИЗ ТРЕБОВАНИЙ К БАЗЕ ДАННЫХ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 9](#_Toc153413854)

[2.1 Выбор СУБД 9](#_Toc153413855)

[2.2 Выбор средства моделирования 10](#_Toc153413856)

[2.3 Описание функциональных требований к программному средству 11](#_Toc153413857)

[2.4 Спецификация функциональных требований 11](#_Toc153413858)

[3 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ 12](#_Toc153413859)

[3.1 Описание таблиц 12](#_Toc153413860)

[3.2 Особенности нормализации 29](#_Toc153413861)

[4 ОПИСАНИЕ БИЗНЕС-ЛОГИКИ 29](#_Toc153413862)

[4.1 Представления 29](#_Toc153413863)

[4.2 Процедуры и функции 30](#_Toc153413864)

[4.3 Триггеры 31](#_Toc153413865)

[4.4 Ограничения 32](#_Toc153413866)

[5 ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ 32](#_Toc153413867)

[ЗАКЛЮЧЕНИЕ 33](#_Toc153413868)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc153413869)

[ПРИЛОЖЕНИЕ А 36](#_Toc153413870)

[ПРИЛОЖЕНИЕ Б 37](#_Toc153413871)

# ВВЕДЕНИЕ

Каждый пользователь мобильного банкинга интересуется вопросами финансов и управления своими счетами, и в наше время базы данных стали одним из основных инструментов для обеспечения удобства и безопасности финансовых операций.

Базы данных, в частности, выступают в роли надежных центров, где банковские специалисты хранят и обрабатывают различные данные, такие как информация о клиентах, транзакциях, кредитах и статистика использования приложения.

Однако успешное функционирование мобильного банкинга требует не только надежной защиты и обработки данных, но и эффективной системы управления контентом. В этом контексте, создание и оптимизация базы данных становятся ключевыми элементами обеспечения стабильной работы и привлекательности банковского приложения.

База данных для мобильного банкинга может быть разработана с использованием системы управления базами данных (СУБД), например, MySQL. Это позволяет обеспечить удобное хранение и быстрый доступ к различным данным, таким как балансы, транзакции, кредитные лимиты и другие финансовые показатели.

Мобильный банкинг может привлечь больше пользователей, если предоставит удобный интерфейс для управления счетами, категоризацию транзакций, а также возможность получения уведомлений о движении средств и состоянии счета. В этом контексте, база данных становится ключевым инструментом для организации и хранения всей необходимой информации.

Управление базой данных мобильного банкинга требует надежных механизмов записи и обработки данных. Такие функциональности, как удобный поиск, категоризация, возможность добавления и редактирования транзакций, а также взаимодействие с пользователями, могут быть реализованы с использованием соответствующих запросов к базе данных.

Цель данного курсового проекта заключается в разработке базы данных, специально адаптированной под потребности мобильного банкинга. Это включает в себя создание структуры для хранения информации о клиентах, счетах, транзакциях, кредитах и других элементах, необходимых для создания динамичного и интерактивного банковского приложения.

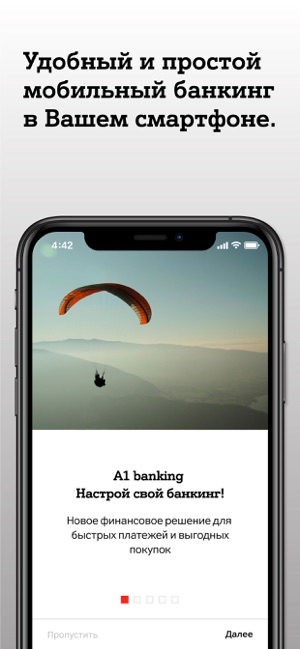
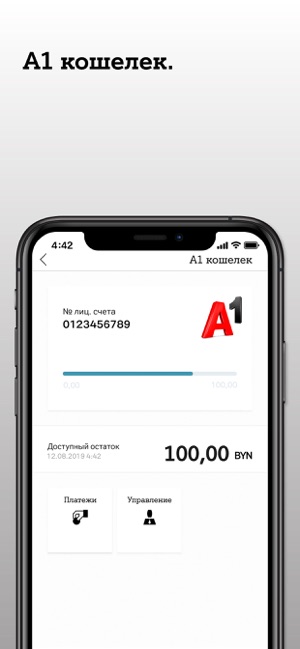
# АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОЙ БАЗЕ ДАННЫХ

## **1.1 Анализ баз данных мобильных банкингов**

Провести полный анализ баз данных мобильных банкингов представленных на рынке очень трудно, как минимум потому что любое приложение работающую с финансами имеет крайнюю защищенность и про их внутреннюю компоновку мало чего известно.

Тем не менее, некоторые сущности баз данных все-таки отражены в пользовательском интерфейсе. В большинстве случаев пользователи могут просмотреть огромный спектр предоставляемых услуг, да и функциональность у таких приложений в основном очень похожа.

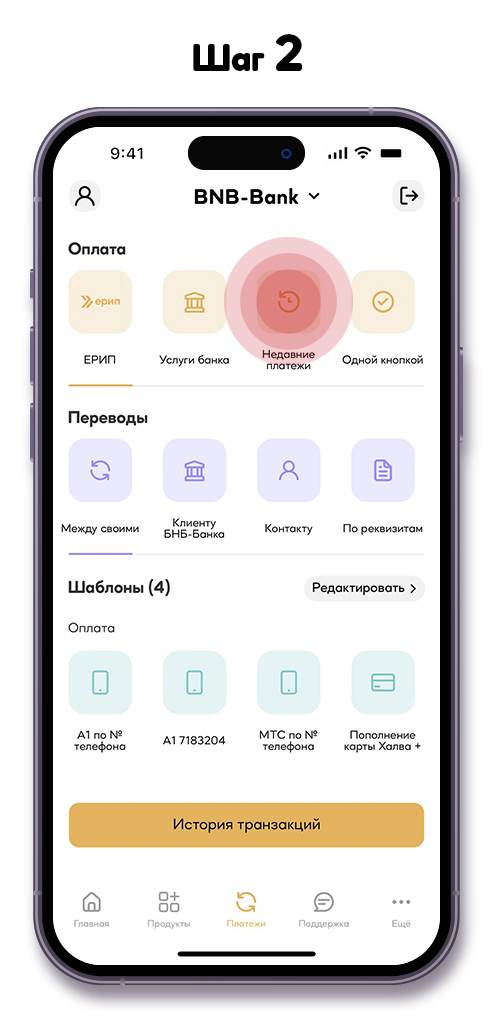
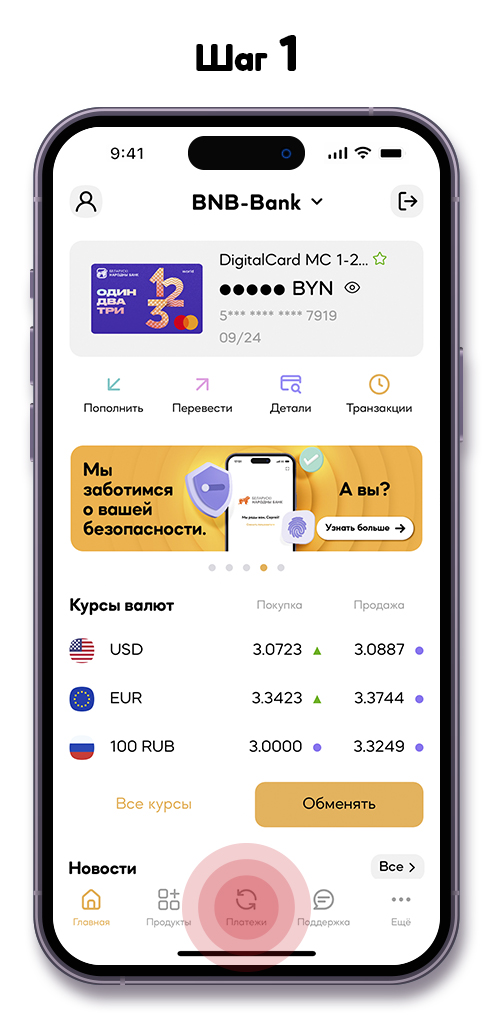
Рассмотрим подробнее несколько реально существующих мобильных банкингов.

  
  
Рисунок 1.1 – Интерфейс мобильного банкинга А1

Интерфейс мобильного банкинга А1, представленный на рисунке 1.1, имеет как положительные стороны, так и отрицательные.

К положительным сторонам можно отнести удобную панель управления, что позволяет пользователям быстро находить необходимую информацию, что увеличивает производительность и сокращает временные затраты.

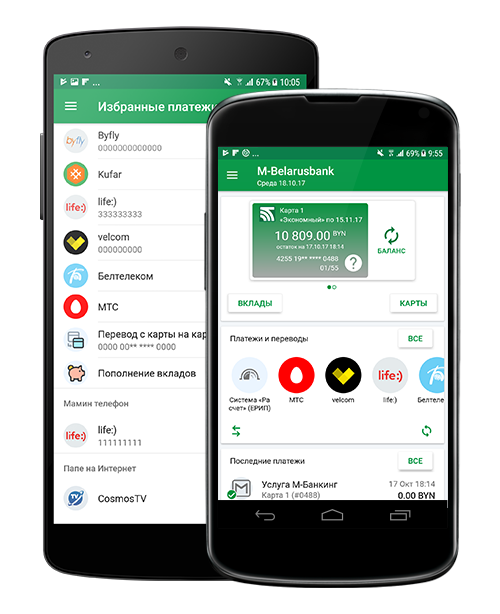
К отрицательным сторонам мобильного банкинга, представленного на рисунке 1.1, можно отнести, например, то, что мобильный банкинг имеет привязку к номеру телефона внутри сети А1, а значит пользователи с других мобильных провайдеров не смогут им пользоваться.

  
  
Рисунок 1.2 – Интерфейс мобильного банкинга БНБ-банка

На рисунке 1.2 представлена еще одна реализация мобильного банкинга.

Преимуществом данной реализации является приятный пользовательский интерфейс, наличие широкого и нужного функционала и красивая цветовая гамма.

Определенным недостатком веб-приложения, представленного на рисунке 1.2., является отсутствие возможности пользователям других банков подключаться пользователям других банков. Помимо этого минусом является возможность утечки пользовательских данных.

