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)

ОТЧЕТ

по лабораторной работе №2

Вариант 1

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.Е.

Минск 2024

# Анализ модели базы данных «Банк»

1. Какие каскадные операции необходимы в этой базе данных?

Сущность Users:

* role\_id – запрет на каскадное удаление, каскадное обновление.

Сущность Accounts:

* user\_id – каскадное удаление, каскадное обновление.
* currency\_code – запрет на каскадное удаление, каскадное обновление.
* status\_id – запрет на каскадное удаление, каскадное обновление.

Сущность Owners:

* address\_id – запрет на каскадное удаление, каскадное обновление.
* owner\_id (внешний ключ на Users.user\_id) – каскадное удаление, каскадное обновление.

Сущность Cards:

* account\_id – каскадное удаление, каскадное обновление.
* status\_id – запрет на каскадное удаление, каскадное обновление.

Сущность Transactions:

* currency\_code – запрет на каскадное удаление, каскадное обновление.

Сущность TransactionStatusHistory:

* transaction\_status\_id – запрет на каскадное удаление, каскадное обновление.
* transaction\_id – каскадное удаление, каскадное обновление.

1. Существует ли возможность аномалий операций вставки, обновления, удаления данных?

Аномалий вставки, удаления и обновления не были обнаружены.

1. Можно ли использовать схемы «звезда» или «снежинка» с этой базой данных, чтобы избежать некоторых аномалий операций с данными?

Так как аномалий обнаружено не было, то в использовании схем «звезда» или «снежинка» нет необходимости.

1. Составьте список всех функциональных зависимостей в базе данных.

Сущность Roles:

* role\_id → role\_name

Сущность Users:

* user\_id → username, password\_hash, created\_at, role\_id

Сущность Currencies:

* currency\_code → currency\_name

Сущность AccountCardStatuses:

* status\_id → status\_name

Сущность Accounts:

* account\_id → created\_at, user\_id, balance, currency\_code, status\_id, iban

Сущность Address:

* address\_id → street, city, state, postal\_code, country

Сущность Owners:

* owner\_id → first\_name, last\_name, middle\_name, birth\_date, phone, email, address\_id, passport\_id

Сущность Cards:

* card\_id → card\_number, expiration\_date, account\_id, status\_id

Сущность Transactions:

* transaction\_id → transaction\_amount, transaction\_type, transaction\_date, currency\_code, source\_iban, destination\_iban

Сущность TransactionStatuses:

* transaction\_status\_id → status\_name

Сущность TransactionStatusHistory:

* status\_history\_id → transaction\_id, transaction\_status\_id, assigned\_at

1. Существуют ли отношения, имеющие многозначные зависимости?

В данной схеме многозначные зависимости отсутствуют.

1. Нарушает ли схема какие-либо «требования нормализации»?

Данная схема не нарушает никаких требований нормализации.

1. Существуют ли какие-либо потенциальные проблемы с производительностью базы данных?

Потенциальные индексы для улучшения производительности

1. Transactions: индексы на source\_iban и destination\_iban для ускорения выборок по IBAN.
2. TransactionStatusHistory: индекс на assigned\_at для быстрого поиска по временным диапазонам.
3. Users: индекс на username для ускорения аутентификации.
4. Owners: уникальные индексы на passport\_id и email для быстрого поиска и предотвращения дублирования.

Возможные проблемы с производительностью

1. Каскадные операции: могут замедлять работу из-за массовых обновлений в связанных таблицах.
2. Накопление данных в Transactions и TransactionStatusHistory: большое число записей со временем снижает скорость выборки. Индексы и архивирование помогут частично снизить нагрузку.
3. Для каждого отношения в базе данных определите, в какой нормальной форме оно находится. Запишите ответ.

Все отношения находятся в третьей нормальной форме, так как:

* каждый атрибут является атомарным,
* каждый атрибут полностью зависит от первичного ключа,
* транзитивных зависимостей отсутствуют.

1. Есть ли отношения с возможной, но ненужной дальнейшей нормализацией?
2. Users

* Поле username можно вынести в отдельную таблицу для отслеживания истории имен, но это избыточно и не требуется в данном случае.

1. Owners

* Поля phone и email можно нормализовать, вынеся их в отдельные таблицы для хранения нескольких номеров или адресов на одного владельца. Однако такой подход не является необходимым для данной предметной области.

1. Accounts

* Поле iban может быть вынесено в отдельную таблицу для хранения его изменений, но это не нужно для данной предметной области.

