Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный технический университет имени Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Теоретическая информатика и компьютерные технологии (ИУ-9)

Информационная система для сети ритейл-магазинов

Тришин А.А.

Группа ИУ9-61

Руководитель

Вишняков И. Э.

Москва, 2018г.

Оглавление

Введение 3
1. Проектирование базы данных 5
1.1. Обзор предметной области 5
1.2. Проектирование базы данных 6
1.2.1 Проектирование основной части базы данных 6
1.2.2. Проектирование базы данных в магазине 10
2. Разработка информационной системы 14
2.1 Архитектура системы 14
2.2. Межсерверное приложение 14
2.3 Приложение с графическим интерфейсом 15
3. Реализация информационной системы 16
3.1 Реализация базы данных 16
3.2 Реализация межсерверного приложения 21
3.3 Реализация приложения с графическим интерфейсом 23
4. Тестирование системы 25
Заключение 28
Список литературы. 29
ПРИЛОЖЕНИЕ А. Описание таблиц в главной базе данных 30
ПРИЛОЖЕНИЕ Б. Описание таблиц в базе данных магазина 34

Введение

Ритейл-аналитика [1] — основа эффективного управления розничными продажами. Отчеты используются в коммерческом, закупочном, операционном департаментах, департаментах маркетинга и рекламы, продаж, финансов, бухгалтерии, других подразделениях торговой компании.

Основные задачи, которые стоят перед ритейлерами – увеличение среднего чека, развитие лояльности посетителей, привлечение новых клиентов. Для ритейла, особенно если говорить о крупных сетях с большим количеством магазинов, крайне важен строгий контроль средств и рациональное управление. Сейчас в них существуют целые штаты аналитиков, которые занимаются отслеживанием десятков показателей эффективности предприятий, в которых они работают. Ведение учета в Excel или вручную неизбежно приводит к ошибкам ввода данных и большой потере времени.

Подобная ситуация повышает издержки и делает невозможным эффективное прогнозирование спроса и предложения, т.к. для него необходимы точные цифры. Другими словами, ручное ведение учета не прозрачно. Автоматизация бизнес-процессов способствует эффективному ведению предприятия и значительному снижению расходов. Автоматизация позволяет решить следующие задачи в торговле:

- учёт движения и контролирование остатков товара;
- расчёт оборачиваемости товара – показывает количество проданного товара, востребованность той или иной продукции;
- контроль деятельности сотрудников магазина, анализируя статистику продаж, фиксируя все произведённые работниками операции;
- анализ финансового результата торговой деятельности, рентабельности работы предприятия.

Автоматизация — это потребность не только крупных компаний, но и необходимость для предприятий малого формата. Благодаря автоматизации оперативного учета руководство получает достоверную картину происходящего на предприятии в режиме реального времени, что позволяет эффективно управлять ассортиментом, бороться с воровством, управлять затратами и запасами, контролировать взаиморасчеты с контрагентами. В современных условиях при высоком уровне конкуренции на рынке управлять предприятием, осуществляя все операции вручную, уже невозможно. Автоматизация бизнеса — это путь к повышению эффективности и возможность завоевать новые рынки в условиях жесткой конкуренции.

Целью данной работы является реализация подобной информационной системы с возможностью отслеживания различных показателей эффективности, их оценки, а также просмотра статистки всей сети магазинов.

1. Проектирование базы данных

1.1. Обзор предметной области

В качестве предметной области данной курсовой работы была выбрана сеть ритейл-магазинов одежды. Чтобы измерить эффективность работы магазина применяются специальные коэффициенты эффективности, или KPI[2]. Соответственно сбор статистики по магазинам осуществлялся по следующим показателям:

- коэффициент конверсии — это отношение количества посетителей магазина к количеству покупателей;
- среднее количество товаров в чеке;
- частота появления каждого идентификатора товарной позиции, или SKU (Store Keeping Unit);
- частота пар SKU;
- средний чек — это выручка за выбранный период, деленная на количество выданных за это же время чеков.
- продажи на квадратный метр — это выручка за выбранный период, деленная на площадь торгового зала в квадратных метрах.
- количество чеков;
- количество возвратов;
- количество посетителей;
- количество проданных позиций;
- выручка с учётом НДС;
- выручка без учёта НДС.

Также в ритейле присутствует система управления взаимоотношениями с клиентами, или CRM-система (Customer Relationship Management system). Одной из её компонент является программа лояльности, которая может реализовываться с помощью дисконтных карт. В связи с этим также осуществлялся сбор отдельной статистики по дисконтным картам по следующим показателям:

– количество каждого купленного товара с точностью до места покупки;
– общая стоимость покупок;
– количество чеков.

Вся статистика собиралась еженедельно, ежемесячно и ежегодно.

1.2. Проектирование базы данных

База данных состоит из двух частей. Первая (основная) предназначена для хранения статистики о магазинах и дисконтных картах, а вторая – для накопления информации внутри магазина.

1.2.1 Проектирование основной части базы данных

Основная часть базы данных необходима для аккумуляции всей полученной статистики. В ней хранится детальная информация о каждом из магазинов и дисконтных картах. Схемы в ER-модели сущностей “Shops” и “Card” и связанных с ними сущностей для статистики, используемые в данной части базы данных представлены на рисунках 1 и 2 соответственно.

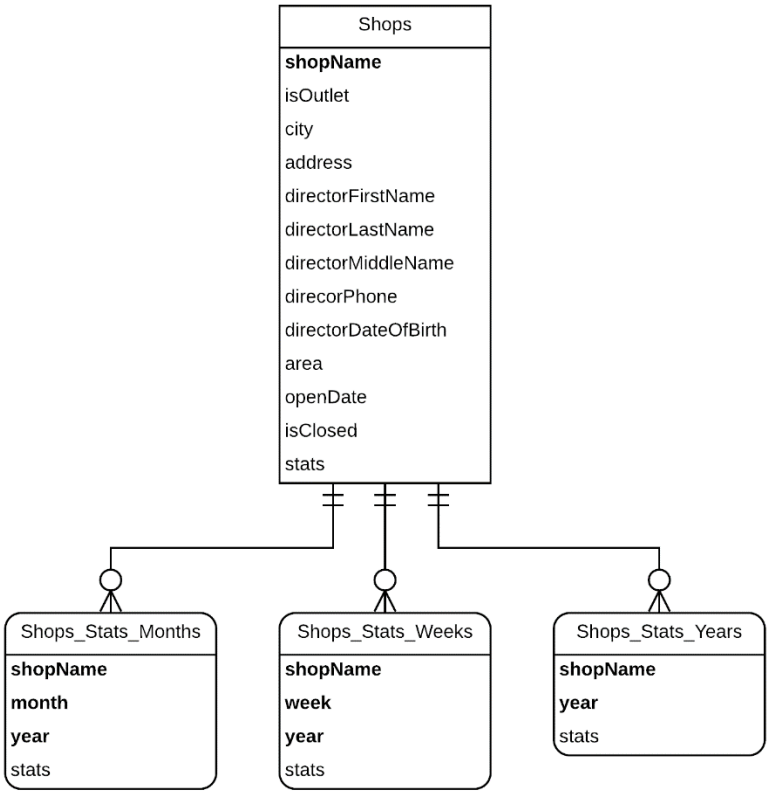

Рисунок 1. ER-модель сущности “Shops”. Идентификаторы выделены жирным шрифтом.

Сущность “Shops” содержит в себе поля:

- shopName – идентификатор, название магазина;
- isOutlet – флаг, является ли магазин аутлетом;
- city – город магазина;
- address – адрес магазина;
- directorFirstName – имя директора магазина;
- directorLastName – фамилия директора магазина;
- directorMiddleName – отчество директора магазина;
- directorPhone – номер телефона директора магазина;
- directorDateOfBirth – дата рождения директора магазина;
- area – площадь магазина;
- isClosed – флаг, закрыт ли магазин;
- openDate – дата открытия;
- stats – текущая статистика магазина;

Сущности со статистикой по магазинам (“Shops_stats_weeks”, “Shops_stats_months”, “Shops_stats_years”) включают в себя следующие поля:

- shopName – часть идентификатора – название соответствующего магазина;
- week/month/year – часть идентификатора – соответствующий период времени;
- stats – статистика за данных период времени.