  
  
Рисунок 1.3 – Интерфейс мобильного банкинга Беларусбанка

На рисунке 1.3 представлена одна из самых популярных и самых старых примеров мобильного банкинга в нашей стране.

Определенным преимущественном данного приложения является простой и понятный пользовательский интерфейс, на главный экран вынесены все необходимые пользователю услуги и сервисы.

Определенным недостатком данного банкинга является устаревший дизайн, отсутствие возможности кастомизировать приложение.Также приложение не лишено недостатков свойственных всем приложением его рода: уязвимость пользовательских данных. Также в приложение реализован крайне незащищенный способ идентификации пользователя.

Вывод: изучив аналоги, приведенные выше, можно заметить их идентичность. Каждый из аналогов имеет как преимущества, так и недостатки, что позволяет после анализа продумать не только функционал, но и нюансы будущей базы данных.

## **1.2 Формирование требований к проектируемому программному средству**

В рамках данного курсового проекта планируется разработка базы данных для мобильного банкинга. Основные функциональные возможности базы данных:

− авторизация обычных пользователей;

− просмотр текущего баланса на карте;

− оформление платежей по кредитам;

− просмотр тематических новостей;

− просмотр тематических предложений;

− возможность добавлять банковские/кредитные карты;

− возможность управлять инвестиционным пакетом;

− возможность блокировки карт;

− возможность совершать платежи;

− возможность покупать/продавать криптовалюты;

− перевод с карты на карту;

# АНАЛИЗ ТРЕБОВАНИЙ К БАЗЕ ДАННЫХ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

## **2.1 Выбор СУБД**

В качестве СУБД была выбрана MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, LAMP. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит и библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.

Максимальный размер таблицы в MySQL ограничен её типом. В общем случае тип MyISAM ограничен предельным размером файла в файловой системе операционной системы. Например, в NTFS этот размер теоретически может быть до 32 эксабайт. В случае InnoDB одна таблица может храниться в нескольких файлах, представляющих единое табличное пространство. Размер последнего может достигать 64 терабайт. В отличие от MyISAM, в InnoDB имеется значительное ограничение на количество столбцов, которое можно добавить в одну таблицу. Размер страницы памяти по умолчанию составляет 16 килобайт, из которых под данные отведено 8123 байта. Размер указателя на динамические поля составляет 20 байт. Таким образом, в случае использования динамического формата строки (ROW\_FORMAT=DYNAMIC)

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в том числе LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Популярность MySQL обусловлена простотой СУБД, богатыми функциональными возможностями, встроенными системами безопасности, расширенными возможностями настройки и полной поддержкой SQL-языка. СУБД не перегружена функциями, поэтому работает быстрее и надежнее многих аналогов.

## **2.2 Выбор средства моделирования**

Главным инструментом визуального моделирования и проектирования был выбран Sparx Systems Enterprise, основанный на OMG UML.

Основные функциональные возможности включают в себя:

– Управление требованиями;

– Моделирование и анализ бизнес-процессов;

– Симуляция процессов (например, методом Монте-Карло);

– Проектирование систем и генерация кода (поддерживается генерация на более чем 10 языках высокого уровня, а также на языках проектирования аппаратного обеспечения (HDL));

– Управление проектами и командами;

– Управление тестами;

– Моделирование данных.

Именно возможность моделирования данных и будет эксплуатироваться в процессе разработки курсового проекта.

Enterprise Architect поддерживает моделирование данных от концептуального до физического уровня, прямое и обратное проектирование схем баз данных и трансформация логических диаграмм (независимых от платформы) в физические (зависят от платформы).

## **2.3 Описание функциональных требований к программному средству**

База данных должна отражать работу мобильного банкинга. На основании требований, выдвинутых в первом разделе, можно сформировать следующие функциональные требования:

– Необходимо хранить данные пользователей;

– Необходимо хранить данные о картах привязанных к аккаунту пользователя;

– Необходимо хранить историю транзакций;

– Необходимо обеспечить хранения данных, из которых формируется годовые/месячные отчеты;

– Необходимо хранить информацию о купленных ценных бумагах;

– Необходимо хранить историю платежей по кредитам;

– Необходимо хранить историю транзакций в криптовалюте;

## **2.4 Спецификация функциональных требований**

Персональные данные пользователя включая информацию о картах, транзакциях, паспорте доступны только пользователю.

Данные о транзакциях с карты являются секретной информацией и доступны только пользователю. Пользователь может в особом порядке запросить свою личную историю транзакций с карты.

Данные о купленных и проданных криптовалютах доступны только пользователю после авторизации. Пользователь может запросить отчетность о всех транзакциях связанных с криптовалютами после удостоверения личности.

Стоимость и текущий курс криптовалют доступны любому пользователю после авторизации. Эти данные едины для всех пользователей.

Данные о платежах по кредитам доступны только для авторизованных пользователей и являются персональной информацией. Пользователь после того как предоставит документ удостоверяющий личность может запросить историю платежей по кредитам.

В базе данных ведется учет всей деятельности пользователей с целью того, чтобы пользователю можно было делать лимитированные предложения и показывать таргетированную рекламу основываясь на предпочтениях пользователя.

Персональные данные пользователь может менять в личном кабинете. Любые изменения будут запрашивать удостоверения личности.

Пользователи должны иметь возможность искать предложения по выгодным предложениям на рынке ценных и криптовалют. Им должна быть показана самая актуальная информация на рынке.

Авторизация и аутентификация пользователей должна осуществляться с использованием безопасных методов, обеспечивая защиту личной информации.

Необходимо реализовать возможность пользователям добавлять новости в закладки и лайкать их.

Вся история транзакций может быть доступна банковским работникам после соответственного разрешения. Без этого разрешения история транзакций доступна только пользователю.

Пользователь может идентифицировать себя по номеру телефона и паролю, который он создал при регистрации.

# ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

## **3.1 Описание таблиц**

Логическая модель базы данных продемонстрирована в приложении А. Таблица пользователей предназначена для хранения всей информации о картах, транзакциях и тд.

Таблица 3.1 – Отношение account

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_account | Идентификатор аккаунта | INT NOT NULL | PK |
| is\_system | Проверка системный ли аккаунт | BOOL NOT NULL |  |
| Id\_credentials | Идентификатор логина и пароля | INT NOT NULL | FK (отношение credentials) |

Таблица 3.2 – Отношение account status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| account\_id\_account | Идентификатор аккаунта | INT NOT NULL | FK(отношение account) |
| status\_id\_status\_list | Идентификатор статуса | INT NOT NULL | FK (отношение account status list) |
| account\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.3 – Отношение account status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_account\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| account\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.4 – Отношение credentials

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id | Идентификатор паспорта и пароля | INT NOT NULL | PK |
| login | Логин | VARCHAR(50) NOT NULL |  |
| password\_hash | Пароль после хэширования | VARCHAR(50) NOT NULL |  |

Таблица 3.5 – Отношение news

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_news | Идентификатор новости | INT NOT NULL | PFK |
| title | Заголовок | VARCHAR(50) NOT NULL |  |
| description | Описание | VARCHAR(50) |  |
| release\_date | Дата выхода | DATETIME NOT NULL |  |

Таблица 3.6 – Отношение promotion

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_promotion | Идентификатор предложения | INT NOT NULL | PFK |
| name | Заголовок | VARCHAR(50) NOT NULL |  |
| description | Описание | VARCHAR(50) |  |
| start\_date | Дата начала | DATETIME NOT NULL |  |
| end\_date | Дата конца | DATETIME NOT NULL |  |

Таблица 3.7 – Отношение crypto wallet

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_crypto\_wallet | Идентификатор крипто кошелька | INT NOT NULL | PK |
| private\_key | Приватный ключ | VARCHAR(50) NOT NULL |  |
| public\_key | Публичный ключ | VARCHAR(50)  NOT NULL |  |
| account | Идентификатор аккаунта | INT NOT NULL | FK(отношение account) |

Таблица 3.8 – Отношение crypto wallet status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| wallet\_id\_wallet | Идентификатор аккаунта | INT NOT NULL | FK(отношение crypto wallet) |
| status\_id\_status | Идентификатор статуса | INT NOT NULL | FK (отношение crypto wallet status list) |
| wallet\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.9 – Отношение crypto wallet status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_wallet\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| wallet\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.10 – Отношение crypto currency in wallet

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_crypto\_currency | Идентификатор крипто валюты в кошельке | INT NOT NULL | PK |
| count | Счет | INT NOT NULL |  |
| wallet | Идентификатор кошелька | INT NOT NULL | FK (отношение crypto wallet) |
| currency | Имя крипто валюты | INT NOT NULL | FK (отношение crypto currency) |

Таблица 3.11 – Отношение crypto currency

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_crypto\_currency | Идентификатор крипто валюты | INT NOT NULL | PK |
| name | Имя крипто валюты | VARCHAR(50) NOT NULL |  |
| token | Уникальный токен | VARCHAR(50) NOT NULL |  |
| contract\_number | Номер контракта | VARCHAR(50) NOT NULL |  |
| contract\_price | Стоимость контракта | VARCHAR(50) NOT NULL |  |
| capitalization | Капитализация | VARCHAR(50) |  |

Таблица 3.12 – Отношение crypto exchange rate

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_crypto\_exchange | Идентификатор обмена крипто валюты | INT NOT NULL | PK |
| currency\_from | Исходная валюта | INT NOT NULL | FK (отношение crypto currency) |
| currency\_to | Входная валюта | INT NOT NULL | FK (отношение crypto currency) |
| rate | Курс | FLOAT(10) NOT NULL |  |

Таблица 3.13 – Отношение crypto transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_crypto\_transaction | Идентификатор крипто транзакции | INT NOT NULL | PK |
| wallet\_from | Исходный кошелек | VARCHAR(50) NOT NULL |  |
| wallet\_to | Кошелек получатель | VARCHAR(50) NOT NULL |  |
| sum | Сумма | VARCHAR(50) NOT NULL |  |
| date | Дата | DATE NOT NULL |  |
| currency | Идентификатор валюты | INT NOT NULL | FK (отношение crypto currency) |

