МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Энергетический Факультет
Кафедра информатики вычислительной техники, прикладной математики

КУРСОВОЙ ПРОЕКТ

По: Базы данных

На тему: Разработка базы банных по учету деятельности ломбарда

Выполнил студент группы ИВТ-17-1 Демидов Владимир Юрьевич

Руководитель работы: старший преподаватель, Гончаров Денис Сергеевич

Чита
2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Энергетический Факультет
Кафедра информатики вычислительной техники, прикладной математики

ЗАДАНИЕ
на курсовой проект

По дисциплине Базы Данных

Студенту Демидову Владимиру Юрьевичу

Специальности: Информатика, вычислительная техника и прикладная математика

1. Тема курсового проекта: Разработка базы банных по учету деятельности ломбарда
2. Срок подачи студентом законченной работы: 28 декабря 2019 г.
3. Исходные данные к работе: литературные источники, источники в сети интернет
4. Перечень подлежащих разработке в курсовом проекте вопросов:
    а) разбор теоретической части;
    б) разработка базы банных.

5 Перечень графического материала:
    а) рисунки;
    б) диаграммы;
    в) результаты вывода программ;
    г) скриншоты сайта JewelBling.

Дата выдачи задания «13» сентября 2019 г.

Руководитель курсового проекта _____ /Гончаров Денис Сергеевич/

Задание принял к исполнению

«___» _____ 20__ г.

Подпись студента _____ /Демидов Владимир Юрьевич/

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Энергетический Факультет
Кафедра информатики вычислительной техники, прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

По: Базы данных

На тему: Разработка базы банных по учету деятельности ломбарда

Выполнил студент группы ИВТ-17-1 Демидов Владимир Юрьевич

Руководитель работы: старший преподаватель, Гончаров Денис Сергеевич

РЕФЕРАТ

Пояснительная записка – 41 страниц, 19 – рисунков, 3 – таблиц, 0 – схем, 0 – диаграмм, 0 – приложений.

БАЗЫ ДАННЫХ, СУБД, POSTGRESQL, SQL, NOSQL, PHP, ТРИГГЕРЫ, НОРМАЛЬНЫЕ ФОРМЫ, ОПЕРАТОРЫ, ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ, ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ, ПРЕДМЕТНАЯ ОБЛАСТЬ, PGMODELER, PGADMIN.

В работе описывается создание, улучшение и администрирование базы данных для ломбарда с помощью свободной объектно-реляционной системы управления базами данных (СУБД) Postgresql. Рассматриваются лабораторные работы по базам данных. Затрагиваются важные темы: проектирование базы данных, системный анализ предметной области, построение нормальных форм, нормализация отношений, ограничения целостности, операторы SQL, триггеры, взаимодействие базы данных с графическим интерфейсом (PHP сайт JewelBling). В работе также содержится теоретическая часть, описывающая общую информацию о базах данных, и конкретно про Postgresql.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Базы данных хранят, организовывают и обрабатывают информацию в таком виде, в котором к ней легко заново обратиться. База данных позволяет многочисленным пользователям обновлять, редактировать и поддерживать хранилище данных быстрым, безопасным и эффективным образом. Существуют множество разнообразных баз данных – и каждая из них имеет свои преимущества и недостатки.

Главная характерная черта баз данных заключается в том, что они представляют внутреннее представление (модель) внешнего интересующего объекта, или набора объектов.

Курсовой проект создавался с целью получения навыков разработки базы данных.

1. Теоретическая часть. Конфликт SQL и NoSQL

Последние 40 лет, бизнес предприятия полностью полагались на реляционные базы данных, которые использовали язык SQL. На картинке с сайта ScaleGrid видно, что реляционные базы данных продолжают доминировать. Согласно ScaleGrid, 60.5% баз данных в 2019 году – реляционные.

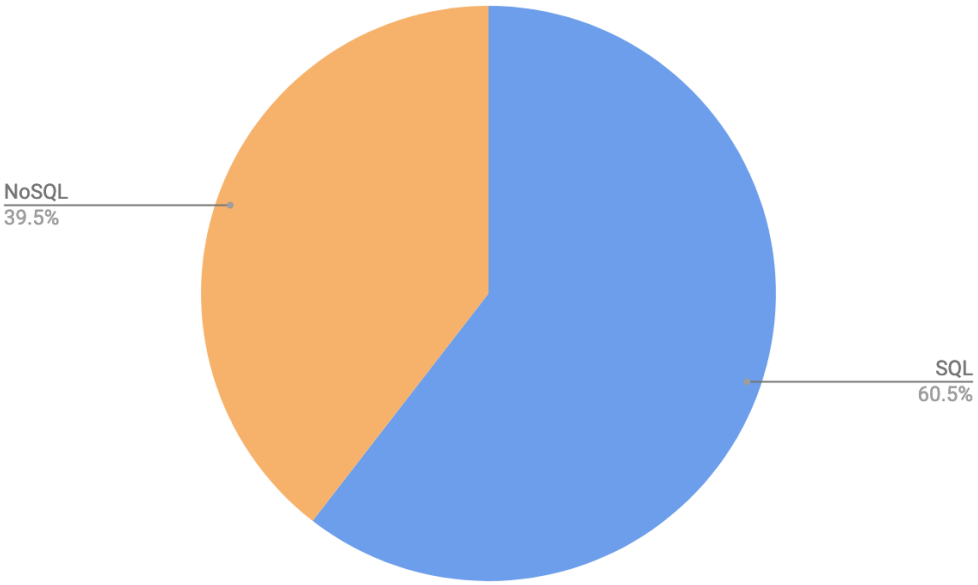

Рисунок 1.1 – Реляционные (Синим) и НеРеляционные (Желтым) базы данных на 2019 год.

С каждым годом нереляционные базы данных становятся популярнее. Это происходит отчасти потому что ученые в сфере информационных технологий хотят преобразовать их инструменты бизнес-аналитики и машинного обучение в более неструктурированную структуру данных.

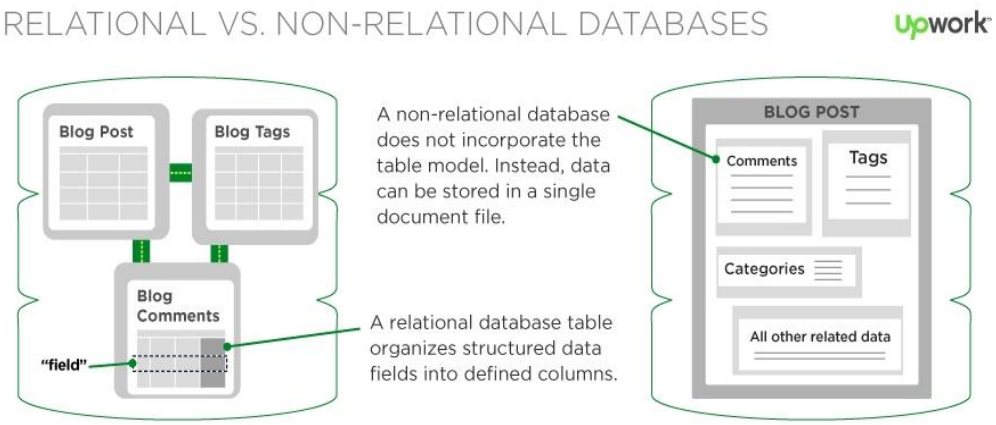

Рисунок 1.2 – Сравнение модели представления Реляционной и Нереляционной базы данных.

1.1 Реляционная база данных (основанная на SQL)

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется.

Важные аспекты реляционных баз данных:

- **SQL**

SQL (Structured Query Language) – основной интерфейс работы с реляционными базами данных. SQL стал стандартом Национального института стандартов США (ANSI) в 1986 году. Стандарт ANSI SQL поддерживается всеми популярными ядрами реляционных баз данных. Некоторые из ядер также включают расширения стандарта ANSI SQL, поддерживающие специфичный для этих ядер функционал. SQL используется для добавления, обновления и удаления строк данных, извлечения наборов данных для обработки транзакций и аналитических приложений, а также для управления всеми аспектами работы базы данных.