Связь сущности “Shops” с “Shops_stats_weeks”, “Shops_stats_months”, “Shops_stats_years” означает наличие у первой статистики за неделю, месяц или год соответственно.

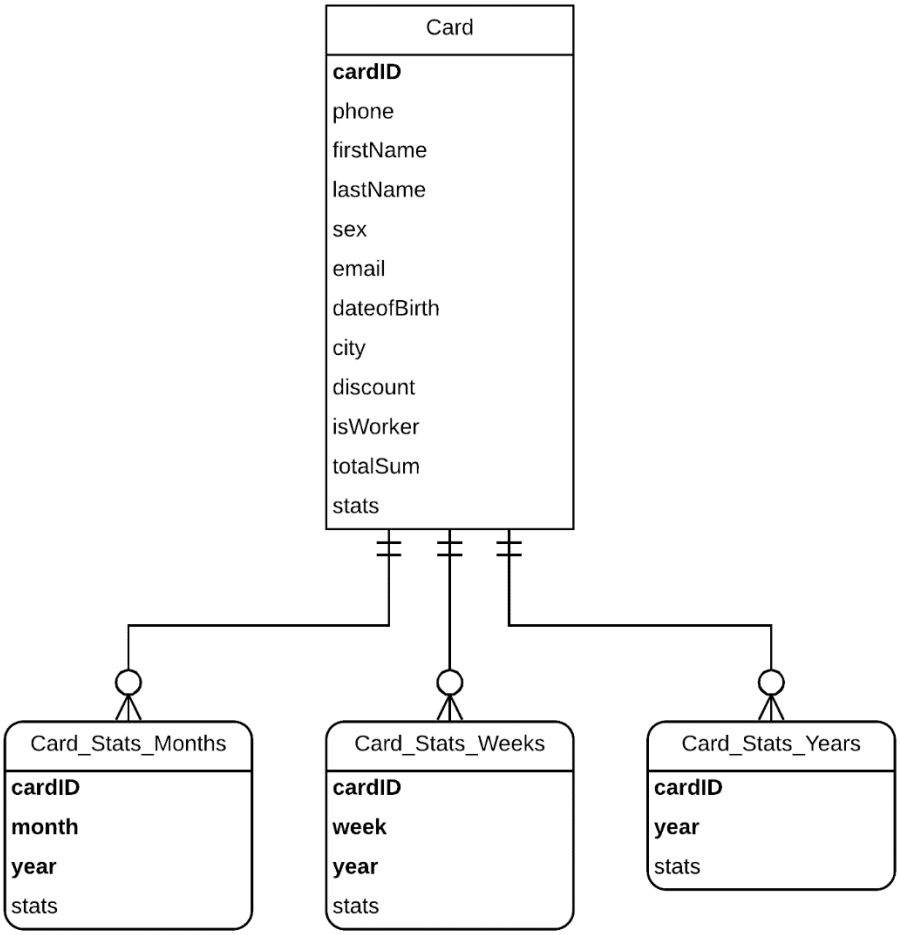

Рисунок 2. ER-модель сущности “Card”. Идентификаторы выделены жирным шрифтом.

Сущность “Card” содержит следующие поля:

- cardID – идентификатор – номер карты;
- phone – номер телефона владельца;
- firstName – имя владельца;
- lastNane – фамилия владельца;
- sex – пол владельца;
- email – адрес электронной почты владельца;
- dateOfBirth – дата рождения владельца;
- city – город проживания владельца;
- discount – размер скидки;
- isWorker – флаг, является ли владелец сотрудником компании;
- totalSum – сумма покупок;
- stats – текущая статистика.

Сущности со статистикой по картам (“Card_stats_weeks”, “Card_stats_months”, “Card_stats_years”) включают в себя следующие поля:

- cardID – часть идентификатора – номер соответствующей карты;
- week/month/year – часть идентификатора – соответствующий период времени;
- stats – статистика за данных период времени.

Аналогично “Shops” связь “Card” с “Card_stats_weeks”, “Card_stats_months”, “Card_stats_years” означает наличие у карты статистики за неделю, месяц или год соответственно.

Также необходимо хранить информацию о продаваемых товаров, для были созданы сущности “item” и “itemType”. Сущность “item” является физической реализацией сущности “itemType”. Схема в ER-модели соответствующих сущностей представлена на рисунке 3.

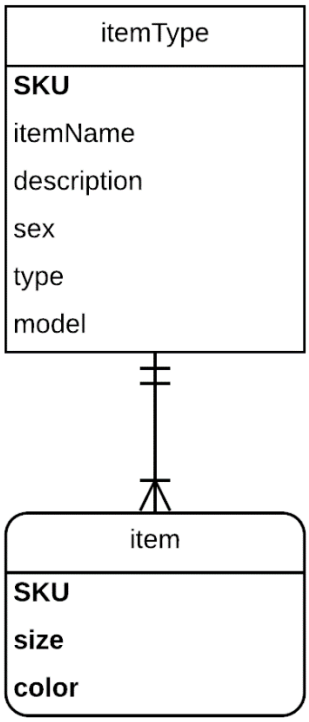

Рисунок 3. ER-модель сущностей “itemType” и “item”. Идентификаторы выделены жирным шрифтом.

Сущность “itemType” включает в себя следующие поля:

- SKU – SKU товара;
- itemName – название товара;

– description – описание товара;
– sex – пол;
– type – вид товара;
– model – модель товара.

Сущность “Item” содержит в себе:

– SKU – SKU товара;
– size – размер товара;
– color – цвет товара.

Для последующего взаимодействия с графическим приложением понадобятся следующие функции:

– получение таблицы частот пар товаров в одном чеке в указанном магазине за года/месяцы/недели в указанном массиве;
– получение частот пар товаров в указанном городе за года/месяцы/недели в указанном массиве;
– получение таблицы частот товаров в указанном магазине за года/месяцы/недели в указанном массиве;
– получение таблицы частот товаров в указанном городе за года/месяцы/недели в указанном массиве;
– получение статистики указанного города за года/месяцы/недели в указанном массиве.

1.2.2. Проектирование базы данных в магазине

Вторая часть базы данных содержит в себе информацию о самом магазине и о транзакциях, которые в нём происходят. Схема ER-модели данной части базы данных представлена на рисунке 4.

