БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет КСиС

Специальность ПОИТ

Лабораторная работа №2

«Доработка модели базы данных»

по дисциплине «Базы данных»

Выполнил студент: Трухан К.А.

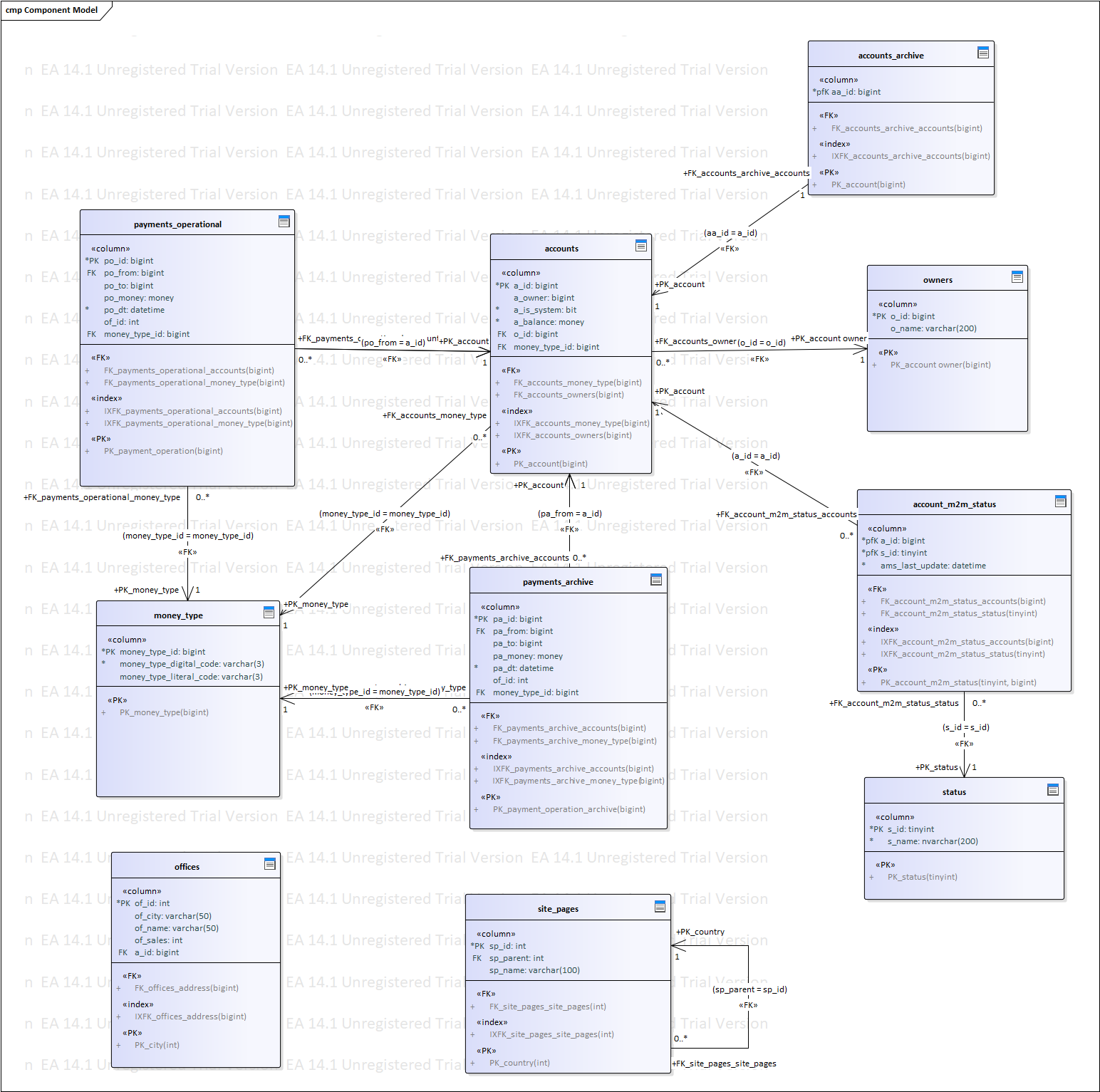
группа 851002

Проверил: Фадеева Е.Е.

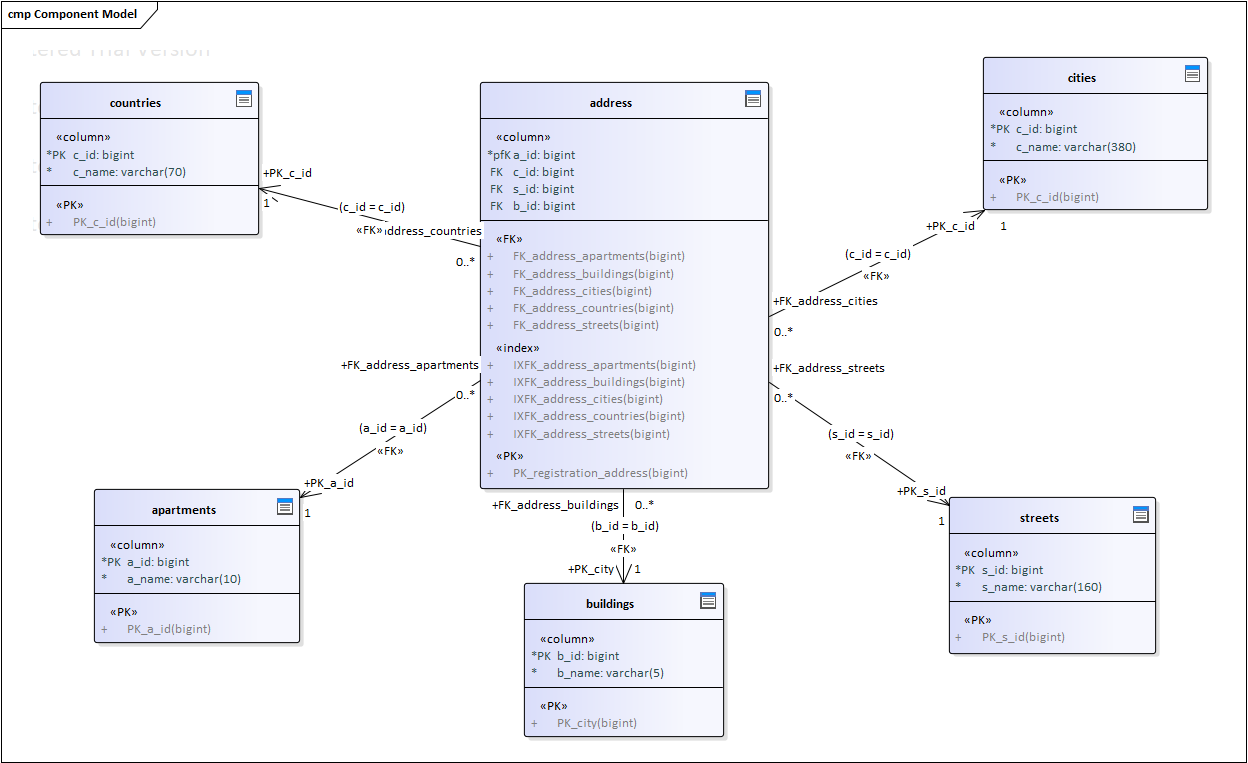
Минск 2021

Исходная база данных «Банк»