- **Целостность данных**

Целостность данных – это полнота, точность и единообразие данных. Для поддержания целостности данных в реляционных базах данных используется ряд инструментов. В их число входят первичные ключи, внешние ключи, ограничения «Not NULL», «Unique», «Default» и «Check». Эти ограничения целостности позволяют применять практические правила к данным в таблицах и гарантировать точность и надежность данных.

- **Транзакции**

Транзакция в базе данных – это один или несколько операторов SQL, выполненных в виде последовательности операций, представляющих собой единую логическую задачу. Транзакция представляет собой неделимое действие, то есть она должна быть выполнена как единое целое и либо должна быть записана в базу данных целиком, либо не должен быть записан ни один из ее компонентов. В терминологии реляционных баз данных транзакция завершается либо действием COMMIT, либо ROLLBACK. Каждая транзакция рассматривается как внутренне связный, надежный и независимый от других транзакций элемент.

- **Соответствие требованиям ACID**

Для соблюдения целостности данных все транзакции в базе данных должны соответствовать требованиям ACID, то есть быть атомарными, единообразными, изолированными и надежными.

1. **Атомарность** – это условие, при котором либо транзакция успешно выполняется целиком, либо, если какая-либо из ее частей не выполняется, вся транзакция отменяется.

2. **Единообразие** – это условие, при котором данные, записываемые в базу данных в рамках транзакции, должны соответствовать всем правилам и ограничениям, включая ограничения целостности, каскады и триггеры.

3. **Изолированность** необходима для контроля над согласованностью и гарантирует базовую независимость каждой транзакции.

4. **Надежность** подразумевает, что все внесенные в базу данных изменения на момент успешного завершения транзакции считаются постоянными.

1.2 Нереляционная база данных (основанная на NoSQL)

Базы данных NoSQL специально созданы для определенных моделей данных и обладают гибкими схемами, что позволяет разрабатывать современные приложения. Базы данных NoSQL получили широкое распространение в связи с простотой разработки, функциональностью и производительностью при любых масштабах. В них применяются различные модели данных, в том числе документные, графовые, поисковые, с использованием пар «ключ-значение» и хранением данных в памяти.

В базах данных NoSQL для доступа к данным и управления ими применяются различные модели данных. Базы данных таких типов оптимизированы для приложений, которые работают с большим объемом данных, нуждаются в низкой задержке и гибких моделях данных. Все это достигается путем смягчения жестких требований к непротиворечивости данных, характерных для других типов баз данных.

Базы данных NoSQL хорошо подходят для многих современных приложений, например мобильных, игровых, интернет-приложений, когда требуются гибкие масштабируемые базы данных с высокой производительностью и широкими функциональными возможностями, способные обеспечивать максимальное удобство использования.

Важные аспекты нереляционных баз данных:

- **Гибкость.**

Как правило, базы данных NoSQL предлагают гибкие схемы, что позволяет осуществлять разработку быстрее и обеспечивает возможность поэтапной реализации. Благодаря использованию гибких моделей данных баз данных NoSQL хорошо подходят для частично структурированных и неструктурированных данных.

- **Масштабируемость.**

Базы данных NoSQL рассчитаны на масштабирование с использованием распределенных кластеров аппаратного обеспечения, а не путем добавления дорогих надежных серверов. Некоторые поставщики облачных услуг проводят эти операции в фоновом режиме, обеспечивая полностью управляемый сервис.

- **Высокая производительность.**

Базы данных NoSQL оптимизированы для конкретных моделей данных (например, документной, графовой или с использованием пар «ключ-значение») и шаблонов доступа, что позволяет достичь более высокой производительности по сравнению с реляционными базами данных.

- **Широкие функциональные возможности.**

Базы данных NoSQL предоставляют API и типы данных с широкой функциональностью, которые специально разработаны для соответствующих моделей данных.

Базы данных NoSQL предлагают компромисс, смягчая жесткие требования свойств ACID ради более гибкой модели данных, которая допускает горизонтальное масштабирование. Благодаря этому базы данных на основе NoSQL – отличный выбор для примеров использования с высокой пропускной способностью и низкой задержкой, в которых требуется горизонтальное масштабирование, не ограниченное рамками одного инстанса.

2. Практическая часть. Системный анализ предметной области базы данных

подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами.

Вариант 3. Ломбард

Основная часть

Вы работаете в ломбарде. Вашей задачей является отслеживание финансовой стороны его работы. Деятельность компании организована следующим образом: к вам обращаются различные лица с целью получения денежных средств под залог определенных товаров. У каждого из приходящих к вам клиентов вы запрашиваете фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара вы определяете сумму, которую готовы выдать на руки клиенту, а также свои комиссионные. Кроме того, определяете срок возврата денег. Если клиент согласен, то ваши договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается у вас. В случае если в указанный срок не происходит возврата денег, товар переходит в вашу собственность.

Развитие постановки задачи

После перехода прав собственности на товар ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке. (Например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы.) Помимо текущей цены, нужно хранить все возможные значения цены для данного товара.

На каждого клиента ломбарда заводится своя таблица. Каждый клиент характеризуется следующими параметрами:

Отношение: Клиент

- паспорт серия и номер;
- имя;
- фамилия;
- отчество;
- почта;

Ценности, сданные под залог клиентами, могут иметь одинаковые названия, и их может быть несколько. На каждого клиента ломбарда заводится своя карточка со всеми заложенными ценностями, они характеризуются следующими параметрами:

Отношение: Предмет клиента

- Индивидуальный номер (id) предмета;
- Категория;
- Название предмета;
- Краткое описание;
- Цена за предмет;
- Комиссионные;
- Срок возврата денег;
- Паспорт владельца;
- Фотография предмета;

После перехода прав собственности на товар ломбард может продавать товары по собственной цене. Из карточки с заложенными ценностями предмет удаляется, и затем переходит в собственный перечень драгоценностей. Цена может меняться несколько раз, и необходимо учитывать все возможные цены для товара.

У ломбарда есть собственный перечень драгоценностей, который характеризуется следующими параметрами:

Отношение: Предмет в собственности ломбарда

- Индивидуальный номер (id) предмета;
- Категория;
- Название предмета;
- Краткое описание;
- Фотография предмета;
- Изначальная цена;
- Нынешняя цена;

Предусмотреть следующие ограничения на информацию в системе:

1. Каждый новый запрос на залог предмета, при уже существующих данных о пользователе, создают новое поле данных в личной карточке клиента. Должны совпадать:
    - Паспортные данные;
2. У каждого пользователя должна быть почта, для отправки оповещения об обработке его запроса и выставления цены и комиссионных.

С данной информационной системой должны работать следующие группы пользователей:

- Клиент;
- Администратор.

При работе с системой клиент должен иметь возможность решать следующие задачи:

1. Оставлять запрос на залог предмета в ломбарде.
2. Смотреть статус выставления цены на его предмет.

При работе с системой Администратор должен иметь возможность решать следующие задачи:

1. Выставлять цену на предмет, комиссионные, и сроки возвращения предмета при отмене договора.
2. После истечения срока содержания предмета переносить его из личной карты клиента в карту ломбарда, а также назначение новой цены на предмет и скидочных цен.

Администрация имеет полный доступ к системе – создание и удаление данных о клиентах, редактирование личных карт клиентов, редактирование карты ломбарда. Клиенты могут только оставлять заявки на залог и просматривать статус заявок.