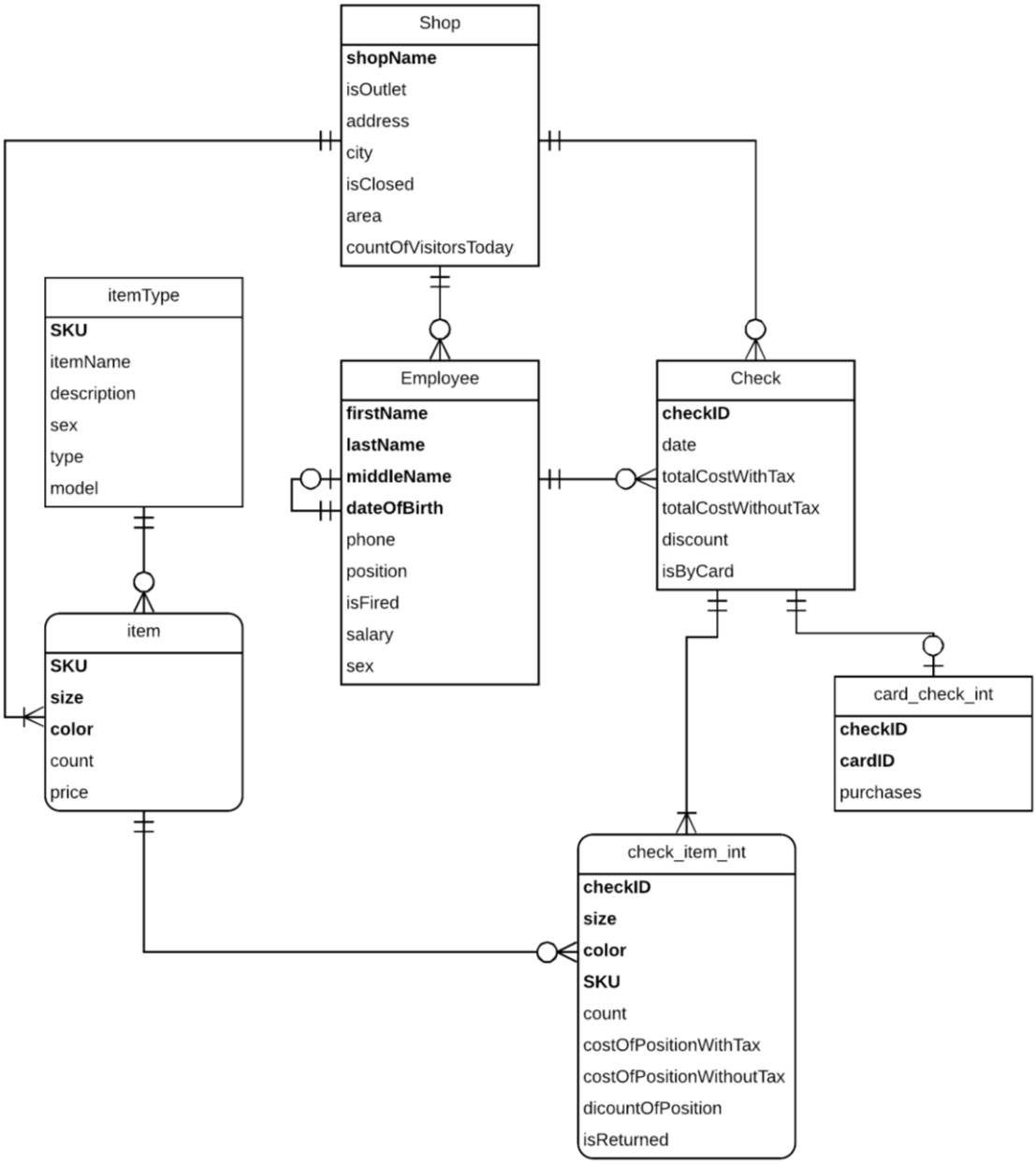

Рисунок 4. Схема ER-модели базы данных в магазине. Идентификаторы выделены жирным шрифтом.

Сущность “shop” хранит данные о магазине и содержит в себе следующие поля:

- shopName – идентификатор – название магазина;
- isOutlet – флаг, является ли магазин аутлетом;
- city – город магазина;
- address – адрес магазина;
- area – площадь магазина;
- isClosed – флаг, закрыт ли магазин;

– countOfVisitorsToday – количество посетителей за текущий рабочий день.

Сущность “employee” используется для хранения информации о сотрудниках магазина, включает в себя поля:

– firstName – часть идентификатора – имя сотрудника;
– lastName – часть идентификатора – фамилия сотрудника;
– middleName – часть идентификатора – отчество сотрудника;
– dateOfBirth – часть идентификатора – дата рождения сотрудника;
– phone – номер телефона сотрудника;
– position – должность сотрудника;
– isFired – флаг, был ли уволен сотрудник;
– salary – зарплата сотрудника;
– sex – пол сотрудника.

Сущность “check” содержит необходимую информацию для описания чеков покупок, также включает в себя поля:

– checkID – идентификатор – номер чека;
– date – дата в чеке;
– totalCostWithTax – итоговая стоимость с учётом НДС;
– totalCostWithoutTax – итоговая стоимость без учёта НДС;
– discount – размер скидки, если она была;
– isByCard – флаг, была ли покупка оплачена картой.

Сущность “check_item_int” описывает позицию товара в чеке. Содержит следующие столбцы:

– checkID – номер чека;
– size – размер товара;
– color – цвет товара;
– SKU – идентификатор товарной позиции;
– count – количество данного товара в чеке;
– costPositionWithTax – стоимость товара с учётом НДС;

– costPositionWithoutTax – стоимость товара без учёта НДС;
– discountOfPostition – размер скидки на данный товар, если она была;
– isReturned – флаг, был ли возвращён данный товар.

Сущность “card_check_int” необходима для описания товаров, купленных с дисконтной картой, включает в себя поля:

– checkID – номер чека;
– cardID – номер карты, по которой была совершена покупка;
– purchases – массив SKU купленных товаров.

Сущность “item” аналогична соответствующей сущности в основной части базы данных, кроме того, расширена полями “count” – количество данного товара на складе и “price” – цена за единицу товара.

Сущность “itemType” полностью аналогична соответствующей в основной части базы данных.

Связь “Employee” и “Shop” означает наличие работающего сотрудника в магазине. Связь “Employee” и “Employee” – руководитель сотрудника. Связь “Check” с “Employee” и “Shop” обозначает выписанный чек в магазине “Shop” сотрудником “Employee”. Связь “Card_check_int” и “Check” – SKU товаров, купленных по “Check”. Связь “Item”, “Check_item_int” и “Check” описывает товарную позицию “Item” в “Check”.

2. Разработка информационной системы

2.1 Архитектура системы

Информационная система состоит из главного сервера, который содержит в себе основную часть базы данных, серверов-клиентов в каждом из магазинов, которые содержат вторую часть базы данных, приложения, которое обеспечивает взаимодействие главного сервера и серверов в магазинах и из приложения с графическим интерфейсом, которое подключается к главному серверу. Таким образом, получается схема, представленная на рисунке 5.

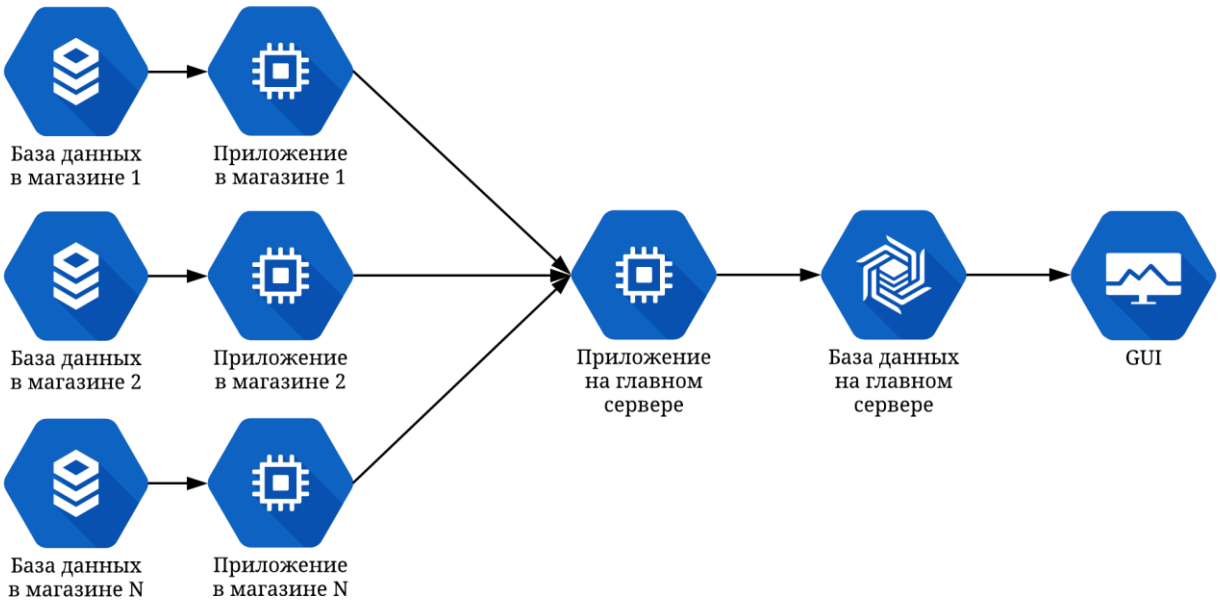

Рисунок 5. Схема информационной системы.

2.2. Межсерверное приложение

Межсерверное приложение обеспечивает сбор, анализ и отправку статистики магазина на главный сервер. Так же, как и база данных, оно разделено на две части. Первая запускается в каждом магазине, а вторая на главном сервере.