Таблица 3.14 – Отношение crypto transaction status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| status\_id\_status\_list | Идентификатор статуса | INT NOT NULL | FK(отношение crypto transaction status list  ) |
| crypto\_transaction\_id\_transaction | Идентификатор транзакции | INT NOT NULL | FK (отношение crypto transaction) |
| crypto\_transaction\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.15 – Отношение crypto transaction status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_crypto\_transaction\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| crypto\_transaction\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.16 – Отношение credit card

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_credit\_card | Идентификатор кредитной карты | INT NOT NULL | PK |
| account | Идентификатор аккаунта | INT NOT NULL | FK(отношение card) |
| owner\_name | Имя владельца | VARCHAR(50) NOT NULL |  |
| cvv | Уникальный айди | INT NOT NULL |  |
| number | Номер карты | VARCHAR(50) NOT NULL |  |
| expire\_month | Месяц истечения годности | DATETIME NOT NULL |  |

Таблица 3.17 – Отношение credit card status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| status\_id\_status | Идентификатор статуса | INT NOT NULL | FK(отношение credit card status list) |
| cr\_card\_id\_card | Идентификатор карты | INT NOT NULL | FK (отношение credit card) |
| cr\_card\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.18 – Отношение credit card status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_credit\_card\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| cr\_card\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.19 – Отношение payment

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_payment | Идентификатор платежа | INT NOT NULL | PK |
| date | Дата | DATE NOT NULL |  |
| number\_to | Номер карты | VARCHAR(50) NOT NULL |  |
| sum | Сумма | INT NOT NULL |  |
| card | Идентификатор кредитной карты | INT NOT NULL | FK(отношение card) |

Таблица 3.20 – Отношение payment status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| status\_id\_status\_list | Идентификатор статуса | INT NOT NULL | FK(отношение payment status list) |
| payment\_id\_payment | Идентификатор платежа | INT NOT NULL | FK (отношение payment) |
| payment\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.21 – Отношение payment status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_credit\_card\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| cr\_card\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.22 – Отношение investment account

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_investment\_account | Идентификатор инвест. аккаунта | INT NOT NULL | PK |
| number | Номер | VARCHAR(50) NOT NULL |  |
| account | Идентификатор аккаунта | INT NOT NULL | FK (отношение account) |

Таблица 3.23 – Отношение investment account status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| invest\_id\_invest | Идентификатор инвест. аккаунта | INT NOT NULL | FK(отношение investment account status list) |
| status\_id\_status | Идентификатор статуса | INT NOT NULL | FK (отношение investment account) |
| invest\_acc\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.24 – Отношение investment account status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_invest\_acc\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| invest\_acc\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.25 – Отношение stock on account

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_stock\_on\_account | Идентификатор акции на аккаунте | INT NOT NULL | PK |
| count | Количество | INT NOT NULL |  |
| account | Идентификатор аккаунта | INT NOT NULL | FK (отношение account) |
| stock | Идентификатор акции | INT NOT NULL | FK (отношение stock) |

Таблица 3.26 – Отношение stock

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_stock | Идентификатор акции | INT NOT NULL | PK |
| name | Имя акции | VARCHAR(50) NOT NULL |  |
| company | Имя компании | VARCHAR(50) NOT NULL |  |

Таблица 3.27 – Отношение stock exchange rate

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_stock\_exchange\_rate | Идентификатор обменного курса | INT NOT NULL | PK |
| stock | Имя акции | INT NOT NULL | FK (отношение stock) |
| price | Цена акции | INT NOT NULL |  |

Таблица 3.28 – Отношение stock transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_stock\_transaction | Идентификатор транзакции | INT NOT NULL | PK |
| account\_from | Исходящий аккаунт | VARCHAR(50) NOT NULL |  |
| account\_to | Входящий аккаунт | VARCHAR(50) NOT NULL |  |
| count | Количество | INT NOT NULL |  |
| stock | Идентификатор акции | INT NOT NULL | FK (отношение stock) |

Таблица 3.29 – Отношение stock transaction status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| status\_id\_status\_list | Идентификатор статуса | INT NOT NULL | FK(отношение stock transaction status list) |
| stock\_tr\_id\_tr | Идентификатор акции | INT NOT NULL | FK (отношение stock transaction) |
| stock\_tr\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.30 – Отношение stock transaction status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_stock\_tr\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| stock\_tr\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.31 – Отношение card

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_card | Идентификатор карты | INT NOT NULL | PK |
| number | Номер карты | VARCHAR(50) NOT NULL |  |
| expire\_month | Срок годности | DATE NOT NULL |  |
| owner\_name | Имя владельца | VARCHAR(50) NOT NULL |  |
| cvv | Уникальный номер | INT NOT NULL |  |
| balance | Баланс | INT NOT NULL |  |
| account | Идентификатор аккаунта | INT NOT NULL | FK (отношение account) |
| currency | Идентификатор валюты | INT NOT NULL | FK (отношение currency) |

Таблица 3.32 – Отношение card status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| status\_id\_status\_list | Идентификатор статуса | INT NOT NULL | FK(отношение card status list) |
| card\_id\_card | Идентификатор карты | INT NOT NULL | FK (отношение card) |
| card\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.33 – Отношение card status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_card\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| card\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.34 – Отношение transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_transaction | Идентификатор транзакции | INT NOT NULL | PK |
| number\_from | Номер исходный | VARCHAR(50) NOT NULL |  |
| number\_to | Срок итоговый | VARCHAR(50) NOT NULL |  |
| sum | Сумма | INT NOT NULL |  |
| date | Дата | DATE NOT NULL |  |
| currency | Идентификатор валюты | INT NOT NULL | FK (отношение currency) |
| card | Идентификатор карты | INT NOT NULL | FK (отношение card) |

Таблица 3.35 – Отношение transaction status

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| status\_id\_status\_list | Идентификатор статуса | INT NOT NULL | FK(отношение transaction status list) |
| transaction\_id\_transaction | Идентификатор транзакции | INT NOT NULL | FK (отношение transaction) |
| transaction\_status\_last\_update | Последнее обновление | DATETIME NOT NULL |  |

Таблица 3.36 – Отношение transaction status list

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_transaction\_status\_list | Идентификатор статуса | INT NOT NULL | PK |
| transaction\_status\_name | Имя статуса | VARCHAR(50) NOT NULL |  |

Таблица 3.37 – Отношение currency

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_currency | Идентификатор валюты | INT NOT NULL | PK |
| name | Имя | VARCHAR(50) NOT NULL |  |
| short\_name | Короткое имя | VARCHAR(10) NOT NULL |  |

Таблица 3.38 – Отношение exchange rate

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_exchange\_rate | Идентификатор обмена | INT NOT NULL | PK |
| currency\_from | Исходная валюта | INT NOT NULL | FK (отношение currency) |
| currency\_to | Нужная валюта | INT NOT NULL | FK (отношение currency) |
| rate | Курс | FLOAT(10) NOT NULL |  |

Таблица 3.39 – Отношение account owner

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_owner | Идентификатор владельца | INT NOT NULL | PK |
| phone\_number | Номер телефона | VARCHAR(50) NOT NULL |  |
| creation\_date | Дата | DATETIME NOT NULL |  |
| passport | Идентификатор паспорта | INT NOT NULL | FK (отношение passport) |
| owner\_type | Идентификатор типа владельца | INT NOT NULL | FK (отношение owner\_type) |
| company | Идентификатор компании | INT | FK (отношение company) |

Таблица 3.40 – Отношение account owner type

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_account\_owner\_type | Идентификатор типа владельца | INT NOT NULL | PK |
| name | Имя | VARCHAR(50) NOT NULL |  |
| description | Короткое имя | VARCHAR(50) |  |

Таблица 3.41 – Отношение office

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_office | Идентификатор офиса | INT NOT NULL | PK |
| name | Имя | VARCHAR(50) NOT NULL |  |
| manager | Идентификатора менеджера | INT NOT NULL | FK (отношение account owner) |
| address | Идентификатор адреса | INT NOT NULL | FK (отношение address) |

Таблица 3.42 – Отношение company

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_company | Идентификатор компании | INT NOT NULL | PK |
| registration\_number | Регистрационный номер | VARCHAR(50) NOT NULL |  |
| name | Имя компании | VARCHAR(50) NOT NULL |  |
| owner | Владелец | INT NOT NULL |  |
| address | Идентификатор адреса | INT NOT NULL | FK (отношение address) |

Таблица 3.43 – Отношение address

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_address | Идентификатор адреса | INT NOT NULL | PK |
| address\_city | Город | VARCHAR(50) NOT NULL |  |
| address\_street | Улица | VARCHAR(50) NOT NULL |  |
| address\_building | Здание | VARCHAR(50) NOT NULL |  |
| address\_apartment | Квартира | VARCHAR(50) NOT NULL |  |
| address\_country | Страна | VARCHAR(50) NOT NULL |  |

Таблица 3.44 – Отношение passport

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_passport | Идентификатор паспорта | INT NOT NULL | PK |
| name | Имя | VARCHAR(50) NOT NULL |  |
| last\_name | Фамилия | VARCHAR(50) NOT NULL |  |
| identification\_number | Идентифик. номер | VARCHAR(50) NOT NULL |  |
| passport\_num | Номер паспорта | VARCHAR(50) NOT NULL |  |
| sex | Пол | VARCHAR(50) |  |
| birth\_date | Дата рождения | DATE NOT NULL |  |
| address | Идентификатор адреса | INT NOT NULL | FK (отношение address) |
| passport\_issuer | Идентификатор кем выдан паспорт | INT NOT NULL | FK (отношение passport issuer) |

Таблица 3.45 – Отношение passport issuer

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Назначение**  **атрибута** | **Тип данных**  **атрибута** | **Примечание** |
| id\_passport\_issuer | Идентификатор кем выдан паспорт | INT NOT NULL | PK |
| name | Имя | VARCHAR(50) NOT NULL |  |
| address | Идентификатор адреса | INT NOT NULL | FK (отношение address) |

## **3.2 Особенности нормализации**

**3.2.1** Нормализация представляет собой процесс разделения переменных отношения на несколько новых, находящихся в более высоких нормальных формах.