1. Какие каскадные операции необходимы в этой базе данных? Опишите их.
   1. Сущность «payments\_operational» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   2. Сущность «payments\_archive» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   3. Сущность «accounts»:
      * a\_id – запрет каскадного удаления:
        + атрибут используется в сущностях «payments\_operational», «payments\_archive» и «accounts\_m2m\_statuses» и определяет счет;
        + информация о счете при удалении не должна быть потерна;
        + при использовании операции «запрет каскадного удаления» в случае выполнения транзакции в течение месяца закрыть счет в банке невозможно. Если данная проблема критична, использовать каскадные операции не нужно, однако ссылочная целостность будет нарушена.
   4. Сущность «owner»:
      * o\_id – запрет каскадного удаления:
        + атрибут используется в сущности «accounts»;
        + если производится операция удаление владельца счета, следовательно, сам счет должен быть так же удален.
   5. Сущность «offices» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   6. Сущность «site\_pages» – каскадное удаление и обновление:
      * sp\_id – каскадное удаление:
        + при удалении страницы необходимо удалить все под страницы.
   7. Сущность «statuses» – каскадное обновление и запрет на каскадное удаление:
      * атрибут используется в сущностях «accounts\_m2m\_statuses»;
      * при необходимости изменения статуса счета, необходимо произвести обновление данных.
      * нельзя удалять статус, пока к нему привязан хоть 1 счет банка.
2. Существует ли возможность аномалий операций вставки, обновления, удаления данных? Составьте список таких случаев и внесите в базу данных исправления, позволяющие избежать таких аномалий.
   1. Сущность «payments\_operational»:
      * Аномалия добавления данных:
        + при добавлении, значение атрибута «money» невозможно определить: неизвестно в какой валюте представлено значение.
      * Аномалия выборки данных:
        + при извлечении значения атрибута «money» невозможно определить в какой валюте представлено значение.
      * Аномалия обновлениях данных:
        + при обновлении значения атрибута «money» невозможно определить в какой валюте представлять значение.
   2. Сущность «payments\_archive»:
      * Аномалия добавления данных:
        + при добавлении, значение атрибута «money» невозможно определить в какой валюте представлено значение.
      * Аномалия выборки данных:
        + при добавлении, значение атрибута «money» невозможно определить: неизвестно в какой валюте представлено значение.
      * Аномалия обновлениях данных:
        + при обновлении значения атрибута «money» невозможно определить в какой валюте представлять значение.
   3. Сущность «accounts»:
      * Аномалия удаления данных:
        + при попытке удаление счета и использовании каскадной операции «запрет каскадного удаления» в случае, когда данный счет связан с одним из атрибутов в «payment\_archive», удаление не может быть произведено.
        + предыдущая проблема порождает новую. Удаление также не будет производится в сущности «account\_m2m\_statuses».
      * Аномалия выборки данных:
        + при извлечении значения атрибута «money» невозможно определить в какой валюте представлено значение.
      * Аномалия обновлениях данных:
        + при обновлении значения атрибута «money» невозможно определить в какой валюте представлять значение.
      * Аномалия добавления данных:
        + при добавлении, значение атрибута «money» невозможно определить: неизвестно в какой валюте представлено значение.
   4. Сущность «owner»:
      * не обнаружено.
   5. Сущность «offices»
      * описание атрибута «of\_city» плохо детализировано. В результате, нельзя однозначно определить: в каком формате представлено местоположение офиса. Представь, что формат атрибута таков: «страна\_город\_улица\_дом\_офис».
      * Аномалия вставки:
        + при вставке необходимо определять значение атрибута «of\_sales», хотя офис может изначально не иметь прибыли.
   6. Сущность «site\_pages»:
      * не обнаружено.
   7. Сущность «statuses»:
      * не обнаружено.



1. Можно ли использовать схемы «звезда» или «снежинка» с этой базой данных, чтобы избежать некоторых аномалий операций с данными? Переработайте схему, сравните новую с исходной и составьте список аномалий работы с данными, которые были устранены (или, наоборот, появились).
   1. Для данной базы данных для всех сущностей невозможно применить топологию «звезда» или «снежинка». Сущность «site\_pages» логически невозможно связать с остальными сущностями.
   2. Топологию «звезда» невозможно применить для всей БД без серьезной переработки архитектуру БД.
   3. Топологию «звезда» можно применить для атрибута «of\_city» сущности «offices». Это позволить лучше детализировать местоположение офиса. Такая топология позволит лучше выполнять операции выборки, сделает схему БД, более структурированной и, в теории, оптимизирует потребление памяти на хранение адресов офисов.



Представление адреса в топологии «звезда»