Часть приложения, которая запускается на каждом магазине собирает необходимую статистику за день в конце каждого рабочего дня из базы данных, анализирует её, а затем отправляет на главный сервер. Если сообщение не дошло или произошла ошибка, то приложение кэширует данное

сообщение и будет пытаться отправлять его до тех пор, пока не придёт ответ об успешном получении сообщения. После успешной отправки кэшированные сообщения удаляются из кэша. Каждое сообщение имеет свой идентификатор, который составляется из кода магазина (откуда отправляется сообщение) и порядкового номера самого сообщения. В конце каждого отслеживаемого периода времени приложение отправляет сообщение о том, что все данные за этот период были высланы, таким образом, сообщив части приложения на главном сервере о возможности сохранения данных за данный период времени данного магазина.

Часть приложения, запускаемая на главном сервере, является средством запуска самого сервера, а также аггрегатором статистики, которая приходит от всех магазинов. Когда приходит сообщение с данными за день от какого-либо магазина данная часть приложения добавляет к эти данные к остальным данным этого же магазина в базе данных. Если приходит контрольное сообщение о том, что вся информация за один из отслеживаемых периодов времени выслана, то она сохраняется в таблицу, отвечающую за данный период времени.

2.3 Приложение с графическим интерфейсом

Приложение с графическим интерфейсом необходимо для просмотра статистики и для её сравнения за различные периоды времени. Вся необходимая логика получения данных из базы данных выполняется в самой базе данных. То есть данное приложение является только способом представления хранящихся данных в базе данных в удобной форме: в виде таблиц или диаграмм.

В нём также возможно вывести статистику о магазинах или о дисконтных картах за выбранные пользователем доступные временные интервалы в виде таблиц или диаграмм, а также сохранить эти таблицы или диаграммы в файлы.

3. Реализация информационной системы

3.1 Реализация базы данных

Для реализации базы данных была выбрана свободная объектно-реляционная система управления базами данных PostgreSQL [3], так как она поддерживает такие типы данных как JSONB, который поддерживает индексирование, быстрее обрабатывается и требует меньше места для объектов (в сравнении с обычным JSON в том же PostgreSQL), и массивы. Также PostgreSQL позволяет создавать собственные типы данных, что является очень удобным средством и непосредственно использовалось в реализации (были созданы типы данных: пол – sex, модель товара – model, тип товара – item_type и цвет – color).

Было реализовано две части базы данных – для магазина: “shopDB” и для главного сервера: “mainDB”, в каждой из них схема – “shopschema”. Описание таблиц в базе “mainDB” и в “shopDB” представлены в приложении А и Б соответственно. На рисунке 6 и 7 представлены схемы обеих частей в реляционной модели.

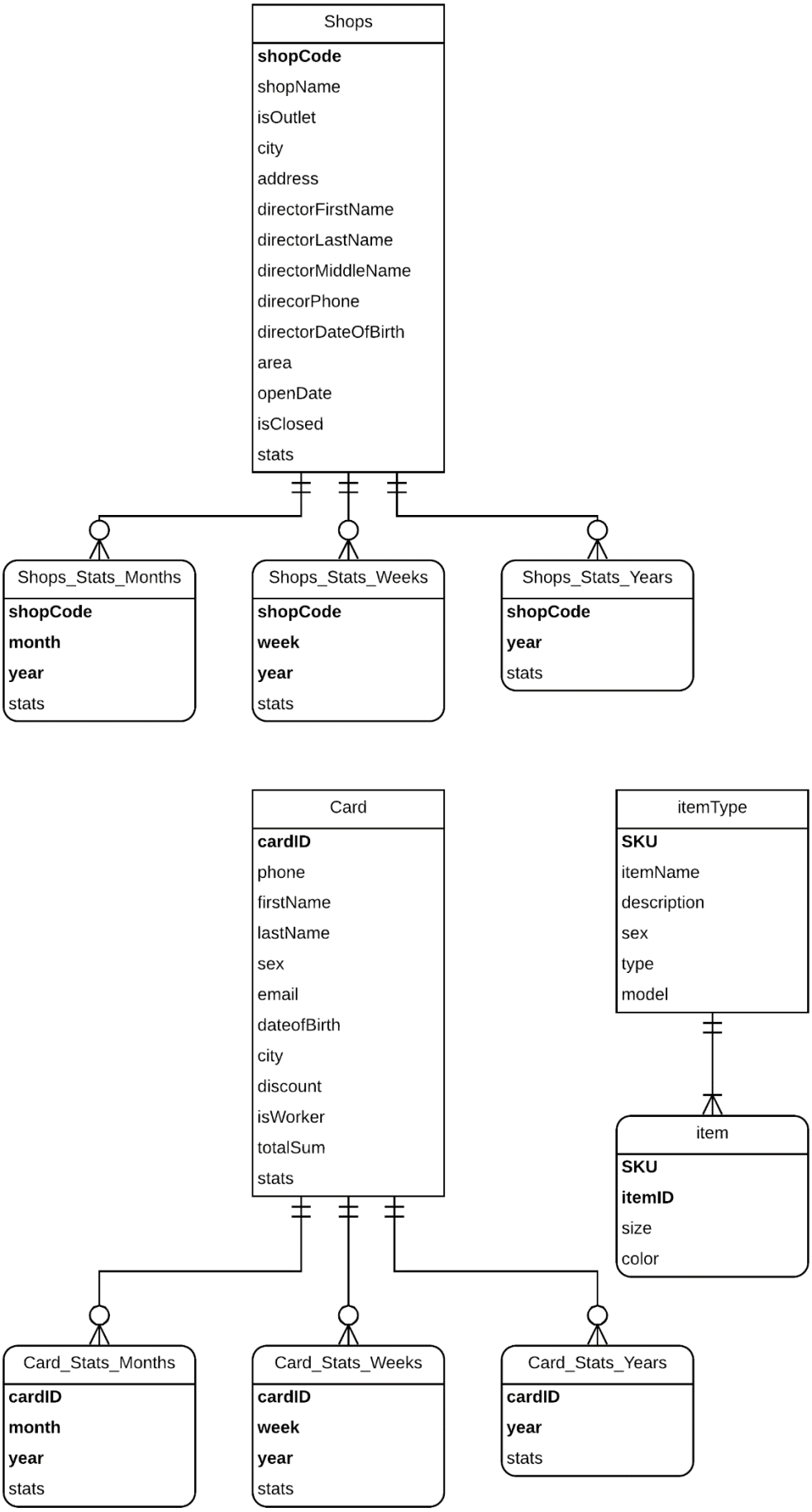

Рисунок 6. Реляционная модель основной части базы данных.