Отношение находится в первой нормальной форме (1НФ), если все его атрибуты атомарны, то есть ни один из них не может быть разделен на более простые атрибуты, соответствующие другим свойствам описываемой сущности.

Вторая нормальная форма (2НФ) предполагает, что каждый столбец, не являющийся ключом, должен зависеть от первичного ключа. Третья нормальная форма (3НФ) подразумевает, что каждый столбец, не являющийся ключом, должен зависеть только от первичного ключа.

Нормальная форма Бойса-Кодда (BCNF) является более строгой версией третьей нормальной формы. Четвертая нормальная форма (4НФ) применяется для устранения многозначных зависимостей, где столбец с первичным ключом имеет связь один-ко-многим со столбцом, не являющимся ключом.

Пятая нормальная форма (5НФ) предполагает разделение таблиц на более мелкие таблицы для устранения избыточности данных. Разбиение продолжается до тех пор, пока нельзя воссоздать оригинальную таблицу путем объединения малых таблиц.

Шестая нормальная форма предостерегает от добавления недопустимых данных, устанавливая ограничения на уровне отношений между таблицами, зависящих только от ограничений ключа и ограничений домена, где домен представляет набор допустимых значений для столбца.

**3.2.2** В спроектированной базе данных все таблицы отвечают требованиям первой нормальной формы. Также все таблицы соответствуют второй нормальной форме, поскольку все неключевые атрибуты полностью функционально зависят от потенциальных ключей в своих отношениях.

# ОПИСАНИЕ БИЗНЕС-ЛОГИКИ

Дамп структуры базы данных представлен в приложении А.

## **4.1 Представления**

Представление – виртуальная производная переменная отношения, значением которой является результат вычисления реляционного выражения (выполнения запроса), заданного при создании представления (такое выражение должно ссылаться хотя бы на одну переменную отношения).

Материализованное представление – производная переменная отношения, значением которой является сохранённый результат заранее вычисленного реляционного выражения, заданного при создании материализованного представления. Повторное вычисление и сохранение полученного результата происходит согласно правилам, определённым при создании материализованного представления.

Представления обладают следующими преимуществами:

* Упрощение выполнения запросов. Представление может быть построено на SQL-запросе любой сложности. Вместо работы с очень сложным запросом, который может занимать десятки и сотни строк, достаточно написать что-то наподобие SELECT \* FROM представление, чтобы получить тот же результат;
* Упрощение бизнес-логики;
* Минимум накладных расходов. Представление практически не занимает места в базе данных, потому даже создание сотен и тысяч представлений не приводит к сколь бы то ни было ощутимому увеличению размера базы данных;
* Производительность. В случае использования материализованных представлений мы получаем возможность многократно использовать результаты выполнения SQL-запроса;
* Безопасность. Представления являются объектами базы данных, потому к ним применимы все механизмы СУБД по контролю прав доступа.

Приведем пример представления, которую показывает информацию о карте привязанной к аккаунту:

CREATE VIEW GET\_CARD\_INFO AS

SELECT owner\_name AS Владелец, phone\_number AS НомерТелефона, cvv AS УникальныйНомер, short\_name As ИмяВалюты

FROM card

JOIN `account owner` ON card.account = `account owner`.id\_owner

JOIN currency ON card.currency = currency.currency\_id;

Просмотреть это представление можно с помощью просто запроса:

SELECT \* FROM GET\_CARD\_INFO;

## **4.2 Процедуры и функции**

Хранимая процедура – подпрограмма (возможно, параметризованная), предназначенная для выполнения ряда операций с данными и структурами базы данных, хранимая на стороне базы данных и доступная как для вызова из кода других процедур и триггеров, так и для непосредственного исполнения.

Хранимая функция, пользовательская функция – подпрограмма (возможно, параметризованная), расширяющая возможности языка SQL и работающая аналогично встроенным в СУБД функциям; обязана возвращать значения.

В обоих случаях достигается следующий ряд преимуществ:

* Необходимые действия описаны один раз, т.е. не нужно писать один и тот же код многократно в разных местах;
* Вызов подпрограммы – это короткая синтаксическая конструкция, что намного упрощает читаемость кода (в сравнении с ситуацией, когда все соответствующие действия были бы описаны явно);
* Подпрограмма работает на стороне СУБД, т.е. никакие данные не передаются в приложение или куда бы то ни было ещё, что повышает производительность и безопасность.

Для удаления всех предложений используется следующая процедура:

DELIMITER $$

CREATE procedure clean\_promotions()

BEGIN

delete from promotion

where DATEDIFF(end\_date, curdate()) < 0;

END$$

DELIMITER ;

**4.3 Триггеры**

Триггер – специальный объект базы данных, описывающий перечень действий, которые необходимо автоматически выполнить при наступлении указанного события.

Наиболее классическим вариантом использования триггеров является обеспечение реакции на модификацию данных, т.е. на операции вставки, обновления и удаления. Очевидно, что при наступлении некоторого события, СУБД может выполнить триггер до, после или вместо соответствующей операции.

С помощью триггеров в общем случае решается следующий спектр задач:

* Организация каскадных операций (если используемый метод доступа таковые не поддерживает) или реализация более сложной логики каскадных операций, чем предоставляет СУБД;
* Обновление данных кэширующих (или агрегирующих) полей и таблиц;
* Контроль и изменение таких значений полей, которые находятся в строгой зависимости от значений других полей (или иных условий);
* Контроль формата и значений данных в случае, если СУБД не поддерживает проверки, или необходимо сформировать информативные сообщения об ошибках, или логика проверки выходит за рамки того, что предоставляют стандартные средства СУБД;
* Прозрачное исправление ошибок в данных.

Перед вставкой информации в курс обмена мы проверяем актуальность вставляемых данных при помощи следующего триггера:

DELIMITER |

create trigger exchange\_rate\_BEFORE\_INSERT

BEFORE insert on exchange\_rate

for each row begin

if NEW.rate < 0 then

set New.id = null;

END IF;

end|

DELIMITER ;

**4.4 Ограничения**

Ограничения UNIQUE можно использовать для обеспечения того, чтобы в указанные столбцы, не входящие в состав первичного ключа, не вводились повторяющиеся значения. Хотя уникальность значений ограничения UNIQUE и PRIMARY KEY гарантируют в равной степени, в случае, когда необходимо обеспечить уникальность в столбце или комбинации столбцов, которые не не являются первичными ключевыми, вместо ограничения PRIMARY KEY следует использовать ограничение UNIQUE.

В отличие от PRIMARY KEY, ограничения UNIQUE допускают значение NULL. Однако, как и всякое другое значение столбца с ограничением UNIQUE, NULL может встречаться только один раз. На ограничение UNIQUE могут ссылаться ограничения FOREIGN KEY.NULL может встречаться только один раз. На ограничение UNIQUE могут ссылаться ограничения FOREIGN KEY.

# ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ

Описание выполненных тестов и их результатов над базой данных представлено в таблице 5.1.

Таблица 5.1 – Проиведенные тесты

| **Номер теста** | **Описание теста** | **Результат** |
| --- | --- | --- |
| 1 | Добавление записи в таблицу. Ожидаемый результат: запись добавлена. | Тест пройден |
| 2 | Изменение данных в записи. Ожидаемый результат: При вставке в таблицу `promotion` значение поля `name` изменено для записи с id = 1 | Тест пройден |
| 3 | Удаление записей из таблицы. Ожидаемый результат: при удалении записи из `promotion` мы ожидаем что будет удалена запись с id = 1. | Тест пройден |
| 4 | Осуществление выборки данных из таблицы. Ожидаемый результат: отображён список записей в выбранной таблице. | Тест пройден |
| 5 | Добавление записи с неверным значением. Ожидаемый результат: запись будет отклонено из-за ограничений. | Тест пройден |
| 6 | Использование хранимой процедуры clean\_promotions(). Ожидаемый результат: удалены истёкшие акции. | Тест пройден |
| 7 | Каскадное удаление статусов после удаления родительской таблицы. Ожидаемый результат: все таблицы у которых стоит каскадное удаление были удалены. | Тест пройден |

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта была проанализирована предметная область, рассмотрены существующие аналоги и выявлены их преимущества и недостатки.

При разработке базы данных использовалась система управления реляционными базами данных MySQL и среда визуального проектирования и моделирования Sparx Enterprise Architect.

На этапе инфологического моделирования были определены основные сущности и связи между ними. На логическом уровне таблицы и общая архитектура базы данных совершенствовалась в соответствии с выдвинутыми функциональными требованиями.

В соответствии с поставленной целью была разработана база данных мобильного банкинга, предоставляющая функционал по созданию нового пользователя, совершения платежей и транзакций, перевод средств с карты на карту, инвестиции, покупка криптовалют и ценных бумаг.

Проведено тестирование базы данных, поставленная цель была выполнена в полном объеме.

В ходе разработки были усовершенствованы знания о СУБД MySQL, о языке запросов SQL, о моделировании, проектировании, нормализации и оптимизации баз данных.