1. Address

* Поля street, city, state, postal\_code, country могут быть вынесены в отдельные таблицы, но такой подход не является необходимым для данной предметной области.

1. Можно ли добиться некоторого повышения производительности за счёт денормализации схемы? Обоснуйте своё мнение.
2. Таблицы Accounts и Transactions:

* Если поля из таблицы Currencies, такие как currency\_name, часто запрашиваются вместе с записями в Accounts и Transactions, то добавление currency\_name непосредственно в Accounts и Transactions избавит от необходимости выполнения JOIN.

1. Таблица TransactionStatusHistory:

* В этой таблице можно добавить status\_name из таблицы TransactionStatuses. Это может повысить скорость запросов, где требуется информация о статусе транзакции. Вместо дополнительных соединений с таблицей TransactionStatuses данные будут доступны напрямую.

1. Таблицм Accounts и Cards:

* Если поле status\_name из таблицы AccountCardStatuses часто используется вместе с данными счета и карты, можно добавить его непосредственно в Accounts и Cards. Это позволит быстрее получать статус счета и карты без дополнительных операций соединения.

1. Таблица Owners:

* Если данные о Address (например, город или страна) часто требуются в запросах к Owners, то денормализация и добавление этих полей непосредственно в Owners может повысить производительность за счёт уменьшения количества операций соединения с таблицей Address.

1. Можно ли добиться некоторого повышения производительности, добавив в схему кэширующие отношения? Обоснуйте своё мнение.
2. Сущность Transactions:

* Если часто требуется агрегировать транзакции по суммам, можно добавить кэшируемое поле для хранения общей суммы транзакций за определённый период. Это позволит избежать постоянных суммирований в реальном времени, особенно если база данных масштабируется и количество транзакций увеличивается.

1. Сущность Accounts:

* Если часто требуется информация о количестве активных карт для каждого аккаунта, можно добавить кэширующее поле, которое будет хранить это значение в таблице Accounts. Это поле можно обновлять при добавлении или удалении карт, относящихся к конкретному аккаунту.

1. Сущность TransactionStatusHistory:

* Если требуется получать последнее состояние транзакции, можно добавить поле в Transactions, которое бы хранило актуальный статус транзакции. Это убережет от необходимости делать соединения с таблицей TransactionStatusHistory при каждом запросе.

1. Сущность Users и роли:

* Если часто нужны отчеты по количеству пользователей определенной роли, добавление кэшируемого поля с количеством пользователей на каждый role\_id в таблице Roles ускорит отчеты и выборки.

1. Добавьте в базу данных все необходимые индексы, представления, хранимые процедуры и т. д.

Сущность Roles

* unq\_role\_name - уникальный индекс для role\_name.

Сущность Users

* idx\_user\_role\_id - индекс для связи с таблицей Roles по role\_id.
* unq\_username - уникальный индекс для username.

Сущность Currencies

* unq\_currency\_name - уникальный индекс для currency\_name.

Сущность AccountCardStatuses

* unq\_status\_name - уникальный индекс для status\_name.

Сущность Accounts

* idx\_account\_currency\_code - индекс для связи с Currencies по currency\_code.
* idx\_account\_status\_id - индекс для связи с AccountCardStatuses по status\_id.
* idx\_account\_user\_id - индекс для связи с Users по user\_id.
* unq\_account\_iban - уникальный индекс для iban.

Сущность Address

* idx\_address\_country\_state - составной индекс по country и state.

Сущность Owners

* idx\_owner\_address\_id - индекс для связи с Address по address\_id.
* unq\_owner\_passport\_id - уникальный индекс для passport\_id.
* unq\_owner\_email - уникальный индекс для email.
* unq\_owner\_phone - уникальный индекс для phone.

Сущность Cards

* idx\_card\_account\_id - индекс для связи с Accounts по account\_id.
* idx\_card\_status\_id - индекс для связи с AccountCardStatuses по status\_id.
* unq\_card\_number - уникальный индекс для card\_number.

Сущность Transactions

* idx\_transaction\_currency\_code - индекс для связи с Currencies по currency\_code.
* idx\_transaction\_date - индекс для ускорения запросов по transaction\_date.

Сущность TransactionStatuses

* unq\_transaction\_status\_name - уникальный индекс для status\_name.

Сущность TransactionStatusHistory

* idx\_history\_transaction\_id - индекс для связи с Transactions по transaction\_id.
* idx\_history\_transaction\_status\_id - индекс для связи с TransactionStatuses по transaction\_status\_id.

# схема базы данных