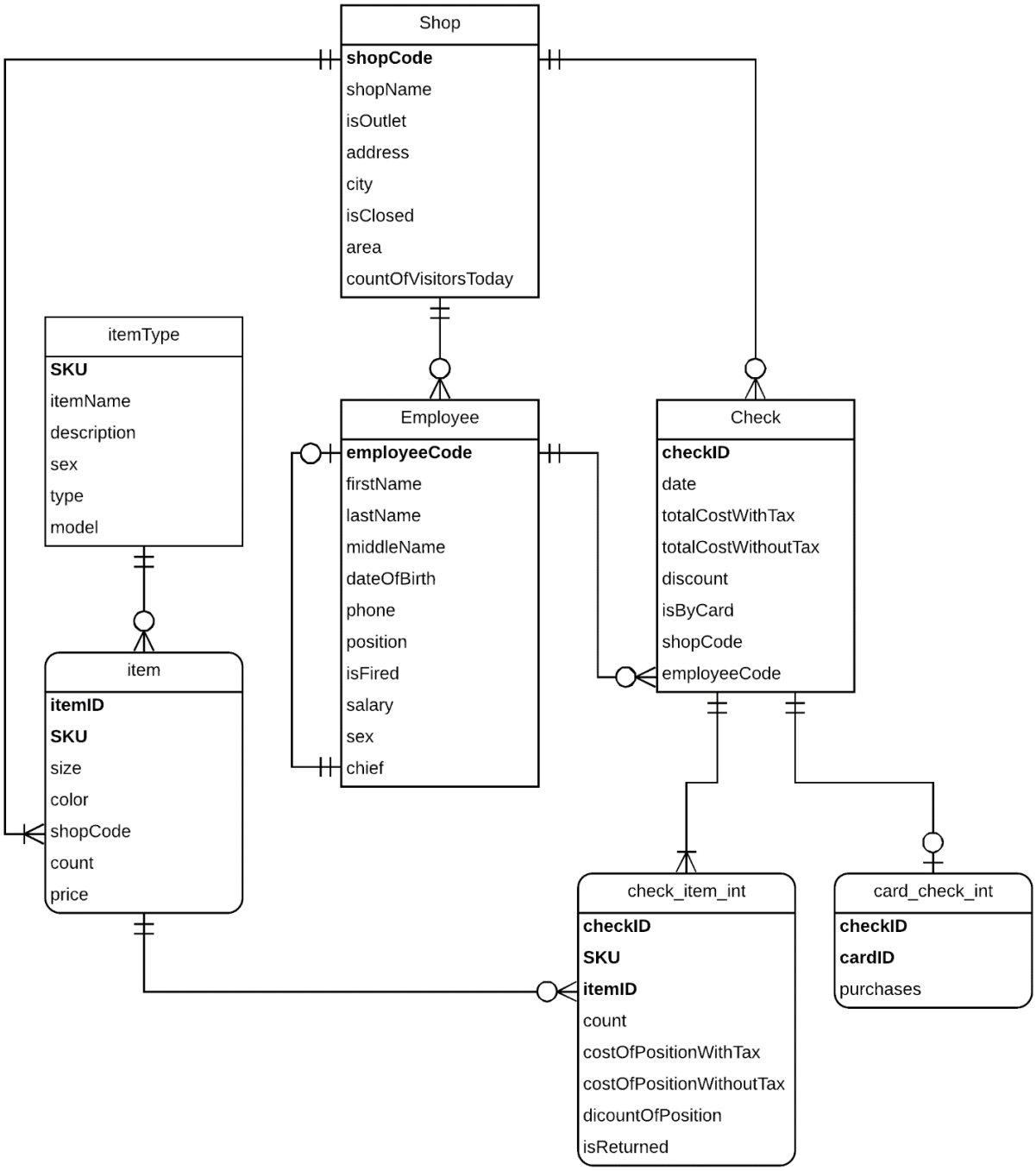

Рисунок 7. Реляционная модель базы данных магазина.

Также в базе “mainDB” было реализовано API, которое позволяет приложению с графическим интерфейсом получать запрашиваемые пользователем данные:

– FUNCTION MainDB.shopschema.get_sku_pairs_frequency_year(shopcode_p INT, years INT[])
RETURNS TABLE(item1 VARCHAR, item2 VARCHAR, sex VARCHAR, count BIGINT);

- FUNCTION MainDB.shopschema.get_common_sku_pairs_frequency_year(years INT[], city INT DEFAULT 0) RETURNS TABLE(item1 VARCHAR, item2 VARCHAR, sex VARCHAR, count BIGINT);
- FUNCTION MainDB.shopschema.get_sku_pairs_frequency_month(shopcode_p INT, year_p INT, months INT[])
  RETURNS TABLE(item1 VARCHAR, item2 VARCHAR, sex VARCHAR, count BIGINT);
- FUNCTION MainDB.shopschema.get_common_sku_pairs_frequency_month(year_p INT, months INT[], city INT DEFAULT 0)
  RETURNS TABLE(item1 VARCHAR, item2 VARCHAR, sex VARCHAR, count BIGINT);
- FUNCTION MainDB.shopschema.get_sku_pairs_frequency_week(shopcode_p INT, year_p INT, weeks INT[])
  RETURNS TABLE(item1 VARCHAR, item2 VARCHAR, sex VARCHAR, count BIGINT);
- FUNCTION MainDB.shopschema.get_common_sku_pairs_frequency_week(year_p INT, weeks INT[], city INT DEFAULT 0) RETURNS TABLE(item1 VARCHAR, item2 VARCHAR, sex VARCHAR, count BIGINT);
- FUNCTION MainDB.shopschema.get_sku_frequency_year (shopcode_p INT, years INT[])
  RETURNS TABLE (sex VARCHAR, item VARCHAR, count BIGINT);
- FUNCTION MainDB.shopschema.get_common_sku_frequency_year(years INT[], city INT DEFAULT 0)
  RETURNS TABLE (sex VARCHAR, item VARCHAR, count BIGINT);

– FUNCTION MainDB.shopschema.get_sku_frequency_month(shopcode_p INT, year_p INT, months INT[])
  RETURNS TABLE (sex VARCHAR, item VARCHAR, count BIGINT);
– FUNCTION MainDB.shopschema.get_common_sku_frequency_month(year_p INT, months INT[], city INT DEFAULT 0)
  RETURNS TABLE (sex VARCHAR, item VARCHAR, count BIGINT);
– FUNCTION MainDB.shopschema.get_sku_frequency_week(shopcode_p INT, year_p INT, weeks INT[])
  RETURNS TABLE (sex VARCHAR, item VARCHAR, count BIGINT);
– FUNCTION MainDB.shopschema.get_common_sku_frequency_week(year_p INT, weeks INT[], city INT DEFAULT 0)
  RETURNS TABLE (sex VARCHAR, item VARCHAR, count BIGINT);
– FUNCTION MainDB.shopschema.get_city_shop_stats_year(years INT[], city_p INT DEFAULT 0)
  RETURNS TABLE (CR FLOAT, UPT FLOAT, avgCheck FLOAT, salesPerArea FLOAT, countOfChecks BIGINT, returnedUnits BIGINT, countOfVisitors BIGINT, proceedsWithTax FLOAT, proceedsWithoutTax FLOAT, countOfSoldUnits BIGINT);
– FUNCTION MainDB.shopschema.get_city_shop_stats_month(year_p INT, months INT[], city_p INT DEFAULT 0)
  RETURNS TABLE (CR FLOAT, UPT FLOAT, avgCheck FLOAT, salesPerArea FLOAT, countOfChecks BIGINT, returnedUnits BIGINT, countOfVisitors BIGINT, proceedsWithTax FLOAT, proceedsWithoutTax FLOAT, countOfSoldUnits BIGINT);
– FUNCTION MainDB.shopschema.get_city_shop_stats_week(year_p INT, weeks INT[], city_p INT DEFAULT 0)
  RETURNS TABLE (CR FLOAT, UPT FLOAT, avgCheck FLOAT, salesPerArea FLOAT, countOfChecks BIGINT, returnedUnits BIGINT,

countOfVisitors BIGINT, proceedsWithTax FLOAT, proceedsWithoutTax FLOAT, countOfSoldUnits BIGINT);

В базе данных “shopDB” были созданы два материализованных представления: “items” и “cards_purchases”, которые в себе объединяют необходимые поля для полного описания чека и покупок по картам соответственно. Так как все запросы из приложения осуществляют выборку из “items” и из “cards_purchases” по столбцу “date”, то он был проиндексирован в обоих представлениях. Аналогичное поле проиндексировано в таблице “check”.

3.2 Реализация межсерверного приложения

В качестве языка программирования для реализации межсерверного приложения был выбран язык Scala [4], так как он очень удобен для работы в связке с фреймворками AKKA [5] HTTP и AKKA Actors, которые и использовались в данной работе. Доступ к базе данных осуществляется с помощью JDBC, также чтобы сделать планировщик (для отправки сообщений по расписанию) использовался AKKA Quartz Scheduler.

На главном сервере поднимается сам REST сервер с помощью AKKA HTTP, а также акторы serverActor (занимается обработкой статистики и добавлением её к текущей) и dumpActor (занимается сохранением статистики за определённый период времени). Таким образом, реализовано следующее REST API:

- post http://{host}/shopstats?msgid={id}&shopcode={code}; в теле запроса должен лежать json со статистикой магазина – данный запрос добавляет к текущей статистики новую из тела запроса к магазину с кодом “code”.
  Возвращает код: 200 OK, в случае успешного выполнения.
- post http://{host}/cardstats?msgid={id}&shopcode={code}; в теле запроса должен лежать json со статистикой по картам из магазина – данный запрос добавляет к текущей статистике новую из тела запросу к статистке по картам.
  Возвращает код: 200 OK, в случае успешного выполнения.