В соответствии с полученным результатом работы можно сделать вывод, что разработанная база данных мобильного банкинга работает корректно и все требования технического задания выполнены в полном объеме.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] MySQL – система управления базами данных [Электронный ресурс] – Режим доступа: <https://web-creator.ru/articles/mysql> – Дата доступа: 15.09.2022

[2] Дентко – современная стоматология в Минске[Электронный ресурс]. – Режим доступа: <https://dentko.by/> – Дата доступа: 18.09.2022

[3] KANO – сеть стоматологических клиник [Электронный ресурс] – Режим доступа: <https://kano.by/> – Дата доступа: 18.09.2022

[4] Мобильное приложение Aibolit [Электронный ресурс] – Режим доступа: https://aibolit.md/mobile – Дата доступа: 18.09.2022

[5] MySQL – Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/MySQL> – Дата доступа: 27.09.2022

[6] Enterprise Architect (software) [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Enterprise\_Architect\_(software) – Дата доступа: 27.09.2022

[7] Администратор баз данных [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Администратор\_баз\_данных – Дата доступа: 01.10.2022

[8] Логист, обязанности и требования [Электронный ресурс] – Режим доступа: <https://enjoy-job.ru/professions/logist/> – Дата доступа: 03.10.2022

[9] Нормализация отношений. Первая и вторая нормальные формы [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/129195/> – Дата доступа: 19.10.2022

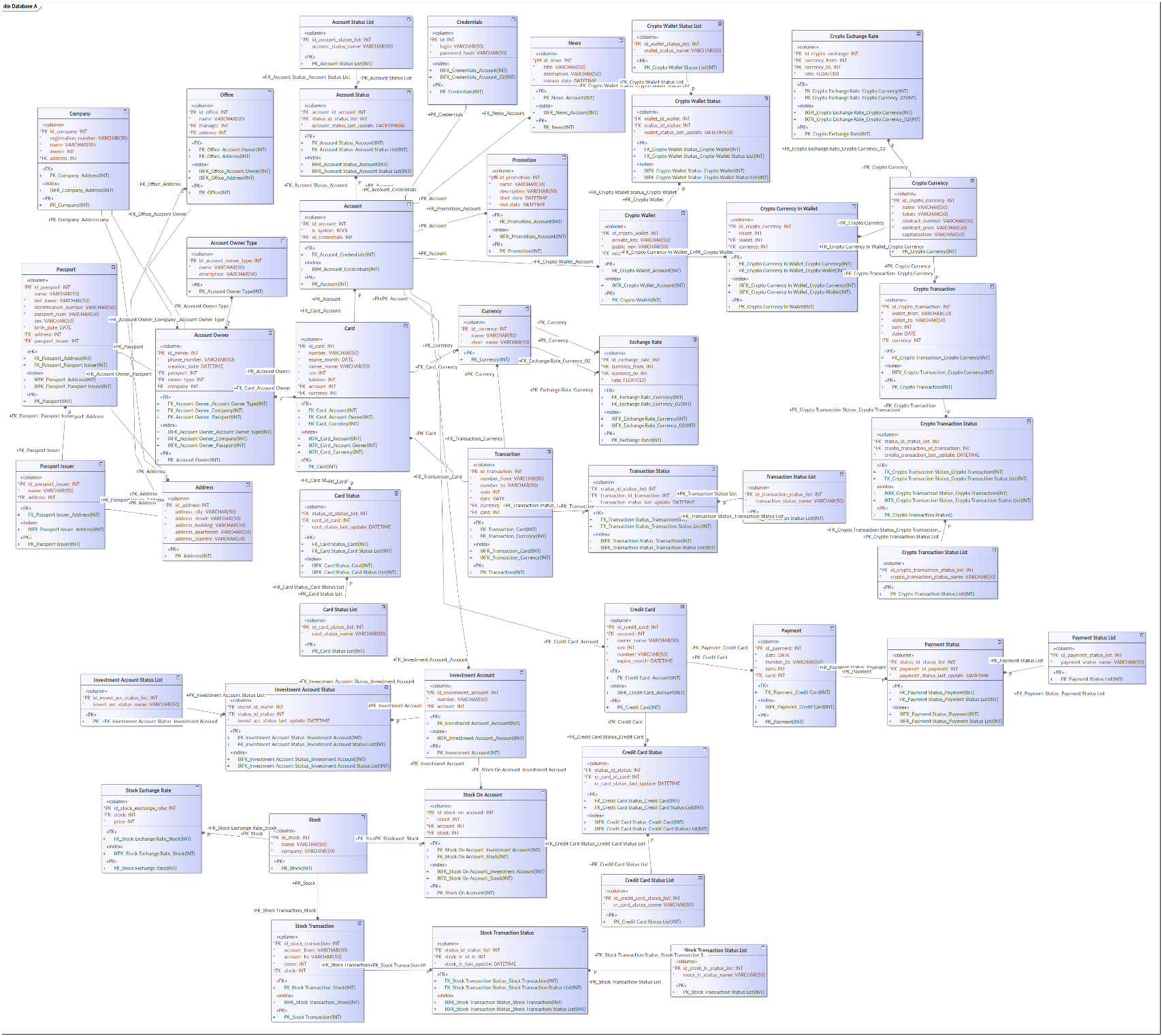
[10] [Нормализация](https://superuser.com/questions/1410289/what-are-commited-memory-cached-paged-not-paged-pool-how-they-are-d) [Электронный ресурс] – Режим доступа: https://metanit.com/sql/tutorial/2.1.php – Дата доступа: 29.10.2021

[11] Представления (VIEW) в MySQL [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/47031/> – Дата доступа: 10.11.2021

[12] Куликов С.С. Реляционные базы данных в примерах: 2021 - 103 с.

# ПРИЛОЖЕНИЕ А

**Логическая схема базы данных**

****

# ПРИЛОЖЕНИЕ Б

**Дамп структуры базы данных**

SET FOREIGN\_KEY\_CHECKS=0

;

/\* Drop Tables \*/

DROP TABLE IF EXISTS `Account` CASCADE

;

DROP TABLE IF EXISTS `Account Owner` CASCADE

;

DROP TABLE IF EXISTS `Account Owner Type` CASCADE

;

DROP TABLE IF EXISTS `Account Status` CASCADE

;

DROP TABLE IF EXISTS `Account Status List` CASCADE

;

DROP TABLE IF EXISTS `Address` CASCADE

;

DROP TABLE IF EXISTS `Card` CASCADE

;

DROP TABLE IF EXISTS `Card Status` CASCADE

;

DROP TABLE IF EXISTS `Card Status List` CASCADE

;

DROP TABLE IF EXISTS `Company` CASCADE

;

DROP TABLE IF EXISTS `Credentials` CASCADE

;

DROP TABLE IF EXISTS `Credit Card` CASCADE

;

DROP TABLE IF EXISTS `Credit Card Status` CASCADE

;

DROP TABLE IF EXISTS `Credit Card Status List` CASCADE

;

DROP TABLE IF EXISTS `Crypto Currency` CASCADE

;

DROP TABLE IF EXISTS `Crypto Currency In Wallet` CASCADE

;

DROP TABLE IF EXISTS `Crypto Exchange Rate` CASCADE

;

DROP TABLE IF EXISTS `Crypto Transaction` CASCADE

;

DROP TABLE IF EXISTS `Crypto Transaction Status` CASCADE

;

DROP TABLE IF EXISTS `Crypto Transaction Status List` CASCADE

;

DROP TABLE IF EXISTS `Crypto Wallet` CASCADE

;

DROP TABLE IF EXISTS `Crypto Wallet Status` CASCADE

;

DROP TABLE IF EXISTS `Crypto Wallet Status List` CASCADE

;

DROP TABLE IF EXISTS `Currency` CASCADE

;

DROP TABLE IF EXISTS `Exchange Rate` CASCADE

;

DROP TABLE IF EXISTS `Investment Account` CASCADE

;

DROP TABLE IF EXISTS `Investment Account Status` CASCADE

;

DROP TABLE IF EXISTS `Investment Account Status List` CASCADE

;

DROP TABLE IF EXISTS `News` CASCADE

;

DROP TABLE IF EXISTS `Office` CASCADE

;

DROP TABLE IF EXISTS `Passport` CASCADE

;

DROP TABLE IF EXISTS `Passport Issuer` CASCADE

;

DROP TABLE IF EXISTS `Payment` CASCADE

;

DROP TABLE IF EXISTS `Payment Status` CASCADE

;

DROP TABLE IF EXISTS `Payment Status List` CASCADE

;

DROP TABLE IF EXISTS `Promotion` CASCADE

;

DROP TABLE IF EXISTS `Stock` CASCADE

;

DROP TABLE IF EXISTS `Stock Exchange Rate` CASCADE

;

DROP TABLE IF EXISTS `Stock On Account` CASCADE

;

DROP TABLE IF EXISTS `Stock Transaction` CASCADE

;

DROP TABLE IF EXISTS `Stock Transaction Status` CASCADE

;

DROP TABLE IF EXISTS `Stock Transaction Status List` CASCADE

;

DROP TABLE IF EXISTS `Transaction` CASCADE

;

DROP TABLE IF EXISTS `Transaction Status` CASCADE

;

DROP TABLE IF EXISTS `Transaction Status List` CASCADE

;

/\* Create Tables \*/

CREATE TABLE `Account`

(

`id\_account` INT NOT NULL,

`is\_system` BOOL NOT NULL,

`id\_credentials` INT NOT NULL,

CONSTRAINT `PK\_Account` PRIMARY KEY (`id\_account` ASC)

)

;

CREATE TABLE `Account Owner`

(

`id\_owner` INT NOT NULL,

`phone\_number` VARCHAR(50) NOT NULL,

`creation\_date` DATETIME NOT NULL,

`passport` INT NOT NULL,

`owner\_type` INT NOT NULL,

`company` INT NULL,

CONSTRAINT `PK\_Account Owner` PRIMARY KEY (`id\_owner` ASC)

)

;

CREATE TABLE `Account Owner Type`

(

`id\_account\_owner\_type` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`description` VARCHAR(50) NULL,

CONSTRAINT `PK\_Account Owner Type` PRIMARY KEY (`id\_account\_owner\_type` ASC)

)

;

CREATE TABLE `Account Status`

(

`account\_id\_account` INT NOT NULL,

`status\_id\_status\_list` INT NOT NULL,

`account\_status\_last\_update` DATETIME(4) NOT NULL

)

;

CREATE TABLE `Account Status List`

(

`id\_account\_status\_list` INT NOT NULL,

`account\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Account Status List` PRIMARY KEY (`id\_account\_status\_list` ASC)

)

;

CREATE TABLE `Address`

(

`id\_address` INT NOT NULL,

`address\_city` VARCHAR(50) NOT NULL,

`address\_street` VARCHAR(50) NOT NULL,

`address\_building` VARCHAR(50) NOT NULL,

`address\_apartment` VARCHAR(50) NULL,

`address\_country` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Address` PRIMARY KEY (`id\_address` ASC)

)

;

CREATE TABLE `Card`

(

`id\_card` INT NOT NULL,

`number` VARCHAR(50) NOT NULL,

`expire\_month` DATE NOT NULL,

`owner\_name` VARCHAR(50) NOT NULL,

`cvv` INT NOT NULL,

`balance` INT NOT NULL,

`account` INT NOT NULL,

`currency` INT NOT NULL,

CONSTRAINT `PK\_Card` PRIMARY KEY (`id\_card` ASC)

)

;