2.1 Инфологическое (семантическое) моделирование предметной области базы данных

Диаграмма потоков данных DFD

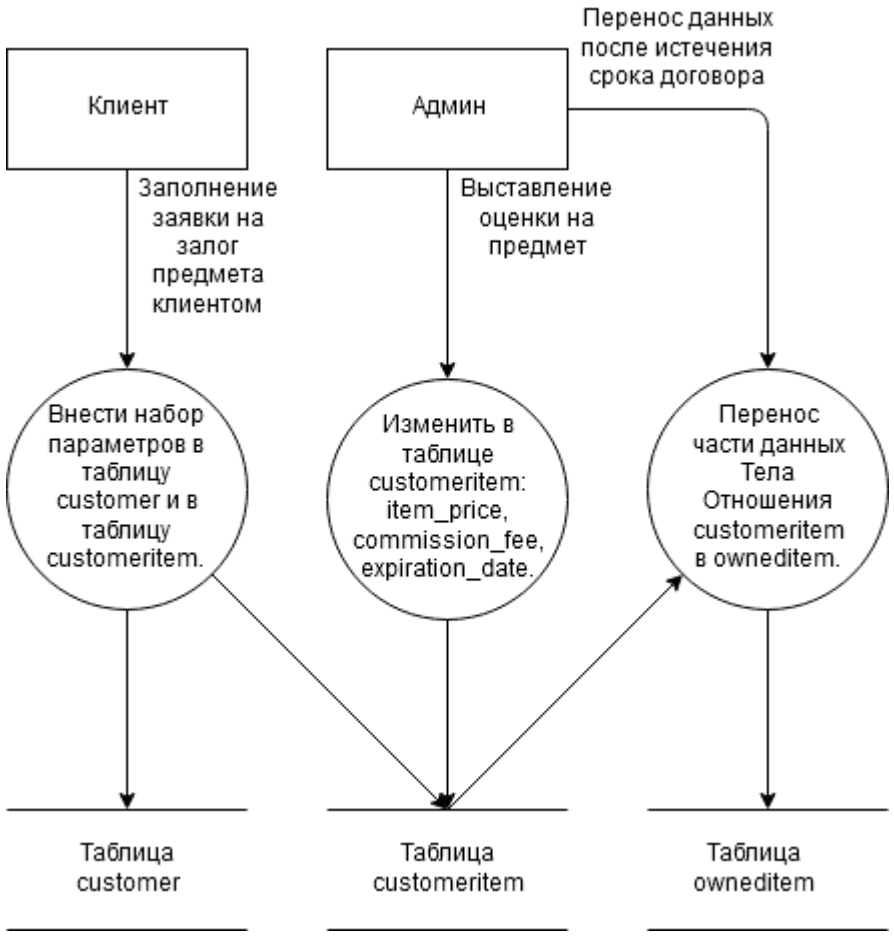

Рисунок 2.1.1 - Диаграмма потоков данных (нотация Йордона – Де Марко)

В дальнейшем для определения сущностей и их атрибутов будем использовать обозначение вида:

Клиенты ломбарда:

Клиент (паспорт серия и номер, имя, фамилия, отчество, почта)

Ключ сущности: паспорт серия и номер

Предметы, заложенные клиентами:

Карта Клиента (Индивидуальный номер (id) предмета, Категория, Название предмета, Краткое описание, Цена за предмет, Комиссионные, Срок возврата денег, Пасспорт владельца, Фотография предмета)

Ключ сущности: Индивидуальный номер (id) предмета

Предметы ломбарда:

Карта Ломбарда (Индивидуальный номер (id) предмета, Категория, Название предмета, Краткое описание, Фотография предмета, Изначальная цена, Нынешняя цена)

Ключ сущности: Индивидуальный номер (id) предмета

Связи между сущностями «Сущность-связь»

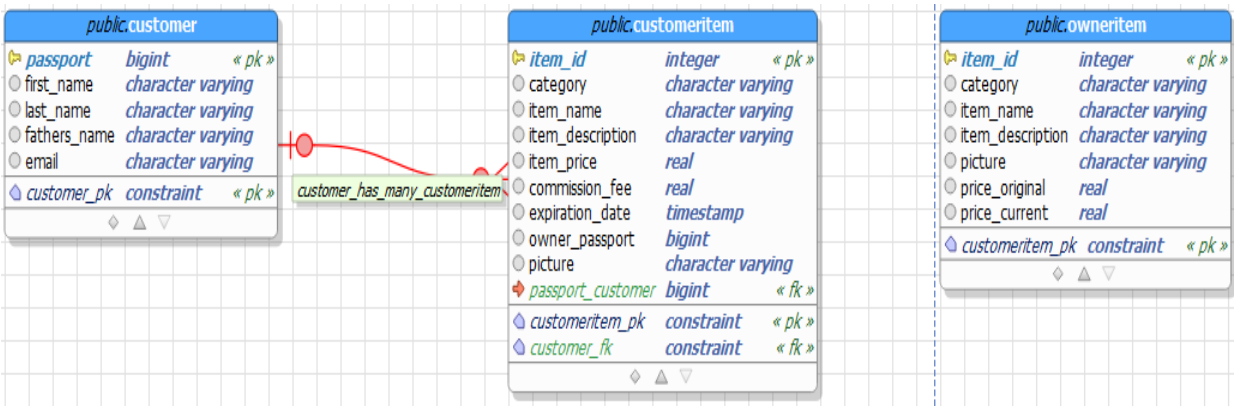

Рисунок 2.1.2 – Диаграмма отношений «Сущность-Связь» (ER диаграмма)

2.2 Нормализация отношений, построение нормальных форм

Отношение customer:

Атрибут отношения	Пример
#passport	7616123456
first_name	Vladimir
last_name	Demidov
fathers_name	Yuryevich
email	vladimirdemidov19@gmail.com

Таблица 2.2.1 Отношение клиент, с примером данных

- Переменные отношения **customer** находятся в 1 нормальной форме (1NF), так как все атрибуты отношения атомарны. (не обеспечивается явная видимость внутренней структуры атрибутов, со всеми значениями можно обращаться только с помощью операций, определенных в соответствующем типе данных).

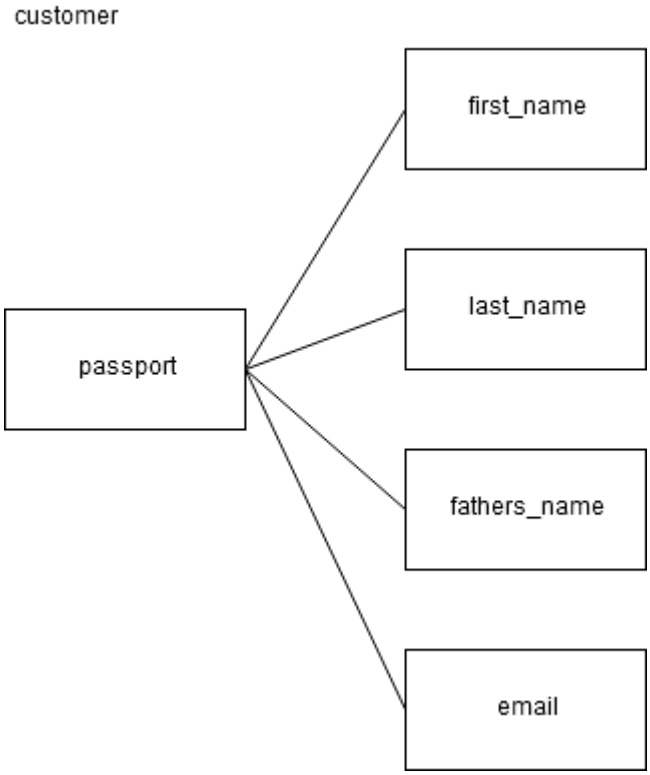

Рисунок 2.2.1 – схема функциональных отношений (FD) customer

- Переменные отношения **customer** находятся во 2 нормальной форме (2NF), так как они уже находится в 1 нормальной форме (1NF), и каждый неключевой атрибут своего отношения минимально функционально зависит от первичного ключа в своем отношении.