– post

  http://{host}/cntrlWeek?msgid={id}&WEEK={week}&YEAR={year}& shopcode={shopcode};

  http://{host}/cntrlMonth?msgid={id}&MONTH={month}&YEAR={year} &shopcode={shopcode};

  http://{host}/cntrlYear?msgid={id}YEAR={year}& shopcode={shopcode}.

  В запросе должен лежать массив из списка номеров кард, которые использовались за период указанный в параметрах “week”, “month” или “year” – данный запрос сообщает о том, что вся статистика от магазина с кодом “shopcode” за определённый период времени была выслана и её необходимо сохранить в базу данных.

  Возвращает код: 200 OK, в случае успешного выполнения.

В API параметр “msgID” – параметр, который генерируется автоматически на стороне клиента, он необходим для идентификации сообщений, чтобы отфильтровывать случайные повторные сообщения и не реагировать на них (если сообщение с определённой связкой “msgid” и “shopcode” не приходило, то оно добавляется в таблицу сообщений, иначе сервер ничего не делает с этим сообщением). Вся статистика для удобства использования сериализуется в объект соответствующего класса (в ShopStats или в CardStats).

Каждая часть приложения, запускаемая в магазине, является “клиентом” HTTP-сервера, и вызывает необходимый из его доступных API. В этой части приложения создаётся актор ShopActor и планировщик Quartz Scheduler, который в указанное в Cron Expression время отсылает ShopActor’у сообщение о том, что необходимо собрать статистику за текущее время и отослать её на главный сервер. Также, если данная отправка сообщения – последняя за отслеживаемый период времени, то после сообщения со статистикой на главный сервер отсылается и контрольное сообщение с текущим периодом времени. Все сообщения, ответ на которые был не 200 OK, кэшируются, затем создаётся еще один планировщик Quartz Scheduler, который заставляет эти

сообщения отправляться еще раз до тех пор, пока не будет получен код об их успешной отправке. Если в кэше ничего не остается, то данный планировщик отключается.

3.3 Реализация приложения с графическим интерфейсом

В ходе данной курсовой работы также было реализовано приложение с графическим интерфейсом. В качестве языка реализации был выбран Python [6] с библиотекой PyQt [7], которая и позволяет создавать графический интерфейс. В нём находятся две вкладки: «Магазины» и «Карты», внутри них можно посмотреть статистику по магазинам и картам соответственно (см. рисунок 8). Также во вкладке «Магазины» можно посмотреть статистику магазинов по городам.

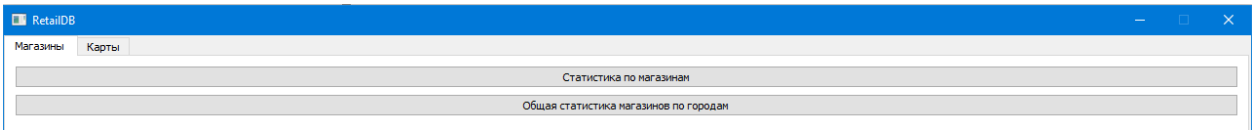

Рисунок 8. Фрагмент окна с выбором просматриваемой статистики.

После того, как пользователь выбирает какую статистику он хочет увидеть, он переходит в следующее окно с выбором времени, за которое необходимо отобразить статистику и магазина или города (см. рисунок 9).

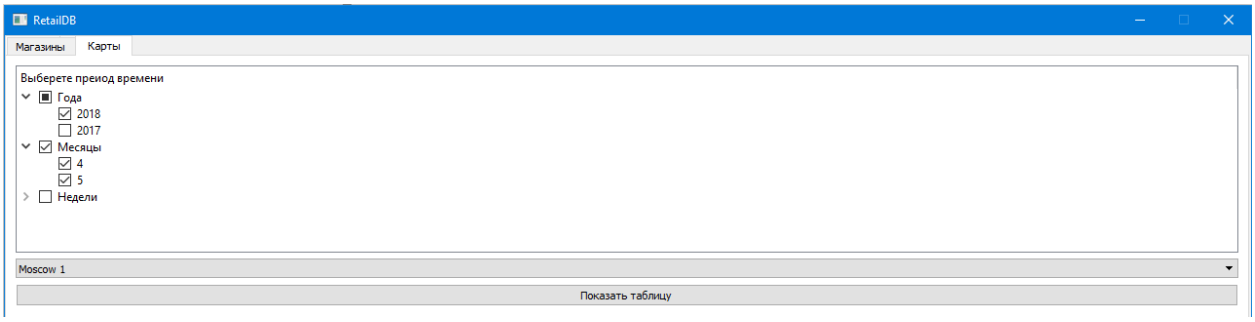

Рисунок 9. Фрагмент окна выбора магазина и дат. Выбран просмотр апреля и мая 2018 года.

Затем пользователь должен нажать на кнопку «Показать таблицу» и тогда он перейдет в следующее окно для просмотра самой статистики в виде таблицы (см. рисунок 10).

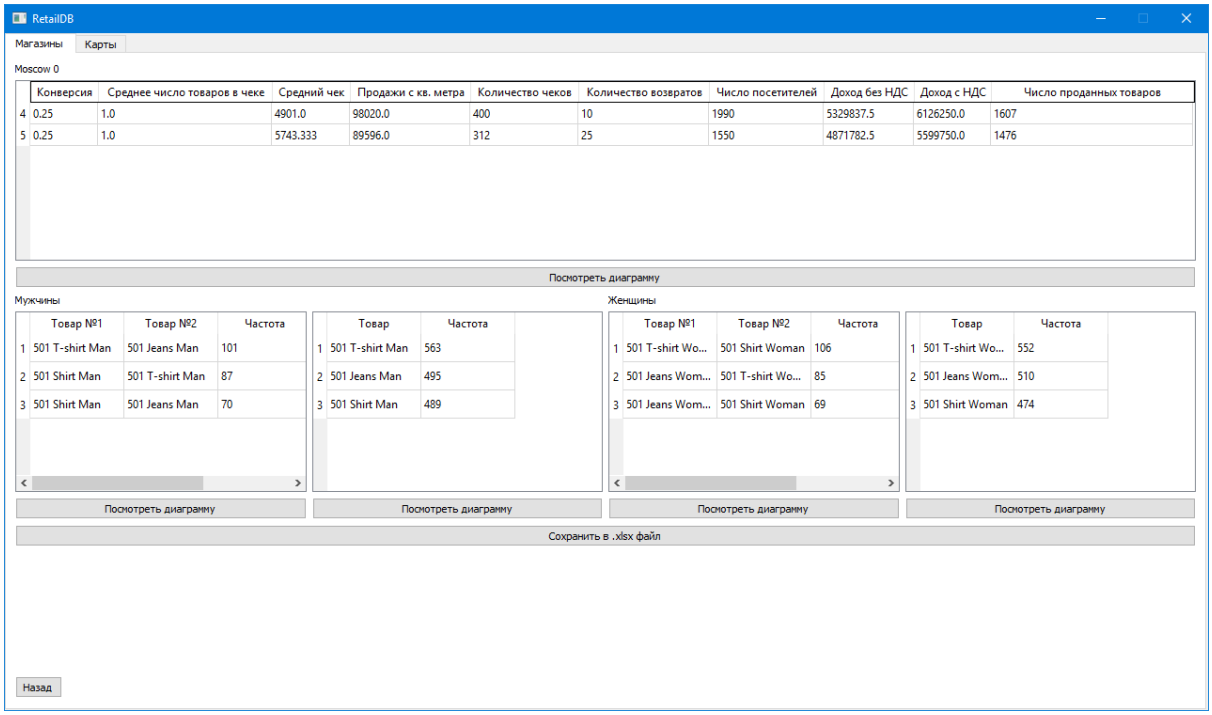

Рисунок 10. Экран просмотра статистики.