CREATE TABLE `Card Status`

(

`status\_id\_status\_list` INT NOT NULL,

`card\_id\_card` INT NOT NULL,

`card\_status\_last\_update` DATETIME NOT NULL

)

;

CREATE TABLE `Card Status List`

(

`id\_card\_status\_list` INT NOT NULL,

`card\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Card Status List` PRIMARY KEY (`id\_card\_status\_list` ASC)

)

;

CREATE TABLE `Company`

(

`id\_company` INT NOT NULL,

`registration\_number` VARCHAR(50) NOT NULL,

`name` VARCHAR(50) NOT NULL,

`owner` INT NOT NULL,

`address` INT NOT NULL,

CONSTRAINT `PK\_Company` PRIMARY KEY (`id\_company` ASC)

)

;

CREATE TABLE `Credentials`

(

`id` INT NOT NULL,

`login` VARCHAR(50) NOT NULL,

`password\_hash` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Credentials` PRIMARY KEY (`id` ASC)

)

;

CREATE TABLE `Credit Card`

(

`id\_credit\_card` INT NOT NULL,

`account` INT NOT NULL,

`owner\_name` VARCHAR(50) NOT NULL,

`cvv` INT NOT NULL,

`number` VARCHAR(50) NOT NULL,

`expire\_month` DATETIME NOT NULL,

CONSTRAINT `PK\_Credit Card` PRIMARY KEY (`id\_credit\_card` ASC)

)

;

CREATE TABLE `Credit Card Status`

(

`status\_id\_status` INT NOT NULL,

`cr\_card\_id\_card` INT NOT NULL,

`cr\_card\_status\_last\_update` DATETIME NOT NULL

)

;

CREATE TABLE `Credit Card Status List`

(

`id\_credit\_card\_status\_list` INT NOT NULL,

`cr\_card\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Credit Card Status List` PRIMARY KEY (`id\_credit\_card\_status\_list` ASC)

)

;

CREATE TABLE `Crypto Currency`

(

`id\_crypto\_currenсy` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`token` VARCHAR(50) NOT NULL,

`contract\_number` VARCHAR(50) NOT NULL,

`contract\_price` VARCHAR(50) NOT NULL,

`capitalization` VARCHAR(50) NULL,

CONSTRAINT `PK\_Crypto Currency` PRIMARY KEY (`id\_crypto\_currenсy` ASC)

)

;

CREATE TABLE `Crypto Currency In Wallet`

(

`id\_crypto\_currency` INT NOT NULL,

`count` INT NOT NULL,

`wallet` INT NOT NULL,

`currency` INT NOT NULL,

CONSTRAINT `PK\_Crypto Currency In Wallet` PRIMARY KEY (`id\_crypto\_currency` ASC)

)

;

CREATE TABLE `Crypto Exchange Rate`

(

`id\_crypto\_exchange` INT NOT NULL,

`currency\_from` INT NOT NULL,

`currency\_to` INT NOT NULL,

`rate` FLOAT(10,0) NOT NULL,

CONSTRAINT `PK\_Crypto Exchange Rate` PRIMARY KEY (`id\_crypto\_exchange` ASC)

)

;

CREATE TABLE `Crypto Transaction`

(

`id\_crypto\_transaction` INT NOT NULL,

`wallet\_from` VARCHAR(50) NOT NULL,

`wallet\_to` VARCHAR(50) NOT NULL,

`sum` INT NOT NULL,

`date` DATE NOT NULL,

`currency` INT NOT NULL,

CONSTRAINT `PK\_Crypto Transaction` PRIMARY KEY (`id\_crypto\_transaction` ASC)

)

;

CREATE TABLE `Crypto Transaction Status`

(

`status\_id\_status\_list` INT NOT NULL,

`crypto\_transaction\_id\_transaction` INT NOT NULL,

`crypto\_transaction\_last\_update` DATETIME NOT NULL,

CONSTRAINT `PK\_Crypto Transaction Status` PRIMARY KEY ()

)

;

CREATE TABLE `Crypto Transaction Status List`

(

`id\_crypto\_transaction\_status\_list` INT NOT NULL,

`crypto\_transaction\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Crypto Transaction Status List` PRIMARY KEY (`id\_crypto\_transaction\_status\_list` ASC)

)

;

CREATE TABLE `Crypto Wallet`

(

`id\_crypto\_wallet` INT NOT NULL,

`private\_key` VARCHAR(50) NOT NULL,

`public\_key` VARCHAR(50) NOT NULL,

`account` INT NOT NULL,

CONSTRAINT `PK\_Crypto Wallet` PRIMARY KEY (`id\_crypto\_wallet` ASC)

)

;

CREATE TABLE `Crypto Wallet Status`

(

`wallet\_id\_wallet` INT NOT NULL,

`status\_id\_status` INT NOT NULL,

`wallet\_status\_last\_update` DATETIME(4) NOT NULL

)

;

CREATE TABLE `Crypto Wallet Status List`

(

`id\_wallet\_status\_list` INT NOT NULL,

`wallet\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Crypto Wallet Status List` PRIMARY KEY (`id\_wallet\_status\_list` ASC)

)

;

CREATE TABLE `Currency`

(

`id\_currency` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`short\_name` VARCHAR(10) NOT NULL,

CONSTRAINT `PK\_Currency` PRIMARY KEY (`id\_currency` ASC)

)

;

CREATE TABLE `Exchange Rate`

(

`id\_exchange\_rate` INT NOT NULL,

`currency\_from` INT NOT NULL,

`currency\_to` INT NOT NULL,

`rate` FLOAT(10,0) NOT NULL,

CONSTRAINT `PK\_Exchange Rate` PRIMARY KEY (`id\_exchange\_rate` ASC)

)

;

CREATE TABLE `Investment Account`

(

`id\_investment\_account` INT NOT NULL,

`number` VARCHAR(50) NOT NULL,

`account` INT NOT NULL,

CONSTRAINT `PK\_Investment Account` PRIMARY KEY (`id\_investment\_account` ASC)

)

;

CREATE TABLE `Investment Account Status`

(

`invest\_id\_invest` INT NOT NULL,

`status\_id\_status` INT NOT NULL,

`invest\_acc\_status\_last\_update` DATETIME NOT NULL

)

;

CREATE TABLE `Investment Account Status List`

(

`id\_invest\_acc\_status\_list` INT NOT NULL,

`invest\_acc\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Investment Account Status List` PRIMARY KEY (`id\_invest\_acc\_status\_list` ASC)

)

;

CREATE TABLE `News`

(

`id\_news` INT NOT NULL,

`title` VARCHAR(50) NOT NULL,

`description` VARCHAR(50) NULL,

`release\_date` DATETIME NOT NULL,

CONSTRAINT `PK\_News` PRIMARY KEY (`id\_news` ASC)

)

;

CREATE TABLE `Office`

(

`id\_office` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`manager` INT NOT NULL,

`address` INT NOT NULL,

CONSTRAINT `PK\_Office` PRIMARY KEY (`id\_office` ASC)

)

;

CREATE TABLE `Passport`

(

`id\_passport` INT NOT NULL,

`name` VARCHAR(50) NULL,

`last\_name` VARCHAR(50) NOT NULL,

`identification\_number` VARCHAR(50) NOT NULL,

`passport\_num` VARCHAR(50) NOT NULL,

`sex` VARCHAR(50) NULL,

`birth\_date` DATE NOT NULL,

`address` INT NOT NULL,

`passport\_issuer` INT NOT NULL,

CONSTRAINT `PK\_Passport` PRIMARY KEY (`id\_passport` ASC)

)

;

CREATE TABLE `Passport Issuer`

(

`id\_passport\_issuer` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`address` INT NOT NULL,

CONSTRAINT `PK\_Passport Issuer` PRIMARY KEY (`id\_passport\_issuer` ASC)

)

;

CREATE TABLE `Payment`

(

`id\_payment` INT NOT NULL,

`date` DATE NOT NULL,

`number\_to` VARCHAR(50) NOT NULL,

`sum` INT NOT NULL,

`card` INT NOT NULL,

CONSTRAINT `PK\_Payment` PRIMARY KEY (`id\_payment` ASC)

)

;

CREATE TABLE `Payment Status`

(

`status\_id\_status\_list` INT NOT NULL,

`payment\_id\_payment` INT NOT NULL,

`payment\_status\_last\_update` DATETIME NOT NULL

)

;

CREATE TABLE `Payment Status List`

(

`id\_payment\_status\_list` INT NOT NULL,

`payment\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Payment Status List` PRIMARY KEY (`id\_payment\_status\_list` ASC)

)

;

CREATE TABLE `Promotion`

(

`id\_promotion` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`description` VARCHAR(50) NULL,

`start\_date` DATETIME NOT NULL,

`end\_date` DATETIME NOT NULL,

CONSTRAINT `PK\_Promotion` PRIMARY KEY (`id\_promotion` ASC)

)

;

CREATE TABLE `Stock`

(

`id\_stock` INT NOT NULL,

`name` VARCHAR(50) NOT NULL,

`company` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Stock` PRIMARY KEY (`id\_stock` ASC)

)

;

CREATE TABLE `Stock Exchange Rate`

(

`id\_stock\_exchange\_rate` INT NOT NULL,

`stock` INT NOT NULL,

`price` INT NOT NULL,

CONSTRAINT `PK\_Stock Exchange Rate` PRIMARY KEY (`id\_stock\_exchange\_rate` ASC)

)

;

CREATE TABLE `Stock On Account`

(

`id\_stock\_on\_account` INT NOT NULL,

`count` INT NOT NULL,

`account` INT NOT NULL,

`stock` INT NOT NULL,

CONSTRAINT `PK\_Stock On Account` PRIMARY KEY (`id\_stock\_on\_account` ASC)

)

;

CREATE TABLE `Stock Transaction`

(

`id\_stock\_transaction` INT NOT NULL,

`account\_from` VARCHAR(50) NOT NULL,

`account\_to` VARCHAR(50) NOT NULL,

`count` INT NOT NULL,

`stock` INT NOT NULL,

CONSTRAINT `PK\_Stock Transaction` PRIMARY KEY (`id\_stock\_transaction` ASC)

)

;

CREATE TABLE `Stock Transaction Status`

(

`status\_id\_status\_list` INT NOT NULL,

`stock\_tr\_id\_tr` INT NOT NULL,

`stock\_tr\_last\_update` DATETIME NOT NULL

)