Отношение customeritem:

Атрибут отношения	Пример
#item_id	1
category	RINGS
item_name	moonring
item_description	silver ring with moonstone
item_price	2999
commission_fee	149.95
expiration_date	21.02.20
owner_passport	7616123456
picture	myPrecious.jpg

Таблица 2.2.2 Отношение предмет клиента, с примером данных

- Переменные отношения **customeritem** находятся в 1 нормальной форме (1NF), так как все атрибуты отношения атомарны. (не обеспечивается явная видимость внутренней структуры атрибутов, со всеми значениями можно обращаться только с помощью операций, определенных в соответствующем типе данных).

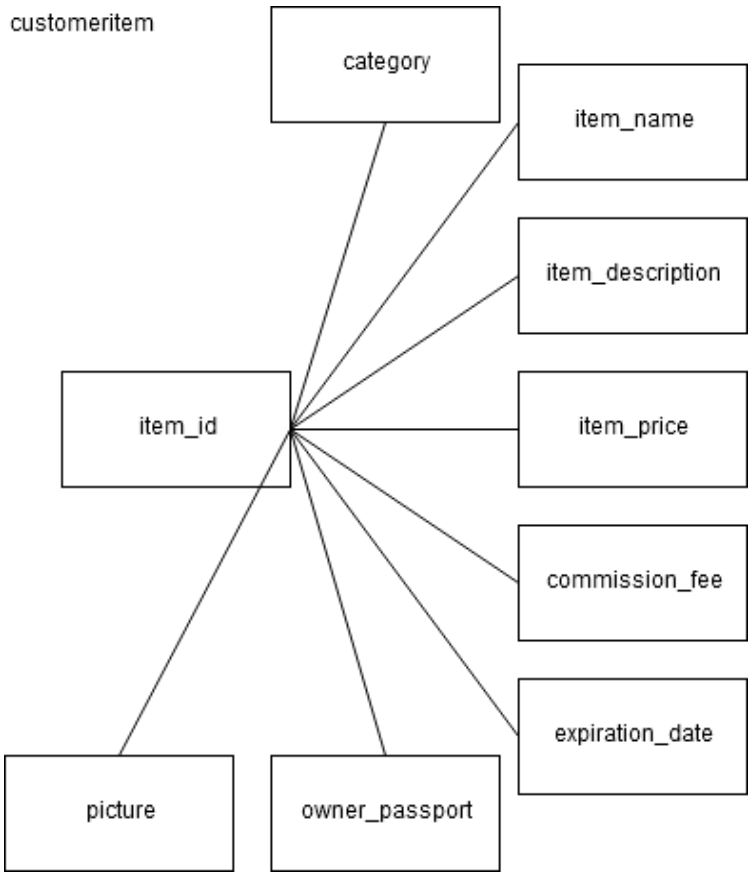

Рисунок 2.2.2 - схема функциональных отношений (FD) customeritem

- Переменные отношения **customeritem** находятся во 2 нормальной форме (2NF), так как они уже находится в 1 нормальной форме (1NF), и каждый неключевой атрибут своего отношения минимально функционально зависит от первичного ключа в своем отношении.

Отношение owneditem:

Атрибут отношения	Пример
#item_id	2
category	UNIQUE
item_name	Golden Crown
item_description	925 probe golden crown.
picture	a_crown_for_a_king.png
price_original	29999
price_current	28499.05

Таблица 2.2.3 Отношение предмет в собственности ломбарда, с примером данных

- Переменные отношения **owneditem** находятся в 1 нормальной форме (1NF), так как все атрибуты отношений атомарны. (не обеспечивается явная видимость внутренней структуры атрибутов, со всеми значениями можно обращаться только с помощью операций, определенных в соответствующем типе данных).

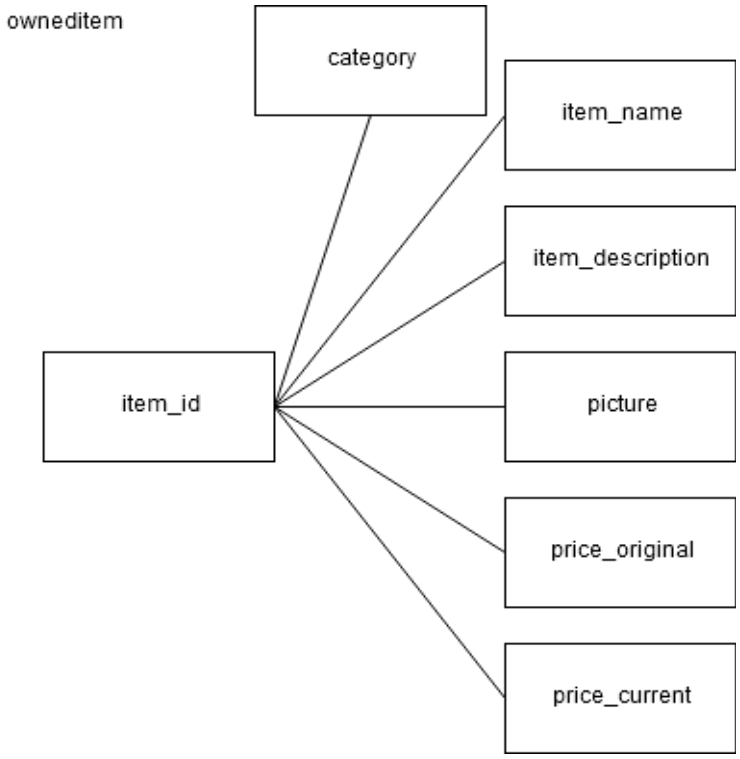

Рисунок 2.2.3 - схема функциональных отношений (FD) owneditem

- Переменные отношения **owneditem** находятся во 2 нормальной форме (2NF), так как они уже находится в 1 нормальной форме (1NF), и каждый неключевой атрибут своего отношения минимально функционально зависит от первичного ключа в своем отношении.

- Переменные отношений **customer, customeritem, owneditem** находятся в 3 нормальной форме (3NF), так как они находятся во второй нормальной форме, и каждый неключевой атрибут своего отношения нетранзитивно функционально зависит от первичного ключа своего отношения (не существует такого множества атрибутов, чтобы одно из отношений

определяло его, и чтобы одновременно оно определяло другое отношение).

- Переменные отношений **customer, customeritem, owneditem** находятся в Нормальной форме Бойса-Кодда (BCNF), так как детерминанты всех их функциональных зависимостей являются потенциальными ключами.

- Переменные отношений **customer, customeritem, owneditem** находятся в 4 Нормальной форме (4NF), так как они уже находятся в нормальной форме Бойса-Кодда, и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от их потенциальных ключей.

Когда человек оставляет заявку на оценку украшения, для него создается поле в отношении customer. Один человек может создавать множество заявок, но поле customer будет только одно. Ключ отношения customer – passport, является единственным и индивидуальным для каждого человека. На один атрибут passport только один экземпляр first_name, last_name, fathers_name и email.

У каждого поля в отношении customeritem первичный ключ – item_id индивидуален для каждого предмета – можно добавлять неограниченное количество предметов. В каждом поле есть атрибут owner_passport, с помощью которого можно узнать какой клиент добавил предмет. Для первичного ключа item_id существует единственный экземпляр атрибутов category, item_name, item_description, item_price, owner_passport, picture, заполняемых клиентом, и единственный экземпляр атрибутов item_price, commission_fee, expiration_date заполняемых администратором (оценка предмета). Между первичными ключами passport отношения customer и item_id отношения customeritem может быть функциональная зависимость.

При переходе предмета в собственность ломбарда происходит перенос поля из отношения customeritem в поле owneditem с первичным ключом item_id, для которого существуют в единственном экземпляре атрибуты item_name, item_description, picture, price_original, price_current.