На данном экране пользователь может сохранить статистику в .xlsx файл, а также посмотреть её в виде диаграммы как показано на рисунке 11. Любую диаграмму можно сохранить .png файл.

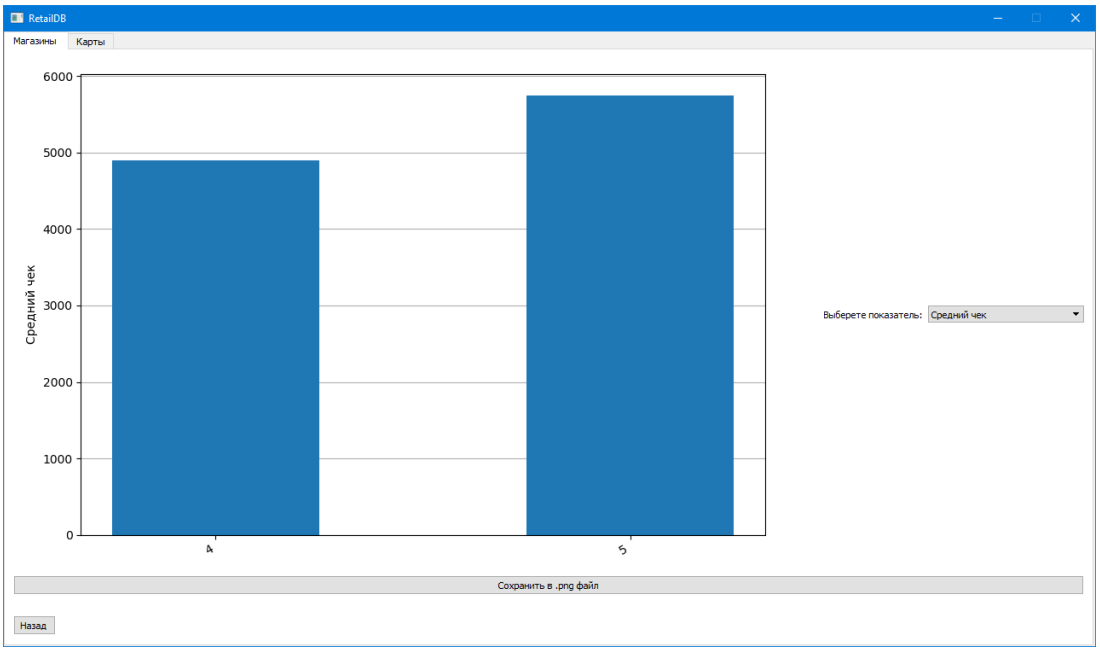

Рисунок 11. Экран просмотра диаграмм. Выбран показатель «средний чек».

4. Тестирование системы

Тестирование системы производилось путём замера времени выполнения запросов, используемых в базе данных, а также HTTP-запроса, получаемого от магазина. В среднем в ритейле за день совершается от 15 от 20 покупок, поэтому для тестирования будем считать, что в день осуществлялось 40 продаж, чтобы специально нагрузить систему. Тестирование осуществлялось на системе с процессором AMD FX-6300 3.5 GHz с 8 Gb ОЗУ.

Проводилось сравнение времени выполнения запросов в базе данных при хранении записей о покупках за 30, 90, 360, 3600 и 36500 дней. Для запросов, связанных с дисконтными картами, считалось, что каждая пятая покупка была с картой. Были протестированы используемые запросы:

1. SELECT count(checkid) as count FROM "Check" WHERE date::date = $1.
2. SELECT SUM(costofposition)::NUMERIC::FLOAT FROM items WHERE date::date = $1 AND NOT isreturned.
3. SELECT DISTINCT i1.checkid, i1.sku, i2.sku FROM items i1 JOIN items i2 ON i1.checkid = i2.checkid AND i1.sku > i2.sku WHERE i1.date::date = $1.
4. SELECT sku, COUNT(sku) FROM items WHERE date::date = $1 GROUP BY sku.
5. SELECT DISTINCT cardid FROM cards_purchases WHERE date::date BETWEEN $1 AND $2.
6. SELECT cardid, totalcostwithtax::NUMERIC::FLOAT, array_agg(sku) FROM cards_purchases GROUP BY cardid, date, totalcostwithtax HAVING date::date = $1;

Результаты тестирования представлены в таблице 1.

Таблица 1 – результаты тестирования запросов, используемых в работе.

Номер запроса	Кол-во записей	Кол-во дней	Ср. результат, мс
1	1200	30	0,2
	3600	90	0,7

Номер запроса	Кол-во записей	Кол-во дней	Ср. результат, мс
1	14400	360	2,2
	144000	3600	24,2
	1460000	36500	182,3
2	1200	30	0,3
	3600	90	1,0
	14400	360	4,1
	144000	3600	38,1
	1460000	36500	271,6
3	1200	30	0,4
	3600	90	1,9
	14400	360	6,7
	144000	3600	67,6
	1460000	36500	293,2
4	1200	30	0,3
	3600	90	1,0
	14400	360	4,0
	144000	3600	37,1
	1460000	36500	293,4
5	240	30	0,3
	720	90	0,7
	2880	360	2,1
	28800	3600	20,5
	292000	36500	166,6
6	240	30	0,1
	720	90	0,1
	2880	360	0,4
	28800	3600	3,5
	292000	36500	33,3

Для тестирования сервера была произведена эмуляция параллельной отправки сообщений каждого вида HTTP-запроса с нескольких магазинов:

1. HTTP-запрос со статистикой по магазину.
2. HTTP-запрос со статистикой по картам.
3. HTTP-запрос с контрольным сообщением.

Получились результаты, представленные в таблице 2.

Таблица 2 – результаты тестирования выполнения HTTP-запросов.

Номер запроса	Кол-во магазинов	Ср. результат
1	25	1.551
	100	3.119
	500	12.358
	5000	124.473
	10000	242.654
2	25	1.545
	100	2.905
	500	16.181
	5000	161.412
	10000	307.370
3	25	1.729
	100	6.188
	500	24.302
	5000	257.133
	10000	511.576

В ходе тестирования было выявлено, что информационная система способна работать не менее ста лет без значительной потери производительности. Также видно, что наибольшее количество времени уходит на выполнение HTTP-запросов, причём наиболее значительная часть времени тратится на сохранение статистики за определенный период времени (HTTP-запрос с контрольным сообщением), при 10000 магазинах на вставку новой статистики приходится около 8.5 минут.

Заключение

В ходе данной курсовой работы была реализована информационная система для сети ритейл-магазинов. Она позволяет частично решить задачу автоматизации сбора, накопления и анализа статистики такой сети. При добавлении новых показателей не надо перестраивать всю базу данных.

В будущем хранимую статистику для дисконтных карт можно использовать для предложения её владельцу контекстной рекламы, а также новой информации о его любимых товаров. Также можно собирать больший объём данных, необходимый компании для более полного анализа покупок своих клиентов или для разработки и внедрения системы управления взаимоотношениями с клиентами.

Список литературы.

[1] Ритейл-аналитика. https://www.sisense.com/glossary/retail-analytics/ – дата обращения 27.05.2018;

[2] KPI. https://www.business.ru/article/835-klyuchevye-pokazateli-effektivnosti-roznichnogo-magazina-kpi – дата обращения 27.05.2018;

[3] Документация PostgreSQL. https://www.postgresql.org/docs/ – дата обращения 27.05.2018;

[4] Документация языка Scala. https://docs.scala-lang.org – дата обращения 27.05.2018;

[5] Документация фрэймворка AKKA. https://akka.io/docs/ – дата обращения 27.05.2018;

[6] Документация языка Python. https://www.python.org/doc/ – дата обращения 27.05.2018;

[7] Документация библиотеки PyQt5. https://pyqt.sourceforge.net/Docs/PyQt5/ – дата обращения 27.05.2018;

ПРИЛОЖЕНИЕ А. Описание таблиц в главной базе данных

Описание каждой из таблиц в главной базе данных представлено в таблицах А.1-А.10.

Таблица А.1 – описание таблицы “shops”.

