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

1 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.

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

Аномалий вставки, удаления и обновления не были обнаружены.

3. 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.

4. 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

5. Существуют ли отношения, имеющие многозначные зависимости?

В данной схеме многозначные зависимости отсутствуют.

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

Данная схема не нарушает никаких требований нормализации.

7. 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.

8. 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.

9. Есть ли отношения с возможной, но ненужной дальнейшей нормализацией?

1. Users
   - Поле username можно вынести в отдельную таблицу для отслеживания истории имен, но это избыточно и не требуется в данном случае.
2. 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.
3. Accounts
   - Поле iban может быть вынесено в отдельную таблицу для хранения его изменений, но это не нужно для данной предметной области.
4. Address
   - Поля street, city, state, postal_code, country могут быть вынесены в отдельные таблицы, но такой подход не является необходимым для данной предметной области.

10. Можно ли добиться некоторого повышения производительности за счёт денормализации схемы? Обоснуйте своё мнение.

1. 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.

2. 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.

3. 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.
4. 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.

11. Можно ли добиться некоторого повышения производительности, добавив в схему кэширующие отношения? Обоснуйте своё мнение.

1. 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.
2. 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.
3. 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.
4. 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.

12. 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.

2 СХЕМА БАЗЫ ДАННЫХ