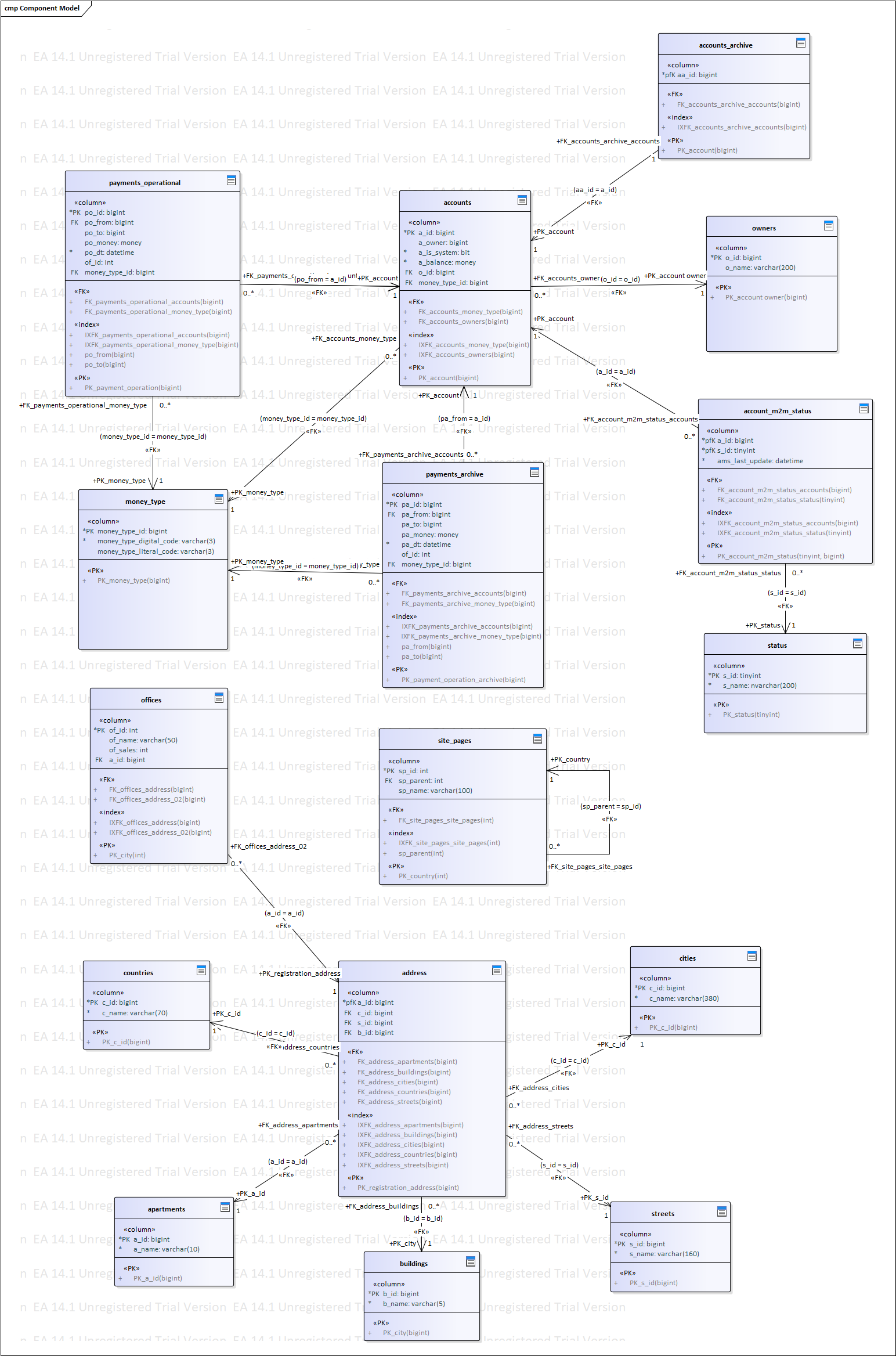
* 1. Полная схема БД в топологии «звезда».



Полная схема БД в топологии звезда

* 1. Такая топология порождает множество новых проблем и аномалий:
     + Аномалии, которые были устранены в пункте 2, вернулись;
     + Теперь невозможно хранить архив транзакций;
     + Сущность «accounts»:
       - Аномалия удаления:
         1. теряется информация о владельце счета.
       - Аномалия вставки:
         1. владелец счета не может иметь 2 счета одинакового типа.