2.3 Реализация декларативных ограничений целостности, проектирование таблицы Базы Данных

Ограничение целостности – некоторое утверждение, которое может быть либо истинным, либо ложным, в зависимости от состояния базы данных. База данных находится в согласованном (целостном) состоянии, если выполнены все ограничения целостности, определенные для базы данных. Любое ограничение целостности появляется как следствие определенных свойств объектов предметной области и/или их взаимосвязей.

У всех клиентов должен быть свой паспорт, по которому в дальнейшем можно будет вести поиск предметов, которые они сдают в ломбард. Так как паспорт для каждого человека индивидуален, он должен быть первичным ключом. У всех клиентов также должны быть имя, фамилия и отчество. У всех клиентов должна быть почта.

Ограничения целостности для Отношения customer:

- Атрибут passport – not null;
- Атрибут passport – primary key;
- Атрибут first_name – not null;
- Атрибут last_name – not null;
- Атрибут fathers_name – not null;
- Атрибут email – not null;

SQL код – реализация:

```
CREATE TABLE customer{
passport bigint NOT NULL,
first_name varchar(255) NOT NULL,
last_name varchar(255) NOT NULL,
fathers_name varchar(255) NOT NULL,
email varchar(255) NOT NULL,
PRIMARY KEY (passport)
};
```

Все предметы клиентов должны иметь собственный идентификационный номер. У каждого предмета должна быть категория, для сортировки предметов (при неправильном указании категории поле этого атрибута редактируется администратором при оценке предмета, и на поиск это не влияет). Предмету необходимо дать название. Описание для предмета – не обязательно, но желательно. Цена предмета – выставляется администратором, и не должна быть меньше нуля. Комиссионные – выставляются администратором, и не должны быть меньше нуля. Дата истечения срока договора – выставляется администратором, и она должна быть не равной текущему дню. Паспорт владельца должен быть заполнен. Фотография предмета тоже необходима для качественной оценки.

Ограничения целостности для Отношения customeritem:

- Атрибут item_id – not null;
- Атрибут item_id – default self-increment +1;
- Атрибут item_id – primary key;
- Атрибут category – not null;
- Атрибут item_name – not null;
- Атрибут item_price – check greater than 0;
- Атрибут commission_fee – check greater than 0;
- Атрибут expiration_date – check current_timeStamp != today;
- Атрибут owner_passport – not null;
- Атрибут owner_passport – foreign key;
- Атрибут picture – not null;

SQL код – реализация:

```
CREATE TABLE customeritem{
item_id bigserial,
category varchar(255) NOT NULL,
item_name varchar(255) NOT NULL,
item_description varchar(255),
item_price real,
commission_fee real,
expiration_date timestamp,
owner_passport bigint NOT NULL,
picture varchar(255) NOT NULL,
PRIMARY KEY (item_id),
FOREIGN KEY (owner_passport) REFERENCES customer (passport),
CHECK (item_price > 0),
CHECK (commission_fee > 0),
CHECK (expiration_date != UNIX_TIMESTAMP(CURDATE()))
};
```

Все предметы, находящиеся в собственности ломбарда, должны иметь собственный идентификационный номер. Они должны иметь категорию, чтобы отображаться на сайте в соответствующих категориях. Они должны иметь название. Описание – желательно. Клиентов не получится заинтересовать товаром без фотографии – она обязательна. Обязательным должен быть атрибут изначальной цены для предмета, так как скидки будут вычисляться из этого значения в обязательный атрибут нынешней цены. При этом изначальная цена и нынешняя цена должны быть больше нуля.

Ограничения целостности для Отношения owneditem:

- Атрибут item_id – not null;
- Атрибут item_id – default self-increment +1;
- Атрибут item_id – primary key;
- Атрибут category – not null;
- Атрибут item_name – not null;
- Атрибут picture – not null;

- Атрибут price_original – not null;
- Атрибут price_original – check greater than 0;
- Атрибут price_current – not null;
- Атрибут price_current –check greater than 0;

SQL код – реализация:

```
CREATE TABLE owneditem{
item_id bigserial,
category varchar(255) NOT NULL,
item_name varchar(255) NOT NULL,
item_description varchar(255),
picture varchar(255) NOT NULL,
price_original real NOT NULL,
price_current real NOT NULL,
PRIMARY KEY (item_id),
CHECK (price_original > 0),
CHECK (price_current > 0),
};
```

Связи между сущностями «Сущность-связь»

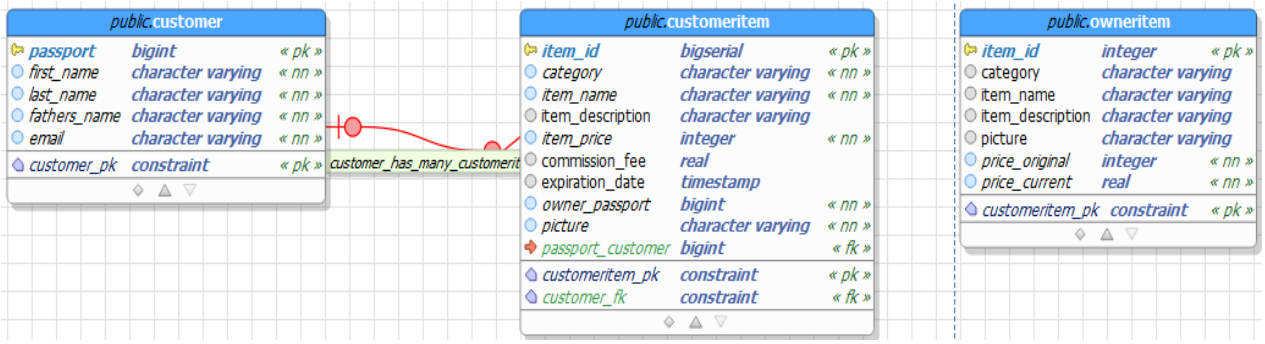

Рисунок 2.3.1 – Диаграмма отношений «Сущность-Связь» базы данных в целостном состоянии (ER диаграмма)

2.4 Реализация запросов к Базе Данных с использованием оператора выборки SELECT

SELECT - оператор запроса в языке SQL, возвращающий набор данных (выборку) из базы данных. Оператор возвращает ноль или более строк. Список возвращаемых столбцов задается в части оператора, называемой предложением SELECT. Поскольку SQL является декларативным языком, запрос SELECT определяет лишь требования к возвращаемому набору данных, но не является точной инструкцией по их вычислению. СУБД транслирует запрос SELECT во внутренний план исполнения («query plan»).

Вариант 3: Ломбард

1. **Список товаров, перешедших в собственность ломбарда.**

При переходе предмета в собственность ломбарда происходит перенос конкретной строки из отношения customeritem в отношение owneditem (истекает срок договора). Для того чтобы вывести полный список товаров, перешедших в собственность ломбарда, можно использовать следующий SQL код:

SQL реализация:

```
SELECT * FROM owneditem;
```

Результат:

	item_id integer	category character varying (255)	item_name character varying (255)	item_description text	picture character varying (255)	price_original real	price_current real
1	1	RINGS	Crowned Heart 851 985	851 silver ring with ...	image_2019_12_16T13_44_...	7999	7999
2	2	BRACELETS	green tennis balls	3 karat emerald 851...	image_2019_12_16T13_44_...	18990	18990
3	3	UNIQUES	Moon locket heart	Designer's special	image_2019_12_16T13_44_...	14990	14990
4	4	UNIQUES	cube world	cube world made by...	1_b74b770d-45a2-4dc3-89...	2999	2999
5	5	UNIQUES	hypnotic frog	Hypnotic frog made...	61gZ+qAuDSL._UY500_.jpg	3999	3999
6	6	UNIQUES	chain ring	851 probe silver cha...	fluidsSilvChainRing851.jpg	1999	1999
7	7	UNIQUES	Outsiders talisman	made from wood an...	0808201821032526151.jpg	4999	4999

Рисунок 2.4.1 – Результат вывода Query Tool в PGAdmin postgresql.