Имя столбца	Тип данных	Ключ	NULL	Описание
shopCode	INTEGER	PK	NOT NULL	Суррогатный ключ, состоит из кода города и номера магазина
shopName	VARCHAR	AK	NOT NULL	UNIQUE
isOutlet	BOOLEAN	-	NOT NULL	DEFAULT FALSE
city	VARCHAR	-	NOT NULL	-
address	VARCHAR	-	NOT NULL	-
directorFirstName	VARCHAR	-	NOT NULL	-
directorLastName	VARCHAR	-	NOT NULL	-
directorMiddleName	VARCHAR	-	NOT NULL	-
directorPhone	CHAR	-	NOT NULL	-
directorDateOfBirth	DATE	-	NOT NULL	-
area	REAL	-	NOT NULL	-
isClosed	BOOLEAN	-	NOT NULL	-
openDate	DATE	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.2 – описание таблицы “shops_stats_weeks”.

Имя столбца	Тип данных	Ключ	NULL	Описание
shopCode	INTEGER	PK, FK	NOT NULL	-
year	INTEGER	-	NOT NULL	-
week	INTEGER	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.3 – описание таблицы “shops_stats_months”.

Имя столбца	Тип данных	Ключ	NULL	Описание
shopCode	INTEGER	PK, FK	NOT NULL	-
year	INTEGER	-	NOT NULL	-
month	INTEGER	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.4 – описание таблицы “shops_stats_years”.

Имя столбца	Тип данных	Ключ	NULL	Описание
shopCode	INTEGER	PK, FK	NOT NULL	-
year	INTEGER	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.5 – описание таблицы “card”.

Имя столбца	Тип данных	Ключ	NULL	Описание
cardID	INTEGER	PK	NOT NULL	DEFAULT NEXTVAL(shopschema.shop_codes)
phone	CHAR	AK	NULL	UNIQUE
firstName	VARCHAR	-	NULL	-
lastName	VARCHAR	-	NULL	-
discount	SMALLINT	-	NOT NULL	DEFAULT 5
sex	SEX	-	NULL	-
email	VARCHAR	-	NULL	-
isWorker	BOOLEAN	-	NOT NULL	-
dateOfBirth	DATE	-	NULL	-
city	VARCHAR	-	NULL	-

Имя столбца	Тип данных	Ключ	NULL	Описание
totalSum	MONEY	-	NULL	-
stats	JSONB	-	NOT NULL	DEFAULT ‘{}’

Таблица А.6 – описание таблицы “card_stats_weeks”.

Имя столбца	Тип данных	Ключ	NULL	Описание
cardID	INTEGER	PK, FK	NOT NULL	-
year	INTEGER	-	NOT NULL	-
week	INTEGER	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.7 – описание таблицы “card_stats_months”.

Имя столбца	Тип данных	Ключ	NULL	Описание
cardID	INTEGER	PK, FK	NOT NULL	-
year	INTEGER	-	NOT NULL	-
month	INTEGER	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.8 – описание таблица “card_stats_years”.

Имя столбца	Тип данных	Ключ	NULL	Описание
cardID	INTEGER	PK, FK	NOT NULL	-
year	INTEGER	-	NOT NULL	-
stats	JSONB	-	NOT NULL	-

Таблица А.9 – описание таблицы “itemType”.

Имя столбца	Тип данных	Ключ	NULL	Описание
sku	INTEGER	PK	NOT NULL	Суррогатный ключ, состоит из кода пола, типа и модели товара
itemName	VARCHAR	AK	NOT NULL	-
description	VARCHAR	-	NOT NULL	-
sex	SEX	-	NOT NULL	-
type	TYPE	-	NOT NULL	-
model	MODEL	-	NOT NULL	-

Таблица А.10 – описание таблицы “item”.

Имя столбца	Тип данных	Ключ	NULL	Описание
itemID	INTEGER	PK	NOT NULL	Суррогатный ключ, состоит из кода размера и цвета товара
sku	INTEGER	PK, FK	NOT NULL	-
size	SMALLINT	-	NOT NULL	-
color	COLOR	-	NOT NULL	-

ПРИЛОЖЕНИЕ Б. Описание таблиц в базе данных магазина

Описание каждой из таблиц в базе данных магазина представлено в таблицах Б.1-Б.7

Таблица Б.1 – описание таблицы “shop”.

Имя столбца	Тип данных	Ключ	NULL	Описание
shopCode	INTEGER	PK	NOT NULL	Суррогатный ключ, состоит из кода города и номера магазина
shopName	VARCHAR	AK	NOT NULL	UNIQUE
isOutlet	BOOLEAN	-	NOT NULL	DEFAULT FALSE
city	VARCHAR	-	NOT NULL	-
address	VARCHAR	-	NOT NULL	-
isClosed	BOOLEAN	-	NOT NULL	DEFAULT FALSE
area	REAL	-	NOT NULL	-
countOfVisitorsToday	INTEGER	-	NOT NULL	-

Таблица Б.2 – описание таблицы “employee”.

Имя столбца	Тип данных	Ключ	NULL	Описание
employeeCode	INTEGER	PK	NOT NULL	DEFAULT nextval(‘emp_codes’)
firstName	VARCHAR	-	NOT NULL	-
lastName	VARCHAR	-	NOT NULL	-
middleName	VARCHAR	-	NOT NULL	-
dateOfBirth	VARCHAR	-	NOT NULL	-
phone	CHAR	AK	NOT NULL	UNIQUE
position	VARCHAR	-	NOT NULL	-
isFired	BOOLEAN	-	NOT NULL	DEFAULT FALSE
salary	MONEY	-	NOT NULL	-
sex	SEX	-	NOT NULL	-
chief	INTEGER	FK1	NOT NULL	-
shopCode	INTEGER	FK2	NOT NULL	-

Таблица Б.3 – описание таблицы “itemType”.

Имя столбца	Тип данных	Ключ	NULL	Описание
description	VARCHAR	-	NOT NULL	-
sex	SEX	-	NOT NULL	-
type	TYPE	-	NOT NULL	-
model	MODEL	-	NOT NULL	-

Таблица Б.4 – описание таблицы “item”.

Имя столбца	Тип данных	Ключ	NULL	Описание
itemID	INTEGER	PK	NOT NULL	Суррогатный ключ, состоит из кода размера и цвета товара
sku	INTEGER	PK, FK	NOT NULL	-
size	SMALLINT	-	NOT NULL	-
color	COLOR	-	NOT NULL	-
count	INTEGER	-	NOT NULL	-
price	MONEY	-	NOT NULL	-

Таблица Б.5 – описание таблицы “check”.

Имя столбца	Тип данных	Ключ	NULL	Описание
checkID	INTEGER	PK	NOT NULL	DEFAULT nextval(‘check_codes’)
date	TIMESTAMP	AK	NOT NULL	UNIQUE
totalCostWithTax	MONEY	-	NOT NULL	-
totalCostWithoutTax	MONEY	-	NOT NULL	-
discount	SMALLINT	-	NOT NULL	-
employeeCode	INTEGER	FK	NOT NULL	-
isByCard	BOOLEAN	-	NOT NULL	DEFAULT FALSE

Таблица Б.6 – описание таблицы “card_check_int”.

Имя столбца	Тип данных	Ключ	NULL	Описание
checkID	INTEGER	PK1.1, FK1	NOT NULL	-
cardID	INTEGER	PK1.2, FK2	NOT NULL	-
purchases	INTEGER[]	-	NOT NULL	-

Таблица Б.7 – описание таблицы “check_item_int”.

Имя столбца	Тип данных	Ключ	NULL	Описание
checkID	INTEGER	PK1.1, FK1	NOT NULL	-
itemID	INTEGER	PK1.2, FK2	NOT NULL	-
sku	INTEGER	PK1.3, FK3	NOT NULL	-
count	INTEGER	-	NOT NULL	-
costOfPositionWithTax	MONEY	-	NOT NULL	-
costOfPositionWithoutTax	MONEY	-	NOT NULL	-
disountOfPosition	SMALLINT	-	NOT NULL	-
isReturned	BOOLEAN	-	NOT NULL	-