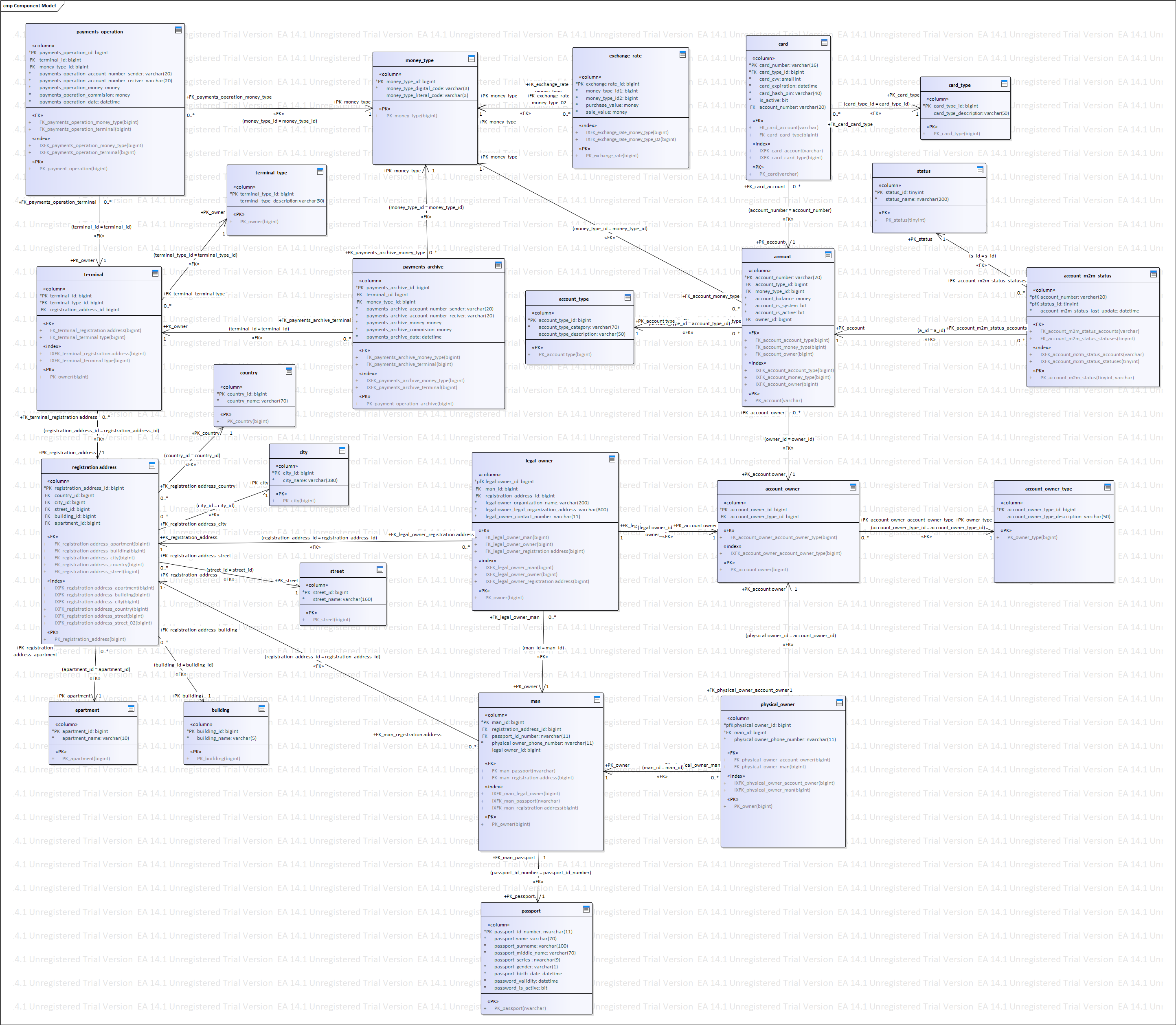
1. Составьте список всех функциональных зависимостей в базе данных.
   1. Сущность «payments\_operational» – все атрибуты функционально зависят от первичного ключа.
   2. Сущность «payments\_archive» – все атрибуты функционально зависят от первичного ключа.
   3. Сущность «accounts» – все атрибуты функционально зависят от первичного ключа.
   4. Сущность «owner» – все атрибуты функционально зависят от первичного ключа.
   5. Сущность «offices»:
      * атрибут «of\_sales» функционально зависит от атрибутов «of\_city» и «of\_name».
   6. Сущность «site\_pages» – все атрибуты функционально зависят от первичного ключа.
   7. Сущность «statuses» – каскадное обновление и запрет на каскадное удаление – все атрибуты функционально зависят от первичного ключа.
2. Существуют ли отношения, имеющие многозначные зависимости? Если «да», как можно переработать схему, чтобы избежать таких зависимостей?
   1. Нет, таких зависимостей не существует.
3. Нарушает ли схема какие-либо «требования нормализации»? Если «да», доработайте схему, чтобы избежать таких нарушений.
   1. Требование производительности системы:
      * Часть атрибутов имеют тип «bigint», хотя оптимально использовать тип «vchar(n)».
   2. Требование минимальности первичных ключей:
      * Размер атрибута «of\_name» сущности «offices» необходима увеличить;
      * Размер атрибута «o\_name» сущности «owner» необходима увеличить.
   3. Требование единому соглашению именования.
4. Существуют ли какие-либо потенциальные проблемы с производительностью базы данных? Если «да», запишите их.
   1. Часть атрибутов имеют тип «bigint», хотя оптимально использовать тип «vchar(n)»;
   2. Размер атрибута «s\_name» сущности «status» можно уменьшить;
   3. В БД не используются кеширующие и агрегирующее таблицы. При хранение большого количества данных в БД, операции удаление, выборки, поиска, сортировки и т.д. будут выполнятся медленно.
   4. Остальных проблем не обнаружено.
   5. Написать про индексы
5. Для каждого отношения в базе данных определите, в какой нормальной форме оно находится. Запишите ответ.
   1. Сущность «payments\_operational» – 3НФ в изначальной схеме и 2НФ после устранения аномалий.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «money\_type\_id» транзитивно зависит от «po\_id» через «po\_money».
   2. Сущность «payments\_archive» – 3НФ в изначальной схеме и 2НФ после устранения аномалий.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «money\_type\_id» транзитивно зависит от «pa\_id» через «pa\_money».
   3. Сущность «accounts» – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «a\_is\_system» транзитивно зависит от «a\_id» через «a\_owner».
   4. Сущность «owner» – 0НФ.
      * атрибут «o\_name» не атомарен.
   5. Сущность «offices» – 0НФ.
      * атрибут «of\_sales» функционально зависит от атрибутов «of\_city» и «of\_name».
      * атрибут «of\_name» и «of\_city» не атомарен.
   6. Сущность «site\_pages» 3НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * транзитивных зависимостей нет.
   7. Сущность «statuses» – 3НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * транзитивных зависимостей нет.
6. Есть ли отношения с возможной, но ненужной дальнейшей нормализацией? Составьте список.
   1. Сущность «payments\_operational» после устранения аномалий.
      * Можно привести сущность в 3НФ, вынеся атрибуты «money\_type\_id» и «po\_money» в отдельную сущность.
   2. Сущность «owner».
      * Можно сделать атрибут «o\_name» атомарным, детализирую информацию в отдельной сущности.
   3. Сущность «offices»
      * Можно сделать атрибут «o\_name» атомарным, детализирую информацию в отдельной сущности. Однако такая нормализация сделает архитектуры БД более гибкой. Вывод: это нормализация нужна.
7. Можно ли добиться некоторого повышения производительности за счёт денормализация схемы? Обоснуйте своё мнение.
   1. Поскольку в БД нет сущностей, находящихся в большей нормальной форме чем третья, можно сделать вывод, что денормализация схемы не требуется.
      * выигрыш в производительности будет мал.
      * образуется множество проблем в архитектуре БД.
      * использовать БД будет достаточно проблематично.
8. Можно ли добиться некоторого повышения производительности, добавив в схему кэширующие отношения? Обоснуйте своё мнение.
   1. можно создать кэширующие таблицы для сущности «payment\_archive». В них будут хранится транзакции по датам.
   2. можно создать кэширующую таблицу для сущности «accounts». В ней будет хранится счета с флагом «a\_is\_system» = true для ускорения поиска.
9. Добавьте в базу данных все необходимые индексы, представления, хранимые процедуры и т. д.
   1. Сущность «payments\_operational»:
      * добавить индекс на атрибут «po\_from» для ускорения поиска;
      * добавить индекс на атрибут «po\_to» для ускорения поиска;
      * процедура переноса транзакции в архив.
   2. Сущность «payments\_archive»:
      * добавить индекс на атрибут «pa\_from» для ускорения поиска;
      * добавить индекс на атрибут «pa\_to» для ускорения поиска.
      * процедура удаление из архива транзакций, чей срок хранения истек.
   3. Сущность «accounts»:
      * представление получения полной информации о счете.
   4. Сущность «owner»:
      * Добавить индекс на атрибут «o\_name» для ускорения поиска.
   5. Сущность «offices»:
      * представление получения полной информации о офисе.
   6. Сущность «address»:
      * представление получения полной информации о адресе.
   7. Сущность «site\_pages»:
      * добавить индекс на атрибут «sp\_parent» для ускорения поиска.