2. **Количество товаров по каждой категории.**

Каждый товар имеет атрибут category, который описывает категорию, к которой относится конкретный товар. Для подсчета количества вообще используется оператор **COUNT**. Для вывода количества товаров по каждой категории можно также использовать оператор **GROUP BY**.

SQL реализация:

```
SELECT category,COUNT(*)
FROM owneditem
GROUP BY category;
```

Результат:

	category character varying (255)	count bigint
1	EARRINGS	6
2	DISCOUNTS	6
3	RINGS	5
4	BRACELETS	6
5	NECKLACES	6
6	CHAINS	6
7	UNIQUES	15

Рисунок 2.4.2 – Результат вывода Query Tool в PGAdmin postgresql.

3. **Список клиентов, у которых не истек срок договора.**

При истечении срока договора (когда timestamp в поле expiration_date принимает значение сегодняшней даты timestamp_current) данные переносятся из отношения customeritem в owneditem. Поэтому можно просто сразу вывести клиентов (пасспорт, имя, фамилия, отчество) из отношения customer, и дату окончания договора из отношения customeritem, в котором указан владелец в атрибуте owner_passport. Так как вещей у клиентов несколько, сроки договора у каждого предмета разные, поэтому нужно вывести все те предметы, у которых срок договора не истек. Для вывода можно использовать **INNER JOIN ON** (условие, что есть соответствие паспортов из отношений customer и customeritem), чтобы были выведены только те существующие клиенты, у которых существует предмет (не перенесен так как срок договора не истек).

SQL реализация:

```
SELECT customer.passport, customer.first_name, customer.last_name,
        customeritem.item_name, customeritem.expiration_date
FROM customer
INNER JOIN customeritem
ON customer.passport = customeritem.owner_passport;
```

Результат:

	passport bigint	first_name character varying (255)	last_name character varying (255)	item_name character varying (255)	expiration_date date
1	7616123456	Vladimir	Demidov	Star stone 52gr	2020-07-19

Рисунок 2.4.3 – Результат вывода Query Tool в PGAdmin postgresql.

4. Сумма цен всех товаров, находящихся в собственности ломбарда.

Все предметы, находящиеся в собственности ломбарда, находятся в отношении owneditem. Атрибут отношения, характеризующий нынешнюю цену товара – price_current. Так как нужно посчитать сумму всех товаров, без учета их категории, можно сразу использовать функцию **SUM()**. Если понадобится конкретная категория, можно ввести ограничение **WHERE** category=’название_категории’.

SQL реализация:

```
SELECT SUM(price_current)
FROM owneditem;
```

Результат:

	sum real
1	931114

Рисунок 2.4.4 – Результат вывода Query Tool в PGAdmin postgresql.

5. Вывести сумму цен из самой прибыльной категории

Все предметы, находящиеся в собственности ломбарда, находятся в отношении owneditem. Атрибут отношения, характеризующий нынешнюю цену товара – price_current. Необходимо вывести сумму той категории, в которой эта сумма является наибольшей. Для решения данной задачи можно использовать вложенный оператор **SELECT**. Сначала (во вложенном **SELECT**) находим сумму цен для каждой категории, и группируем их по категории, затем во внешнем **SELECT** выводим максимальное из получившехся.

SQL реализация:

```
SELECT MAX(sumRes.sumPrice)
from (SELECT category, SUM(price_current) sumPrice
FROM owneditem
GROUP BY category) sumRes;
```

Результат:

	max real
1	458654

Рисунок 2.4.5 – Результат вывода Query Tool в PGAdmin postgresql.

2.5 Реализация процедурных ограничений целостности, создание триггеров.

Триггеры - триггерные процедуры, которые вызываются при изменениях данных или событиях в базе данных. Триггерная процедура создаётся командой CREATE FUNCTION, при этом у функции не должно быть аргументов, а типом возвращаемого значения должен быть trigger (для триггеров, срабатывающих при изменениях данных) или event_trigger (для триггеров, срабатывающих при событиях в базе). Для триггеров автоматически определяются специальные локальные переменные с именами вида PG_переменная, описывающие условие, повлекшее вызов триггера.

Триггер DELETE

Для базы данных ломбард необходим триггер, который срабатывает при удалении данных. Все удаленные данные помещаются в виртуальную таблицу **DELETED**. Удаляются данные из отношения customeritem, при истечении срока договора с клиентом. При этом, предметы переходят в собственность ломбарда, значит после удаления нужно перенести данные в отношение owneditem. Переносить необходимо только часть данных – категория удаляемого предмета, название удаляемого предмета, введенное клиентом (которое в дальнейшем может изменить администратор), описание удаляемого предмета, введенное клиентом, фотография удаляемого предмета, загруженная клиентом, и та цена, которая была выставлена при оценке.

Важно отметить, что из-за того что отношения customeritem и owneditem это два разных отношения, первый характеризует предметы которые принадлежат клиентам, а второй характеризует предметы, находящиеся в собственности ломбарда, каждый предмет характеризуется собственным индивидуальным номером, который не нужно переносить. Тут и не нужен триггер, проверяющий совпадения индивидуальных номеров, так как даже при их совпадении, оба отношения никак не связаны. Для генерации индивидуального номера в обоих отношениях используется тип **bigserial**, который автоматически инкрементируется во время каждого нового **INSERT** в отношение, что исключает возможность совпадения индивидуальных номеров

при переносе, но только внутри собственного отношения. Поэтому, наличие триггеров проверки совпадений не требуется.

SQL Реализация:

```
CREATE TRIGGER owneditem_DELETE
ON owneditem
AFTER DELETE
AS
INSERT INTO owneditem (category, item_name, item_description, picture,
price_original)
SELECT category, item_name, item_description, picture, item_price
FROM DELETED
```

Результат:

После удаления данных в отношении customeritem вручную, данные были автоматически добавлены с помощью триггера в отношение owneditem в конец, был автоматически сгенерирован индивидуальный номер №48, а данные в атрибутах категория, название, описание, картинка, цена были перенесены.

	item_id integer	category character varying (255)	item_name character varying (255)	item_description text	picture character varying (255)	price_original real	price_current real
39	42	NECKLACES	Giants eye	necklace with blue ...	61FsAbkJAPL._UY395_.jpg	4700	4700
40	43	NECKLACES	Heart locket	Red gold locked in a...	37240245_1_640.png	6100	6100
41	45	NECKLACES	Honeycombs	Unique golden neckl...	Ana_Luisa_Necklaces_Laye...	6400	6400
42	46	NECKLACES	Sunshine	golden necklace wit...	cbd18yams02sg_detail.jpg	7100	7100
43	47	NECKLACES	Zoo twin necklaces	Two necklaces with ...	rose_gold_product_2100_1_...	6100	6100
44	48	NECKLACES	Star system	Svarovski multicolor...	swarovski-symbolic57155.j...	8400	8400

Рисунок 2.5.1 – Результат вывода Query Tool в PGAdmin postgresql.

2.6 Методы взаимодействия с базой данных через пользовательский интерфейс

Использование PgModeler или PgAdmin недостаточно для полноценного взаимодействия с базой данных (получение, изменение или удаление данных в базе). В каждом языке программирования свои методы доступа к базе данных. Один из них – получение данных с помощью скриптового языка PHP, и последующая загрузка данных на сайт. PHP это скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Был разработан сайт «JewelBling» на базе PHP, в котором можно просматривать каталог товаров ломбарда (загрузка из базы данных на сайт), и добавление украшений в залог, в режиме онлайн (добавление в базу данных с сайта).

Благодаря наличию в отношении owneditem атрибута category, в PHP можно запросить ассоциативные массивы по условию содержания в них строго определенной категории, с помощью использования SQL вставок прямо в PHP.