;

CREATE TABLE `Stock Transaction Status List`

(

`id\_stock\_tr\_status\_list` INT NOT NULL,

`stock\_tr\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Stock Transaction Status List` PRIMARY KEY (`id\_stock\_tr\_status\_list` ASC)

)

;

CREATE TABLE `Transaction`

(

`id\_transaction` INT NOT NULL,

`number\_from` VARCHAR(50) NOT NULL,

`number\_to` VARCHAR(50) NOT NULL,

`sum` INT NOT NULL,

`date` DATE NOT NULL,

`currency` INT NOT NULL,

`card` INT NOT NULL,

CONSTRAINT `PK\_Transaction` PRIMARY KEY (`id\_transaction` ASC)

)

;

CREATE TABLE `Transaction Status`

(

`status\_id\_status\_list` INT NOT NULL,

`transaction\_id\_transaction` INT NOT NULL,

`transaction\_status\_last\_update` DATETIME NOT NULL

)

;

CREATE TABLE `Transaction Status List`

(

`id\_transaction\_status\_list` INT NOT NULL,

`transaction\_status\_name` VARCHAR(50) NOT NULL,

CONSTRAINT `PK\_Transaction Status List` PRIMARY KEY (`id\_transaction\_status\_list` ASC)

)

;

/\* Create Primary Keys, Indexes, Uniques, Checks \*/

ALTER TABLE `Account`

ADD INDEX `IXFK\_Account\_Credentials` (`id\_credentials` ASC)

;

ALTER TABLE `Account Owner`

ADD INDEX `IXFK\_Account Owner\_Account Owner Type` (`owner\_type` ASC)

;

ALTER TABLE `Account Owner`

ADD INDEX `IXFK\_Account Owner\_Company` (`company` ASC)

;

ALTER TABLE `Account Owner`

ADD INDEX `IXFK\_Account Owner\_Passport` (`passport` ASC)

;

ALTER TABLE `Account Status`

ADD INDEX `IXFK\_Account Status\_Account` (`account\_id\_account` ASC)

;

ALTER TABLE `Account Status`

ADD INDEX `IXFK\_Account Status\_Account Status List` (`status\_id\_status\_list` ASC)

;

ALTER TABLE `Card`

ADD INDEX `IXFK\_Card\_Account` (`account` ASC)

;

ALTER TABLE `Card`

ADD INDEX `IXFK\_Card\_Account Owner` (`account` ASC)

;

ALTER TABLE `Card`

ADD INDEX `IXFK\_Card\_Currency` (`currency` ASC)

;

ALTER TABLE `Card Status`

ADD INDEX `IXFK\_Card Status\_Card` (`card\_id\_card` ASC)

;

ALTER TABLE `Card Status`

ADD INDEX `IXFK\_Card Status\_Card Status List` (`status\_id\_status\_list` ASC)

;

ALTER TABLE `Company`

ADD INDEX `IXFK\_Company\_Address` (`address` ASC)

;

ALTER TABLE `Credentials`

ADD INDEX `IXFK\_Credentials\_Account` (`id` ASC)

;

ALTER TABLE `Credentials`

ADD INDEX `IXFK\_Credentials\_Account\_02` (`id` ASC)

;

ALTER TABLE `Credit Card`

ADD INDEX `IXFK\_Credit Card\_Account` (`account` ASC)

;

ALTER TABLE `Credit Card Status`

ADD INDEX `IXFK\_Credit Card Status\_Credit Card` (`cr\_card\_id\_card` ASC)

;

ALTER TABLE `Credit Card Status`

ADD INDEX `IXFK\_Credit Card Status\_Credit Card Status List` (`status\_id\_status` ASC)

;

ALTER TABLE `Crypto Currency In Wallet`

ADD INDEX `IXFK\_Crypto Currency In Wallet\_Crypto Currency` (`currency` ASC)

;

ALTER TABLE `Crypto Currency In Wallet`

ADD INDEX `IXFK\_Crypto Currency In Wallet\_Crypto Wallet` (`wallet` ASC)

;

ALTER TABLE `Crypto Exchange Rate`

ADD INDEX `IXFK\_Crypto Exchange Rate\_Crypto Currency` (`currency\_to` ASC)

;

ALTER TABLE `Crypto Exchange Rate`

ADD INDEX `IXFK\_Crypto Exchange Rate\_Crypto Currency\_02` (`currency\_from` ASC)

;

ALTER TABLE `Crypto Transaction`

ADD INDEX `IXFK\_Crypto Transaction\_Crypto Currency` (`currency` ASC)

;

ALTER TABLE `Crypto Transaction Status`

ADD INDEX `IXFK\_Crypto Transaction Status\_Crypto Transaction` (`crypto\_transaction\_id\_transaction` ASC)

;

ALTER TABLE `Crypto Transaction Status`

ADD INDEX `IXFK\_Crypto Transaction Status\_Crypto Transaction Status List` (`status\_id\_status\_list` ASC)

;

ALTER TABLE `Crypto Wallet`

ADD INDEX `IXFK\_Crypto Wallet\_Account` (`account` ASC)

;

ALTER TABLE `Crypto Wallet Status`

ADD INDEX `IXFK\_Crypto Wallet Status\_Crypto Wallet` (`wallet\_id\_wallet` ASC)

;

ALTER TABLE `Crypto Wallet Status`

ADD INDEX `IXFK\_Crypto Wallet Status\_Crypto Wallet Status List` (`status\_id\_status` ASC)

;

ALTER TABLE `Exchange Rate`

ADD INDEX `IXFK\_Exchange Rate\_Currency` (`currency\_from` ASC)

;

ALTER TABLE `Exchange Rate`

ADD INDEX `IXFK\_Exchange Rate\_Currency\_02` (`currency\_to` ASC)

;

ALTER TABLE `Investment Account`

ADD INDEX `IXFK\_Investment Account\_Account` (`account` ASC)

;

ALTER TABLE `Investment Account Status`

ADD INDEX `IXFK\_Investment Account Status\_Investment Account` (`invest\_id\_invest` ASC)

;

ALTER TABLE `Investment Account Status`

ADD INDEX `IXFK\_Investment Account Status\_Investment Account Status List` (`status\_id\_status` ASC)

;

ALTER TABLE `News`

ADD INDEX `IXFK\_News\_Account` (`id\_news` ASC)

;

ALTER TABLE `Office`

ADD INDEX `IXFK\_Office\_Account Owner` (`manager` ASC)

;

ALTER TABLE `Office`

ADD INDEX `IXFK\_Office\_Address` (`address` ASC)

;

ALTER TABLE `Passport`

ADD INDEX `IXFK\_Passport\_Address` (`address` ASC)

;

ALTER TABLE `Passport`

ADD INDEX `IXFK\_Passport\_Passport Issuer` (`passport\_issuer` ASC)

;

ALTER TABLE `Passport Issuer`

ADD INDEX `IXFK\_Passport Issuer\_Address` (`address` ASC)

;

ALTER TABLE `Payment`

ADD INDEX `IXFK\_Payment\_Credit Card` (`card` ASC)

;

ALTER TABLE `Payment Status`

ADD INDEX `IXFK\_Payment Status\_Payment` (`payment\_id\_payment` ASC)

;

ALTER TABLE `Payment Status`

ADD INDEX `IXFK\_Payment Status\_Payment Status List` (`status\_id\_status\_list` ASC)

;

ALTER TABLE `Promotion`

ADD INDEX `IXFK\_Promotion\_Account` (`id\_promotion` ASC)

;

ALTER TABLE `Stock Exchange Rate`

ADD INDEX `IXFK\_Stock Exchange Rate\_Stock` (`stock` ASC)

;

ALTER TABLE `Stock On Account`

ADD INDEX `IXFK\_Stock On Account\_Investment Account` (`account` ASC)

;

ALTER TABLE `Stock On Account`

ADD INDEX `IXFK\_Stock On Account\_Stock` (`stock` ASC)

;

ALTER TABLE `Stock Transaction`

ADD INDEX `IXFK\_Stock Transaction\_Stock` (`stock` ASC)

;

ALTER TABLE `Stock Transaction Status`

ADD INDEX `IXFK\_Stock Transaction Status\_Stock Transaction` (`stock\_tr\_id\_tr` ASC)

;

ALTER TABLE `Stock Transaction Status`

ADD INDEX `IXFK\_Stock Transaction Status\_Stock Transaction Status List` (`status\_id\_status\_list` ASC)

;

ALTER TABLE `Transaction`

ADD INDEX `IXFK\_Transaction\_Card` (`card` ASC)

;

ALTER TABLE `Transaction`

ADD INDEX `IXFK\_Transaction\_Currency` (`currency` ASC)

;

ALTER TABLE `Transaction Status`

ADD INDEX `IXFK\_Transaction Status\_Transaction` (`transaction\_id\_transaction` ASC)

;

ALTER TABLE `Transaction Status`

ADD INDEX `IXFK\_Transaction Status\_Transaction Status List` (`status\_id\_status\_list` ASC)

;

/\* Create Foreign Key Constraints \*/

ALTER TABLE `Account`

ADD CONSTRAINT `FK\_Account\_Credentials`

FOREIGN KEY (`id\_credentials`) REFERENCES `Credentials` (`id`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Account Owner`

ADD CONSTRAINT `FK\_Account Owner\_Account Owner Type`

FOREIGN KEY (`owner\_type`) REFERENCES `Account Owner Type` (`id\_account\_owner\_type`) ON DELETE Restrict ON UPDATE No Action

;

ALTER TABLE `Account Owner`

ADD CONSTRAINT `FK\_Account Owner\_Company`

FOREIGN KEY (`company`) REFERENCES `Company` (`id\_company`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Account Owner`

ADD CONSTRAINT `FK\_Account Owner\_Passport`

FOREIGN KEY (`passport`) REFERENCES `Passport` (`id\_passport`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Account Status`

ADD CONSTRAINT `FK\_Account Status\_Account`

FOREIGN KEY (`account\_id\_account`) REFERENCES `Account` (`id\_account`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Account Status`

ADD CONSTRAINT `FK\_Account Status\_Account Status List`

FOREIGN KEY (`status\_id\_status\_list`) REFERENCES `Account Status List` (`id\_account\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Card`