Финальная схема БД «Банк»

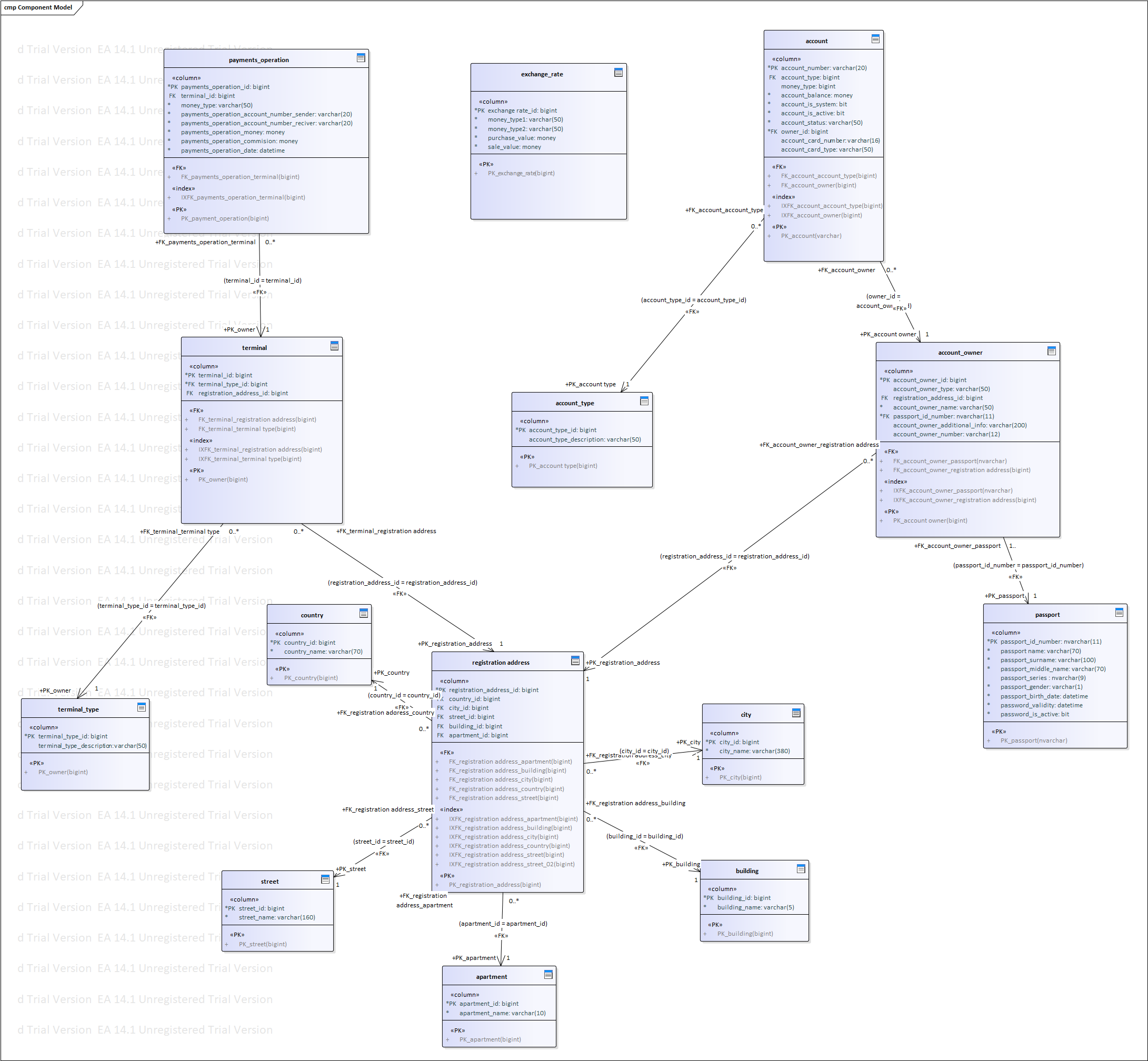
Исправленная база данных «Банк»

1. Какие каскадные операции необходимы в этой базе данных? Опишите их.
   1. Сущность «account»:
      * account\_id – каскадное удаление и обновление. При удалении кортежа, должны удалится все связанные значения в других сущностях.
   2. Сущность «account\_type»:
      * account\_type\_id – запрет каскадной операции:
        + пока с типом счета связано хоть 1 значение, удалять тип нельзя.
   3. Сущность «card» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   4. Сущность «card\_type»:
      * card\_type\_id – каскадное обновление и запрет каскадной операции:
        + пока с типом карты связано хоть 1 значение, удалять тип нельзя.
   5. Сущность «account\_m2m\_status» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   6. Сущность «status»:
      * status \_type\_id – запрет каскадной операции:
        + пока с типом карты связано хоть 1 значение, удалять тип нельзя.
   7. Сущность «money\_type»:
      * money\_type\_id – запрет каскадной операции:
        + пока с типом валюта связано хоть 1 значение счета банка, удалять тип нельзя.
   8. Сущность «exchange\_rate – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   9. Сущность «payments\_operation» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   10. Сущность «payments\_operational\_archive» – каскадные операции не требуются поскольку атрибуты сущности не используются как FK.
   11. Сущность «terminal» – установка пустых ключей.
       * terminal\_id:
         + при удалении терминала нельзя удалять связанные с ним транзакции.
         + так же нельзя установить операцию «запрет на каскадное удаление». В архиве транзакциях могут хранится значения, связанные с терминалом. В таком случае терминал удалить невозможно.
         + Установить значения по умолчанию невозможно. (отсутствует)
   12. Сущность «terminal\_type»:
       * terminal\_type\_id – запрет каскадной операции:
         + пока с типом терминала связано хоть 1 значение терминала, удалять тип нельзя.
   13. Сущность «account\_owner»:
       * account\_owner\_id – каскадное удаление:
         + при удалении владельца аккаунта необходимо удалить связанные с ним счета.
   14. Сущность «account\_owner\_type»:
       * account\_owner\_type\_id – запрет каскадной операции:
         + пока с типом связано хоть 1 значение владельца, удалять тип нельзя.
   15. Сущность «legal\_owner»:
       * legal\_owner\_id – каскадное удаление:
         + если удаляется юридическое лицо, должен быть удален связанный с ним счет.
   16. Сущность «physical\_owner»:
       * physical\_owner\_id – каскадное удаление:
         + если удаляется физический лицо, должен быть удален связанный с ним счет.
         + однако, если удаляется физическое лицо, связанное юридическое лицо также удаляется. То есть в данной БД существует аномалия удаления данных.
   17. Сущность «passport»:
       * passport\_id\_number – каскадное удаление и обновление:
         + паспортные данные удаляются в случае, если с ними нет связанных владельцев.
   18. Сущность «registration\_address»:
       * registration\_address\_id – установка пустых ключей:
         + если терминал физический, у него должен быть адрес.
         + если терминал виртуальный, физического адреса быть не может. В таком случае, адрес = NULL.
   19. Сущность «country»:
       * country\_id – каскадное удаление:
         + при удалении части адреса, безопаснее удалить весь адрес.
         + при установке значений по умолчанию или NULL-значений адрес будет все равно потерян.
   20. Сущность «city»:
       * city\_id – каскадное удаление:
         + при удалении части адреса, безопаснее удалить весь адрес.
         + при установке значений по умолчанию или NULL-значений адрес будет все равно потерян.
   21. Сущность «street»:
       * street\_id – каскадное удаление:
         + при удалении части адреса, безопаснее удалить весь адрес.
         + при установке значений по умолчанию или NULL-значений адрес будет все равно потерян.
   22. Сущность «building»:
       * building\_id – каскадное удаление:
         + при удалении части адреса, безопаснее удалить весь адрес.
         + при установке значений по умолчанию или NULL-значений адрес будет все равно потерян.
   23. Сущность «apartment»:
       * apartment\_id – каскадное удаление:
         + при удалении части адреса, безопаснее удалить весь адрес.
         + при установке значений по умолчанию или NULL-значений адрес будет все равно потерян.
2. Существует ли возможность аномалий операций вставки, обновления, удаления данных? Составьте список таких случаев и внесите в базу данных исправления, позволяющие избежать таких аномалий.
   1. Сущность «account»:
      * Аномалия вставки:
        + при вставке значений необходимо заранее знать значение атрибута «account\_balance». Если значение неизвестно, придётся вставлять значение по умолчанию.
   2. Сущность «account\_type».
   3. Сущность «card».
      * Аномалия обновления и выборки:
        + у банковской карты есть строк действия, следовательно, по истечению срока карта не должна работать. Для определения активна ли карта необходимо каждый раз сравнивать дату с сегодняшней датой, что может затрачивать много ресурсов.
          - Ввести флаг. Операции с флагом требует меньше ресурсов.
   4. Сущность «card\_type».
   5. Сущность «account\_m2m\_status».
   6. Сущность «status».
   7. Сущность «money\_type».
   8. Сущность «exchange\_rate».
   9. Сущность «payments\_operation»:
      * Аномалия вставки:
        + при вставке непонятно, в какой валюте определять значение денежного перевода.
          - необходимо добавить атрибут тип валюты.
      * Аномалия удаления:
        + потеря информации о взятой с транзакции комиссии, хранящейся в атрибуте «payments\_operation\_commision».
          - заказчиком не было определено, что эти данные нужно хранить, поэтому данную аномалию можно оставить.
   10. Сущность «payments\_operational\_archive»:
       * Аномалия удаления:
         + потеря информации о взятой с транзакции комиссии, хранящейся в атрибуте «payments\_operation\_commision».
           - заказчиком не было определено, что эти данные нужно хранить, поэтому данную аномалию можно оставить.
   11. Сущность «terminal»
   12. Сущность «terminal\_type»:
   13. Сущность «account\_owner»:
   14. Сущность «account\_owner\_type»:
   15. Сущность «legal\_owner»
   16. Сущность «physical\_owner»:
       * Аномалия удаления:
         + если у человека является физическим лицом (имеется счет) и юридическим лицом (владелец счета), то при закрытии счета физического лица данные о владельца юридического лица будут потеряны.
           - изменить архитектуру, добавив сущностей «man».
   17. Сущность «passport»:
   18. Сущность «registration\_address»:
   19. Сущность «country»:
   20. Сущность «city»:
   21. Сущность «street»:
   22. Сущность «building»:
   23. Сущность «apartment»:



Результат изменения БД

1. Можно ли использовать схемы «звезда» или «снежинка» с этой базой данных, чтобы избежать некоторых аномалий операций с данными? Переработайте схему, сравните новую с исходной и составьте список аномалий работы с данными, которые были устранены (или, наоборот, появились).

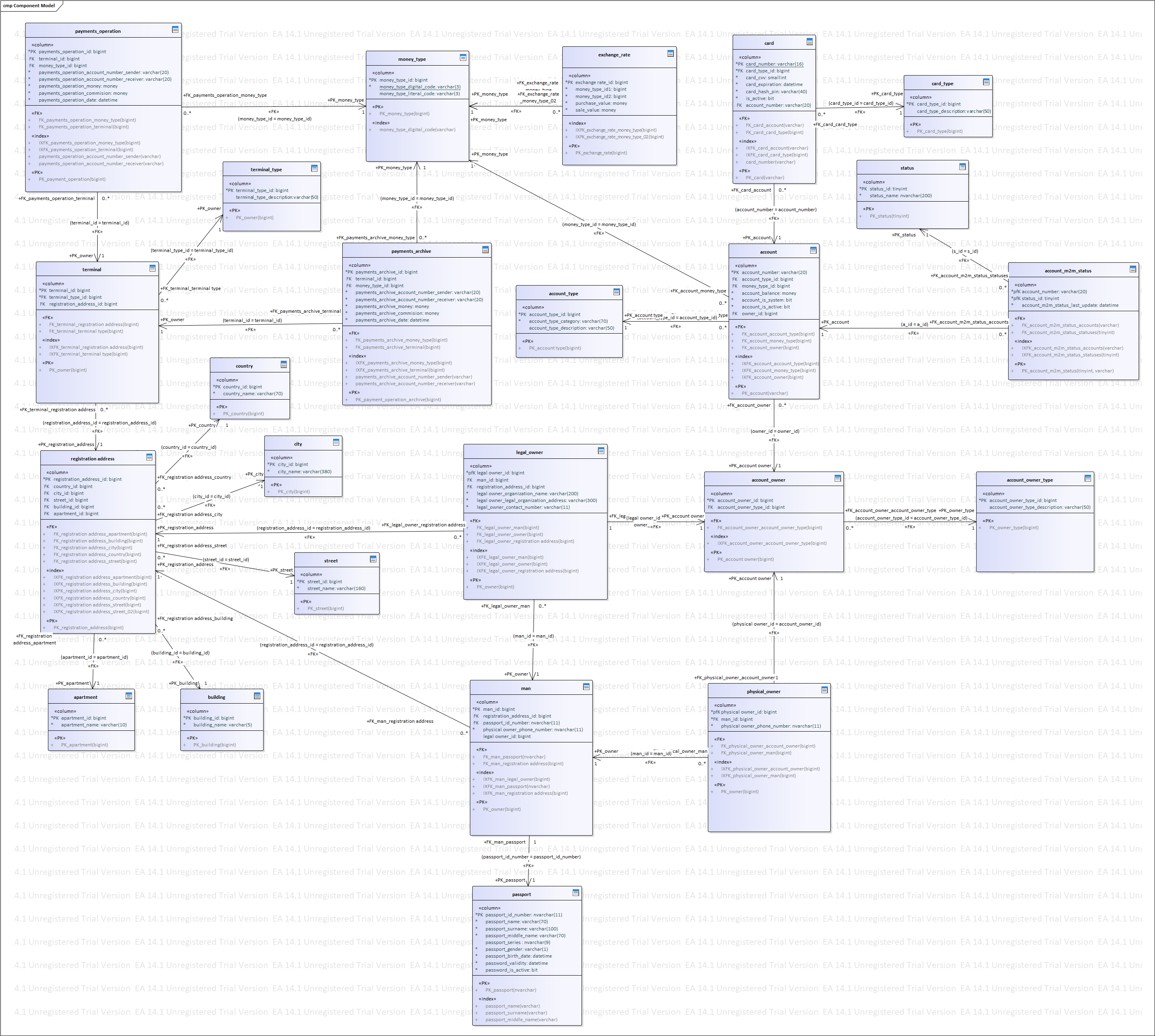


Данная схема несет множество проблем и порождаем различные аномалии.