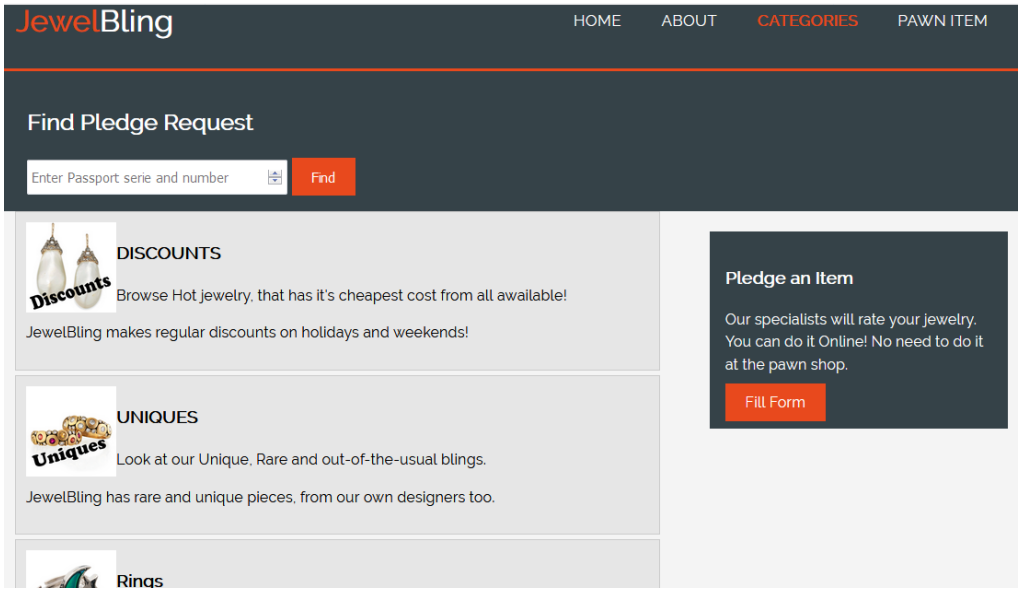

Рисунок 2.6.1 – Категории на сайте JewelBling

Загрузка данных с базы данных на сайт

В каждую категорию необходимо загружать только соответствующие ей данные. Это реализовано в PHP коде сайта.

PHP реализация:

```
<?php
	require('config/db.php'); // Первый блок

	// Create Query // Второй блок
	$query = 'SELECT item_name, item_description,
	picture, price_original, price_current FROM owneditem
	WHERE category = \'DISCOUNTS\'";

	// Get Result // Третий блок
	$result = pg_query($conn, $query);

	// Fetch data // Четвертый блок
	$items = pg_fetch_all($result);
	#var_dump($items);

	// Free results, close connection // Пятый блок
	pg_free_result($result);
	pg_close($conn);
?>
```

В первом блоке кода мы подгружаем отдельный скрытый файл, в котором хранятся данные о базе данных (имя суперпользователя, название базы данных, пароль итп). Они необходимы для того чтобы точно определить с какой базой данных мы взаимодействуем, и предоставить права суперпользователя.

Во втором блоке создается очередь, в которую передается SQL код. В этом конкретном случае мы загружаем все данные из категории discounts.

В третьем блоке мы получаем результат с данными. В функцию pg_query передается переменная conn, которая содержит все данные о базе данных.

В четвертом блоке мы получаем ассоциативный массив, в котором можно обращаться к данным прямо по имени атрибутов в базе данных, что очень удобно.

В пятом блоке для экономии памяти освобождается переменная с результатом, и прерывается соединение с базой данных.

Необходимо взять во внимание, что в PHP для доступа к базе данных Postgresql использовались отдельные функции, для подключения к другим видам баз данных необходимо использовать другие функции, однако методы их использования идентичны.

Содержание скрытых файлов config.php и db.php

#config.php

```
<?php // Шестой блок
        define('ROOT_URL', 'http://localhost/phpsandbox/JewelBling/');
        define('DB_HOST', 'host=localhost');
        define('DB_NAME', 'dbname=jewelblingdb');
        define('DB_USER', 'user=postgres');
        define('DB_PW', 'password=12345');
?>
```

#db.php

```
<?php // Седьмой блок
        require('config/config.php');
        // Create connection to DB
        $conn = pg_connect(DB_HOST.' '.DB_NAME.' '.DB_USER.' '.DB_PW);
?>
```

Ассоциативный массив можно перебирать в цикле, загружая данные в необходимые части сайта, обращаясь к данным по названию атрибутов, в которых они хранились в базе данных.

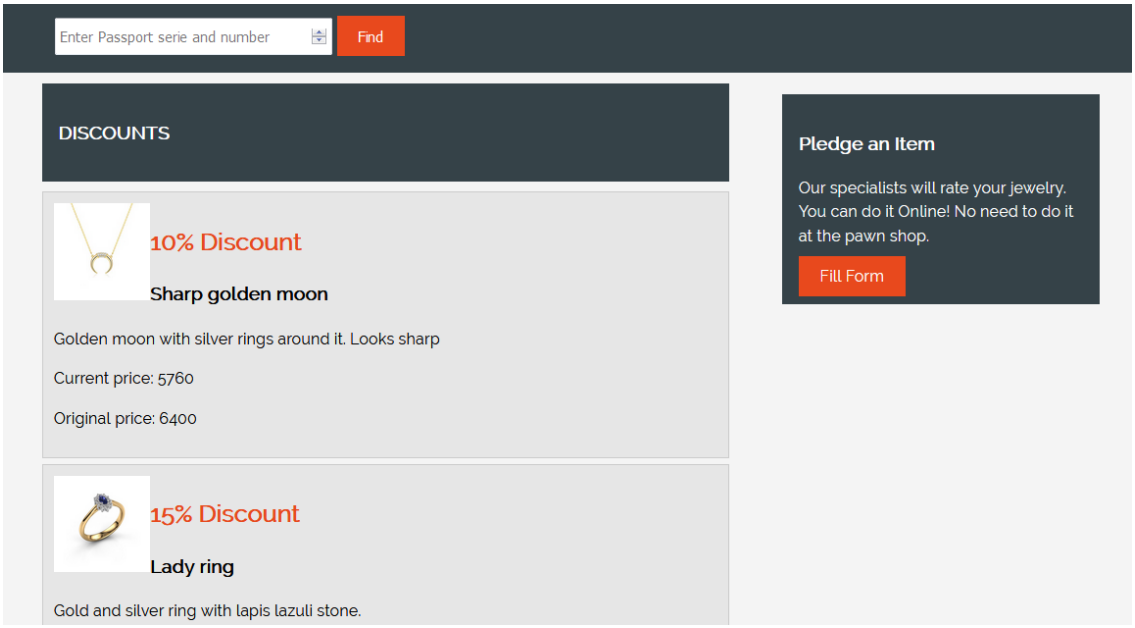

Рисунок 2.6.2 – Данные из базы данных на сайте

Также, из базы данных происходит загрузка заявок, которые оставлялись клиентами. PHP реализация и метод получения ассоциативного массива аналогичны методу и реализации в разделах категорий, только на этот раз в SQL коде происходит сравнивание и проверка атрибута owner_passport.