ADD CONSTRAINT `FK\_Card\_Account`

FOREIGN KEY (`account`) REFERENCES `Account` (`id\_account`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Card`

ADD CONSTRAINT `FK\_Card\_Account Owner`

FOREIGN KEY (`account`) REFERENCES `Account Owner` (`id\_owner`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Card`

ADD CONSTRAINT `FK\_Card\_Currency`

FOREIGN KEY (`currency`) REFERENCES `Currency` (`id\_currency`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Card Status`

ADD CONSTRAINT `FK\_Card Status\_Card`

FOREIGN KEY (`card\_id\_card`) REFERENCES `Card` (`id\_card`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Card Status`

ADD CONSTRAINT `FK\_Card Status\_Card Status List`

FOREIGN KEY (`status\_id\_status\_list`) REFERENCES `Card Status List` (`id\_card\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Company`

ADD CONSTRAINT `FK\_Company\_Address`

FOREIGN KEY (`address`) REFERENCES `Address` (`id\_address`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Credit Card`

ADD CONSTRAINT `FK\_Credit Card\_Account`

FOREIGN KEY (`account`) REFERENCES `Account` (`id\_account`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Credit Card Status`

ADD CONSTRAINT `FK\_Credit Card Status\_Credit Card`

FOREIGN KEY (`cr\_card\_id\_card`) REFERENCES `Credit Card` (`id\_credit\_card`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Credit Card Status`

ADD CONSTRAINT `FK\_Credit Card Status\_Credit Card Status List`

FOREIGN KEY (`status\_id\_status`) REFERENCES `Credit Card Status List` (`id\_credit\_card\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Crypto Currency In Wallet`

ADD CONSTRAINT `FK\_Crypto Currency In Wallet\_Crypto Currency`

FOREIGN KEY (`currency`) REFERENCES `Crypto Currency` (`id\_crypto\_currenсy`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Crypto Currency In Wallet`

ADD CONSTRAINT `FK\_Crypto Currency In Wallet\_Crypto Wallet`

FOREIGN KEY (`wallet`) REFERENCES `Crypto Wallet` (`id\_crypto\_wallet`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Crypto Exchange Rate`

ADD CONSTRAINT `FK\_Crypto Exchange Rate\_Crypto Currency`

FOREIGN KEY (`currency\_to`) REFERENCES `Crypto Currency` (`id\_crypto\_currenсy`) ON DELETE Restrict ON UPDATE Restrict

;

ALTER TABLE `Crypto Exchange Rate`

ADD CONSTRAINT `FK\_Crypto Exchange Rate\_Crypto Currency\_02`

FOREIGN KEY (`currency\_from`) REFERENCES `Crypto Currency` (`id\_crypto\_currenсy`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Crypto Transaction`

ADD CONSTRAINT `FK\_Crypto Transaction\_Crypto Currency`

FOREIGN KEY (`currency`) REFERENCES `Crypto Currency` (`id\_crypto\_currenсy`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Crypto Transaction Status`

ADD CONSTRAINT `FK\_Crypto Transaction Status\_Crypto Transaction`

FOREIGN KEY (`crypto\_transaction\_id\_transaction`) REFERENCES `Crypto Transaction` (`id\_crypto\_transaction`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Crypto Transaction Status`

ADD CONSTRAINT `FK\_Crypto Transaction Status\_Crypto Transaction Status List`

FOREIGN KEY (`status\_id\_status\_list`) REFERENCES `Crypto Transaction Status List` (`id\_crypto\_transaction\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Crypto Wallet`

ADD CONSTRAINT `FK\_Crypto Wallet\_Account`

FOREIGN KEY (`account`) REFERENCES `Account` (`id\_account`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Crypto Wallet Status`

ADD CONSTRAINT `FK\_Crypto Wallet Status\_Crypto Wallet`

FOREIGN KEY (`wallet\_id\_wallet`) REFERENCES `Crypto Wallet` (`id\_crypto\_wallet`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Crypto Wallet Status`

ADD CONSTRAINT `FK\_Crypto Wallet Status\_Crypto Wallet Status List`

FOREIGN KEY (`status\_id\_status`) REFERENCES `Crypto Wallet Status List` (`id\_wallet\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Exchange Rate`

ADD CONSTRAINT `FK\_Exchange Rate\_Currency`

FOREIGN KEY (`currency\_from`) REFERENCES `Currency` (`id\_currency`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Exchange Rate`

ADD CONSTRAINT `FK\_Exchange Rate\_Currency\_02`

FOREIGN KEY (`currency\_to`) REFERENCES `Currency` (`id\_currency`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Investment Account`

ADD CONSTRAINT `FK\_Investment Account\_Account`

FOREIGN KEY (`account`) REFERENCES `Account` (`id\_account`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Investment Account Status`

ADD CONSTRAINT `FK\_Investment Account Status\_Investment Account`

FOREIGN KEY (`invest\_id\_invest`) REFERENCES `Investment Account` (`id\_investment\_account`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Investment Account Status`

ADD CONSTRAINT `FK\_Investment Account Status\_Investment Account Status List`

FOREIGN KEY (`status\_id\_status`) REFERENCES `Investment Account Status List` (`id\_invest\_acc\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `News`

ADD CONSTRAINT `FK\_News\_Account`

FOREIGN KEY (`id\_news`) REFERENCES `Account` (`id\_account`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Office`

ADD CONSTRAINT `FK\_Office\_Account Owner`

FOREIGN KEY (`manager`) REFERENCES `Account Owner` (`id\_owner`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Office`

ADD CONSTRAINT `FK\_Office\_Address`

FOREIGN KEY (`address`) REFERENCES `Address` (`id\_address`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Passport`

ADD CONSTRAINT `FK\_Passport\_Address`

FOREIGN KEY (`address`) REFERENCES `Address` (`id\_address`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Passport`

ADD CONSTRAINT `FK\_Passport\_Passport Issuer`

FOREIGN KEY (`passport\_issuer`) REFERENCES `Passport Issuer` (`id\_passport\_issuer`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Passport Issuer`

ADD CONSTRAINT `FK\_Passport Issuer\_Address`

FOREIGN KEY (`address`) REFERENCES `Address` (`id\_address`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Payment`

ADD CONSTRAINT `FK\_Payment\_Credit Card`

FOREIGN KEY (`card`) REFERENCES `Credit Card` (`id\_credit\_card`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Payment Status`

ADD CONSTRAINT `FK\_Payment Status\_Payment`

FOREIGN KEY (`payment\_id\_payment`) REFERENCES `Payment` (`id\_payment`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Payment Status`

ADD CONSTRAINT `FK\_Payment Status\_Payment Status List`

FOREIGN KEY (`status\_id\_status\_list`) REFERENCES `Payment Status List` (`id\_payment\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Promotion`

ADD CONSTRAINT `FK\_Promotion\_Account`

FOREIGN KEY (`id\_promotion`) REFERENCES `Account` (`id\_account`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Stock Exchange Rate`

ADD CONSTRAINT `FK\_Stock Exchange Rate\_Stock`

FOREIGN KEY (`stock`) REFERENCES `Stock` (`id\_stock`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Stock On Account`

ADD CONSTRAINT `FK\_Stock On Account\_Investment Account`

FOREIGN KEY (`account`) REFERENCES `Investment Account` (`id\_investment\_account`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Stock On Account`

ADD CONSTRAINT `FK\_Stock On Account\_Stock`

FOREIGN KEY (`stock`) REFERENCES `Stock` (`id\_stock`) ON DELETE Set Null ON UPDATE Cascade

;

ALTER TABLE `Stock Transaction`

ADD CONSTRAINT `FK\_Stock Transaction\_Stock`

FOREIGN KEY (`stock`) REFERENCES `Stock` (`id\_stock`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Stock Transaction Status`

ADD CONSTRAINT `FK\_Stock Transaction Status\_Stock Transaction`

FOREIGN KEY (`stock\_tr\_id\_tr`) REFERENCES `Stock Transaction` (`id\_stock\_transaction`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Stock Transaction Status`

ADD CONSTRAINT `FK\_Stock Transaction Status\_Stock Transaction Status List`

FOREIGN KEY (`status\_id\_status\_list`) REFERENCES `Stock Transaction Status List` (`id\_stock\_tr\_status\_list`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Transaction`

ADD CONSTRAINT `FK\_Transaction\_Card`

FOREIGN KEY (`card`) REFERENCES `Card` (`id\_card`) ON DELETE Restrict ON UPDATE Restrict

;

ALTER TABLE `Transaction`

ADD CONSTRAINT `FK\_Transaction\_Currency`

FOREIGN KEY (`currency`) REFERENCES `Currency` (`id\_currency`) ON DELETE Restrict ON UPDATE Cascade

;

ALTER TABLE `Transaction Status`

ADD CONSTRAINT `FK\_Transaction Status\_Transaction`

FOREIGN KEY (`transaction\_id\_transaction`) REFERENCES `Transaction` (`id\_transaction`) ON DELETE Cascade ON UPDATE Cascade;

DELIMITER $$

CREATE FUNCTION assemble\_address(id int) RETURNS VARCHAR(150) DETERMINISTIC

BEGIN

RETURN (select concat(address.num, street.`name`, city.`name`, region.`name`, country.`name`)

from address

join street on address.street = street.id

join city on street.city = city.id

join region on city.region = region.id

join country on region.country = country.id

where address.id = id);

END$$

DELIMITER ;

DELIMITER |

create trigger exchange\_rate\_BEFORE\_INSERT

BEFORE insert on exchange\_rate

for each row begin

if NEW.rate < 0 then

set New.id = null;

END IF;

end|

DELIMITER ;

DELIMITER $$

CREATE procedure clean\_promotions()

BEGIN

delete from promotion

where DATEDIFF(end\_date, curdate()) < 0;

END$$

DELIMITER ;

CREATE VIEW GET\_CARD\_INFO AS

SELECT owner\_name AS Владелец, phone\_number AS НомерТелефона, cvv AS УникальныйНомер, short\_name As ИмяВалюты

FROM card

JOIN `account owner` ON card.account = `account owner`.id\_owner

JOIN currency ON card.currency = currency.currency\_id;