* 1. Сущность «account»:
     + Аномалия вставки:
       - при вставке часть значений атрибутов неизвестно.
     + Аномалия удаления:
       - При удалении возможно потеря данных атрибутов «account\_card\_number», «account\_card\_type», «account\_money\_type» и «account\_status».
     + Аномалия обновления:
       - При обновлении атрибута «account\_money\_type» необходимо обновлять так же значение атрибута «account\_balance» в соответствующей валюте.
  2. Сущность «account\_type».
  3. Сущность «exchange\_rate».
     + Аномалия вставки:
       - Значение атрибутов тип валюты нигде не указано и возможно ошибка при вставке.
     + Аномалия удаления:
       - При удалении возможна потеря типов валюты.
     + Аномалия обновление
  4. Сущность «payments\_operation»:
     + Старые аномалии.
     + Аномалия вставки и удаление типа «money\_type». (описание проблемы выше)
  5. Сущность «terminal».
  6. Сущность «terminal\_type».
  7. Сущность «account\_owner»:
     + Аномалии вставки:
       - Множество возможных типов владельцев счета не определено, а значит при определении значения атрибута «account\_owner\_type» может возникнуть ошибка.
     + Аномалии обновления:
       - Поскольку множество атрибутов не атомарны, возможна потеря информации при попытке обновления атрибута.
  8. Сущность «passport».
  9. Сущность «registration\_address».
  10. Сущность «country».
  11. Сущность «city».
  12. Сущность «street».
  13. Сущность «building».
  14. Сущность «apartment».
  15. Данная БД нарушает много правил нормализации и порождает дополнительные проблемы:
      + отсутствует архив транзакций;
      + низкий уровень нормализации;
        - сущности в основном находятся в 0НФ и 1НФ.
      + Удобство пользования БД меньше, так как отсутствует множества типов различных сущностей.
      + вероятность возникновения скрытых аномалий увеличена.
      + разделение владельца счета на юридическое и юридическое лицо потеряно;
      + требования неизбыточности данных не выполняется;
      + и т.д.

1. Составьте список всех функциональных зависимостей в базе данных.
   1. Сущность «account»:
      * все атрибуты функционально зависят от первичного ключа.
   2. Сущность «account\_type»:
      * все атрибуты функционально зависят от первичного ключа.
   3. Сущность «card»:
      * атрибут «card\_is\_active» транзитивно зависит от «card\_id» через «card\_expiration».
   4. Сущность «card\_type»:
      * все атрибуты функционально зависят от первичного ключа.
   5. Сущность «account\_m2m\_status»:
      * все атрибуты функционально зависят от первичного ключа.
   6. Сущность «status»:
      * все атрибуты функционально зависят от первичного ключа.
   7. Сущность «money\_type»:
      * атрибут «money\_type\_literal\_code» транзитивно зависит от «money\_type\_id» через «money\_type\_digital\_code».
      * остальные атрибуты функционально зависят от первичного ключа.
   8. Сущность «exchange\_rate»:
      * атрибут «purchase\_value» зависит от «money\_type\_id1»;
      * атрибут «sale\_value» зависит от «money\_type\_id2».
   9. Сущность «payments\_operation»:
      * атрибут «payments\_operation\_money» зависит от «money\_type\_id»;
      * атрибут «payments\_operation\_commision» зависит от «money\_type\_id».
   10. Сущность «payments\_operational\_archive»:
       * атрибут «payments\_operational\_archive\_money» зависит от «money\_type\_id»;
       * атрибут «payments\_operational\_archive\_commision» зависит от «money\_type\_id».
   11. Сущность «terminal»:
       * все атрибуты функционально зависят от первичного ключа.
   12. Сущность «terminal\_type»:
       * все атрибуты функционально зависят от первичного ключа.
   13. Сущность «account\_owner»:
       * все атрибуты функционально зависят от первичного ключа.
   14. Сущность «account\_owner\_type»:
       * все атрибуты функционально зависят от первичного ключа.
   15. Сущность «legal\_owner»:
       * все атрибуты функционально зависят от первичного ключа.
   16. Сущность «physical\_owner»:
       * все атрибуты функционально зависят от первичного ключа.
   17. Сущность «passport»:
       * атрибут «password\_is\_active» транзитивно зависит от «passport\_id\_number» через «password\_validity».
   18. Сущность «registration\_address»:
       * все атрибуты функционально зависят от первичного ключа.
   19. Сущность «country»:
       * все атрибуты функционально зависят от первичного ключа.
   20. Сущность «city»:
       * все атрибуты функционально зависят от первичного ключа.
   21. Сущность «street»:
       * все атрибуты функционально зависят от первичного ключа.
   22. Сущность «building»:
       * все атрибуты функционально зависят от первичного ключа.
   23. Сущность «apartment»:
       * все атрибуты функционально зависят от первичного ключа.
2. Существуют ли отношения, имеющие многозначные зависимости? Если «да», как можно переработать схему, чтобы избежать таких зависимостей?
   1. Нет, таких зависимостей не существует.
3. Нарушает ли схема какие-либо «требования нормализации»? Если «да», доработайте схему, чтобы избежать таких нарушений.
   1. Требование производительности системы
      * на атрибуты, часто участвующие в операциях поиска необходимо добавить индексы.
   2. Требование единого соглашения именования.
      * индекс сущности «apartment» называется «PK\_city\_id».
      * и схожие проблемы.
4. Существуют ли какие-либо потенциальные проблемы с производительностью базы данных? Если «да», запишите их.
   1. При больших объёмах данных из-за недостатка индексов операции поиска могут выполнятся слишком медленно.
5. Для каждого отношения в базе данных определите, в какой нормальной форме оно находится. Запишите ответ.
   1. Сущность «account» – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «account\_type» транзитивно зависит от «account\_number» через «owner\_id».
   2. Сущность «account\_type» – 3НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * транзитивных зависимостей нет.
   3. Сущность «card» – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «card\_is\_active» транзитивно зависит от «card\_id» через «card\_expiration».
   4. Сущность «card\_type» – 3НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * транзитивных зависимостей нет.
   5. Сущность «money\_type» – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «money\_type\_literal\_code» транзитивно зависит от «money\_type\_id» через «money\_type\_digital\_code».
   6. Сущность «exchange\_rate» – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «purchase\_value» зависит от «money\_type\_id1»;
      * атрибут «sale\_value» зависит от «money\_type\_id2».
   7. Сущность «payments\_operation» – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «payments\_operation\_money» зависит от «money\_type\_id»;
      * атрибут «payments\_operation\_commision» зависит от «money\_type\_id».
   8. Сущность «payments\_operational\_archive – 2НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * атрибут «payments\_operational\_archive\_money» зависит от «money\_type\_id»;
      * атрибут «payments\_operational\_archive\_commision» зависит от «money\_type\_id».
   9. Сущность «terminal» – 3НФ.
      * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
      * транзитивных зависимостей нет.
   10. Сущность «terminal\_type» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет.
   11. Сущность «account\_owner» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет.
   12. Сущность «account\_owner\_type» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет.
   13. Сущность «legal\_owner» – 1НФ.
       * В зависимости от требований заказчика, атрибут «legal\_owner\_contact\_number» можно считать не атомарным;
   14. Сущность «physical\_owner» – 1НФ.
       * В зависимости от требований заказчика, атрибут «legal\_owner\_contact\_number» можно считать не атомарным;
   15. Сущность «passport» – 2НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * атрибут «password\_is\_active» транзитивно зависит от «password\_id\_number» через «password\_validity».
   16. Сущность «registration\_address» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет;
   17. Сущность «country» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет;
   18. Сущность «city» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет;
   19. Сущность «street» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет;
   20. Сущность «building» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет;
   21. Сущность «apartment» – 3НФ.
       * все атрибуты сущности атомарные, следовательно, 1НФ соблюдена;
       * транзитивных зависимостей нет.
6. Есть ли отношения с возможной, но ненужной дальнейшей нормализацией? Составьте список.
   1. Сущность «card».
      * необходимо вынести транзитивную зависимость в отдельную сущность;
      * данная нормализация не требуется, так как плохо отразится на операциях выборки, поиска и т.д.;
   2. Сущность «money\_type».
      * Можно изменить первичный ключ на атрибут «digital\_code».
      * данная нормализацию можно сделать, не обязательно:
        + плюс. Первичный ключ стане естественным.
        + минус. Может возникнуть аномалия обновления.
   3. Сущность «payments\_operation».
      * необходимо вынести транзитивные зависимости в отдельную сущность;
      * данная нормализация не требуется, так как плохо отразится на операциях выборки, добавление, удаления, поиска и т.д.;
   4. Сущность «payments\_operational\_archive.
      * необходимо вынести транзитивные зависимости в отдельную сущность;
      * данная нормализация не требуется, так как плохо отразится на операциях выборки, добавление, удаления, поиска и т.д.;
   5. Сущность «legal\_owner»:
      * атрибут «legal\_owner\_contact\_number» можно сделать атомарным, создав отдельную таблицу;
      * данная нормализация не требуется, так как заказчик не указывал необходимость выборки из БД отдельных частей номера.
   6. Сущность «physical\_owner»:
      * атрибут «physical\_owner \_contact\_number» можно сделать атомарным, создав отдельную таблицу;
      * данная нормализация не требуется, так как заказчик не указывал необходимость выборки из БД отдельных частей номера.
   7. Сущность «passport»:
      * необходимо вынести транзитивную зависимость в отдельную сущность;
      * данная нормализация не требуется, так как плохо отразится на операциях выборки поиска и т.д.;
      * преимущества новой архитектуры БД незначительны.
7. Можно ли добиться некоторого повышения производительности за счёт деморализации схемы? Обоснуйте своё мнение.
   1. Поскольку в БД нет сущностей, находящихся в большей нормальной форме чем третья, можно сделать вывод, что *деморализация схемы* не требуется.
      * выигрыш в производительности будет мал;
      * образуется множество проблем в архитектуре БД;
      * использовать БД будет достаточно проблематично.