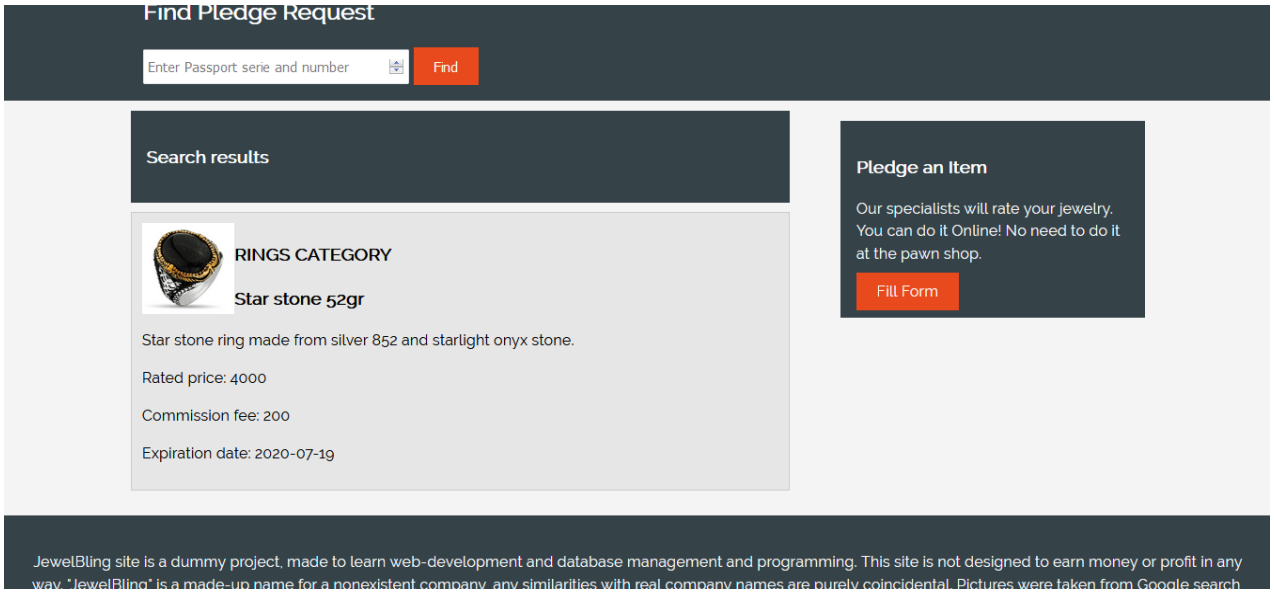

Рисунок 2.6.3 – Результаты поиска по паспорту для оставленной на сайте заявки.

Добавление данных с сайта в базу данных

Каждый клиент, просматривающий сайт, может оставить заявку на оценку украшений, которые он хочет отдать в залог. Для этого необходимо заполнить форму. На страницу заполнения формы можно перейти из любого раздела сайта. При заполнении формы создается единственный экземпляр данных клиента в отношении customer, и сколько угодно заявок на оценку в отношение customeritem.

PHP реализация:

```
<?php
	require('config/db.php'); // database information // Первый блок
	// Get submitted data and send it to DB
	if(isset($_POST['upload'])){
		// Get all the submitted data from the form
		$firstName = pg_escape_string($_POST['first_name']);
		$lastName = pg_escape_string($_POST['last_name']);
		$fathersName = pg_escape_string($_POST['fathers_name']);
		$email = $_POST['email'];
		$passport = $_POST['passport'];
		$jewelryCategory = $_POST['category'];
		$jewelryName = pg_escape_string($_POST['jewelry_name']);
		$jewelryDescriprion = pg_escape_string($_POST['jewelry_description']);

		// Check if customer record already exists // Второй блок
		$res1 = pg_query("SELECT * FROM customer
		WHERE passport='$passport'");
		$num_rows = pg_num_rows($res1);
		if(!$num_rows){ // New customer // Третий блок
			// First create new customer in db
			$query1 = "INSERT INTO customer(passport, first_name,
			last_name,fathers_name,email) VALUES('$passport',
			'$firstName','$lastName','$fathersName','$email')";
			if(pg_query($conn, $query1)){
				// Than create new item for that customer // Четвертый блок
				$image = $_FILES['image']['name'];
				$target = "img/localDB/".basename($image);
```

```
                        $query2 = "INSERT INTO customeritem(category, item_name,
                        item_description, owner_passport, picture)
                        VALUES('$jewelryCategory',
                        '$jewelryName','$jewelryDescriprion','$passport','$image')";
                        if(pg_query($conn, $query2)){
                                #header('Location: '.ROOT_URL.");
                                move_uploaded_file($_FILES['image']['tmp_name'], $target);
                                echo '<script type="text/javascript">alert("New customer,
                                request sent!");</script>';
                        }
                        else{
                                echo '<script type="text/javascript">alert("ERROR. Failed to
                                update database customeritem");</script>';
                        }

                }
                else{
                        echo '<script type="text/javascript">alert("ERROR. Failed to update
                database customer");</script>';
                }
        }
        else{ // Existing customer - we need just the jewelry info // Пятый блок
                $image = $_FILES['image']['name'];
                $target = "img/localDB/".basename($image);
                $query2 = "INSERT INTO customeritem(category, item_name,
                item_description, owner_passport, picture) VALUES('$jewelryCategory',
                '$jewelryName','$jewelryDescriprion','$passport','$image')";
                if(pg_query($conn, $query2)){
                        #header('Location: '.ROOT_URL.");
                        move_uploaded_file($_FILES['image']['tmp_name'], $target);
                        echo '<script type="text/javascript">alert("Existing customer, request
                        sent!");</script>';
                }
                else{
                        echo '<script type="text/javascript">alert("ERROR. Failed to update
                        database customeritem");</script>';
                }
        }
}
?>
```

В первом блоке происходит подключение к базе данных, затем заполненные данные, заполненные в полях, передаются в переменные.

Во втором блоке происходит проверка по паспорту, существует ли уже карта для клиента в отношении customer. Если да – происходит выполнение третьего блока – в отношение customer добавляются данные о клиенте, с индивидуальным первичным ключом – passport. Если же такой клиент уже существует – происходит переход в пятый блок.

В четвертом блоке уже происходит загрузка в базу данных информации о предмете, которую клиент загрузил.

В пятом блоке происходит тоже самое, что и в четвертом, но без изменения информации в отношении customer.

При ошибке загрузки в базу данных любого атрибута, данные в ней не изменяются, и происходит вывод сообщения об ошибке.

Следует отметить, что изменение данных реализуется аналогично добавлению данных, но нужно заменить в SQL коде INSERT на UPDATE, и передавать в WHERE параметры, соответствующие требованиям (что где менять).

ЗАКЛЮЧЕНИЕ

В 21 веке, веке информационных технологий, многие компании и предприятия должны выбирать, как взаимодействовать с данными, выбирать новые модели баз данных, и находить правильные аналитические инструменты для достижения максимума эффективности и пользы от данных, а также для разработки программ и приложений.

Технологии, используемые в базах данных, в основном поддерживают внутреннее представление в согласованности с внешней реальностью; это следует из последних исследований и разработок за последние 30 лет во многих отраслях, в которых еще очень много неизученного.

Процесс проектирования базы данных представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели.

Этапы проектирования включают: Системный анализ и словесное описание информационных объектов предметной области, Проектирование инфологической модели предметной области — частично формализованное описание объектов предметной области в терминах некоторой семантической модели, Даталогическое проектирование базы данных — описание базы данных в терминах принятой даталогической модели данных, и физическое проектирование базы данных — выбор эффективного размещения базы данных на внешних носителях для обеспечения наиболее эффективной работы приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wikipedia. Реляционная база данных [Электронный ресурс]//Электронный Текстовый документ. Режим доступа:

   https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85 свободный

2. Wikipedia. База Данных [Электронный ресурс]//Электронный Текстовый документ. Режим доступа:

   https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85 свободный

3. Wikipedia. NoSQL [Электронный ресурс]//Электронный Текстовый документ. Режим доступа: https://ru.wikipedia.org/wiki/NoSQL свободный

4. Wikipedia. Нормальная форма [Электронный ресурс]//Электронный Текстовый документ. Режим доступа: https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0 свободный

5. Wikipedia. PHP [Электронный ресурс]//Электронный Текстовый документ. Режим доступа: https://ru.wikipedia.org/wiki/PHP свободный

6. Wikipedia. SQL [Электронный ресурс]//Электронный Текстовый документ. Режим доступа: https://ru.wikipedia.org/wiki/SQL свободный

7. ScaleGrid. Облачный хостинг баз данных [Электронный ресурс]//Оказание услуг, электронная текстовая документация. Режим доступа: https://scalegrid.io/blog/ свободный