1. Можно ли добиться некоторого повышения производительности, добавив в схему кэширующие отношения? Обоснуйте своё мнение.
   1. Можно добавить в схему кэширующее отношение адресов для более быстрого обращения;
   2. Можно добавить кэширующие отношение списка транзакций за определенных промежуток времени или определенного отправителя/получателя;
   3. Можно добавить кэширующее отношение владельца аккаунта для более быстрой операции выборки данных.
2. Добавьте в базу данных все необходимые индексы, представления, хранимые процедуры и т. д.
   1. Сущность «account»:
      * представление получения полной информации о счете.
   2. Сущность «account\_type».
   3. Сущность «card»:
      * добавить уникальный индекс на атрибут «card\_number» для ускорения поиска карты;
      * триггер, изменяющий значение атрибута «card\_is\_active» по истечению срока эксплуатации, хранимого в атрибуте «card\_expiration».
   4. Сущность «card\_type».
   5. Сущность «account\_m2m\_status».
   6. Сущность «money\_type»:
      * добавить уникальный индекс на атрибут «card\_digital\_code» для ускорения поиска валюты;
   7. Сущность «exchange\_rate».
   8. Сущность «payments\_operation»:
      * добавить индекс на атрибут «payments\_operation\_sender» для ускорения поиска;
      * добавить индекс на атрибут «payments\_operation\_receiver» для ускорения поиска;
      * процедура переноса транзакции в архив.
   9. Сущность «payments\_operational\_archive»:
      * добавить индекс на атрибут «payments\_operational\_archive \_sender» для ускорения поиска;
      * добавить индекс на атрибут «payments\_operational\_archive \_receiver» для ускорения поиска.
      * процедура удаление из архива транзакций, чей срок хранения истек.
   10. Сущность «terminal»:
       * представление получения полной информации о терминале.
   11. Сущность «terminal\_type».
   12. Сущность «account\_owner»:
       * представление получения полной информации о владельце счета.
   13. Сущность «account\_owner\_type».
   14. Сущность «legal\_owner»:
       * добавить индекс на атрибут «legal\_owner\_organization\_name» для ускорения поиска организации.
   15. Сущность «physical\_owner».
   16. Сущность «passport»:
       * добавить индекс на атрибут «passport\_name» для ускорения поиска;
       * добавить индекс на атрибут «passport\_surname» для ускорения поиска;
       * добавить индекс на атрибут «passport\_middle\_name» для ускорения поиска.
       * триггер, изменяющий значение атрибута «passport\_is\_active» по истечению срока эксплуатации, хранимого в атрибуте «passport\_validity».
   17. Сущность «registration\_address»:
       * представление получения полного адреса.
   18. Сущность «country».
   19. Сущность «city».
   20. Сущность «street».
   21. Сущность «building».
   22. Сущность «apartment».



Финальная исправленная версия БД