Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

«ВЕБ-СЕРВИС УПРАВЛЕНИЯ ИНТЕРНЕТ МАГАЗИНОМ»

БГУИР КП 1-40 01 01 112 ПЗ

Студент: гр. 151001 Матюшенко В. А.

Руководитель: асс. Фадеева Е. Е.

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5

1. АНАЛИЗ ПРОТОТИПОВ И ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ 6

2. АНАЛИЗ ТРЕБОВАНИЙ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 8

2.1 Выбор СУБД 8

2.2 Выбор инструмента для визуального проектирования 8

2.3 Описание функциональных требований 8

2.4 Спецификация функциональных требований 9

3. ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ 10

4. ОПИСАНИЕ БИЗНЕС-ЛОГИКИ 18

4.1 Описание таблиц 18

4.2 Процедуры и функции 19

4.3 Триггеры 20

5. ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ 21

ЗАКЛЮЧЕНИЕ 22

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23

ПРИЛОЖЕНИЕ А 24

ПРИЛОЖЕНИЕ Б 25

# ВВЕДЕНИЕ

На сегодняшний день одним из наиболее востребованных направлением в веб-разработке является создание веб-сервисов управления интернет магазином. База данных является неотъемлемой частью веб-сервиса. Именно от того, как спроектирована база данных, будут зависеть функциональность веб-сервиса, целостность данных, а также скорость записи и извлечения информации.

Пользователи ценят те интернет магазины, которые позволяют эффективно найти то, чего они хотят. Необходимо, чтобы в базе данных была возможность хранить информацию о распределении разных товаров по разным категориям, связанным иерархически. При этом должна быть возможность создания контента интернет магазина на разных языках. Атрибуты товаров не должны быть фиксированными, а могут меняться в зависимости от выбранного языка или самого товара. Помимо этого для администрирования интернет магазина удобно распределять товары по виртуальным магазинам, а те в свою очередь связывать со странами. Это также позволит ограничить набор товаров для жителей определённой страны. Обязательным для интернет магазина является совершение заказов, что тоже должно быть отражено в базе данных.

Цель данного курсового проекта заключается в разработке такой базы данных, которая может быть использована в качестве базы данных веб-сервиса управления интернет магазином. Такая база данных должна включать в себя таблицы для хранения информации о покупателях, товарах, категориях, заказах и других сущностях, необходимых для создания интернет магазином.

В настоящей пояснительной записке отражены следующие этапы написания курсового проекта:

1. Анализ прототипов и литературных источников.

2) Анализ требований и разработка функциональных требований.

3) Инфологическая модель.

4) Описание бизнес-логики.

5) Тестирование базы данных.

# АНАЛИЗ ПРОТОТИПОВ И ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

Прототипом программного средства является система управления интернет магазинами Magento, написанная на PHP и использующая реляционные базы данных. На Magento работают интернет‑магазины многих известных компаний: 20th Century Fox, Gap, Lenovo, Longines, Nestle Nespresso, Nokia, Olympus, Samsung, Skype, Time Out.

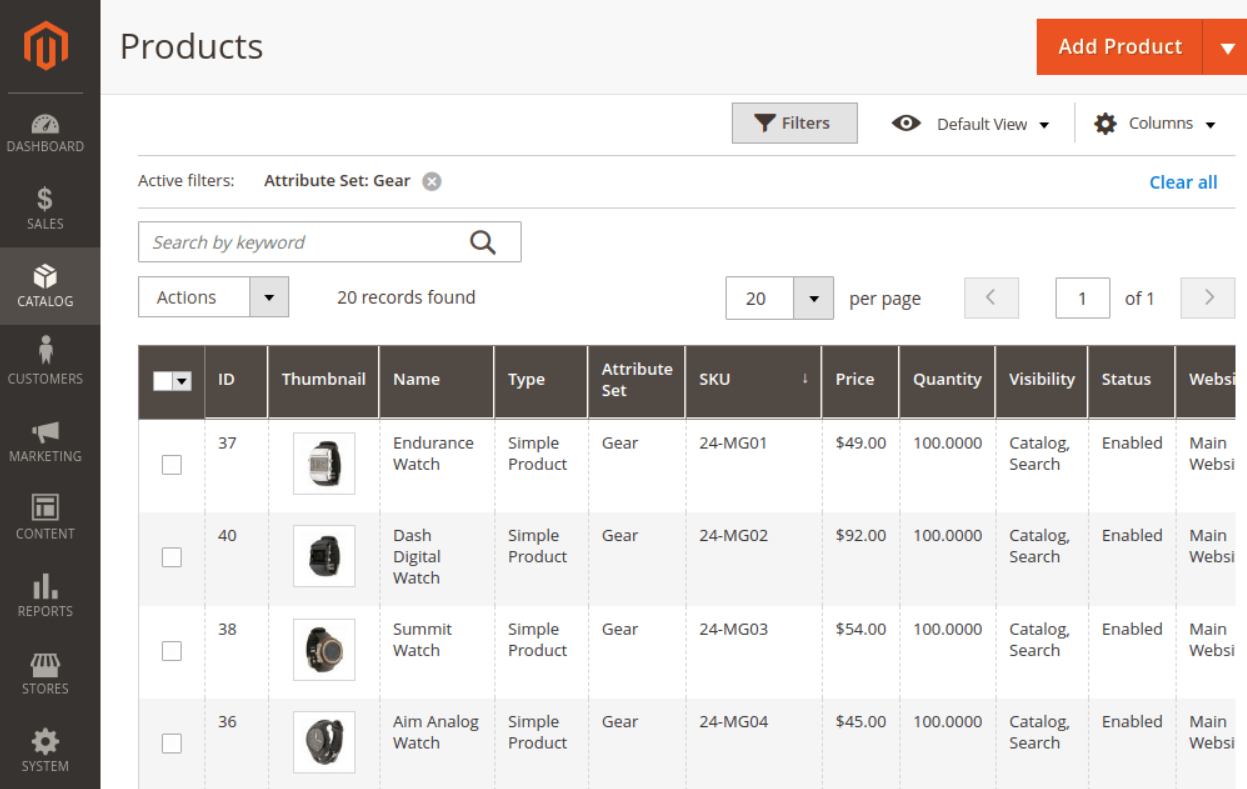


Рисунок 1.1 – Интерфейс панели администрирования в Magento

Magento предоставляет следующий список возможностей:

– возможность задания различных атрибутов для товара;

– сортировка и фильтрация товара по предопределенным атрибутам;

– управление торговой сетью интернет‑магазинов на разных доменах, языках, с разным товарным ассортиментом из панели администрирования;

– работа с разной валютой;

– гибкие инструменты работы с ценами;

– отчёты по продажам, по корзинам покупателя, по списку отмеченных товаров, по отзывам, по тегам, по поиску;

– поддержка различных способов оплаты и доставки;

– связь с покупателем;

– список желаний, корзина покупок, оформление заказа, личные кабинеты покупателей.

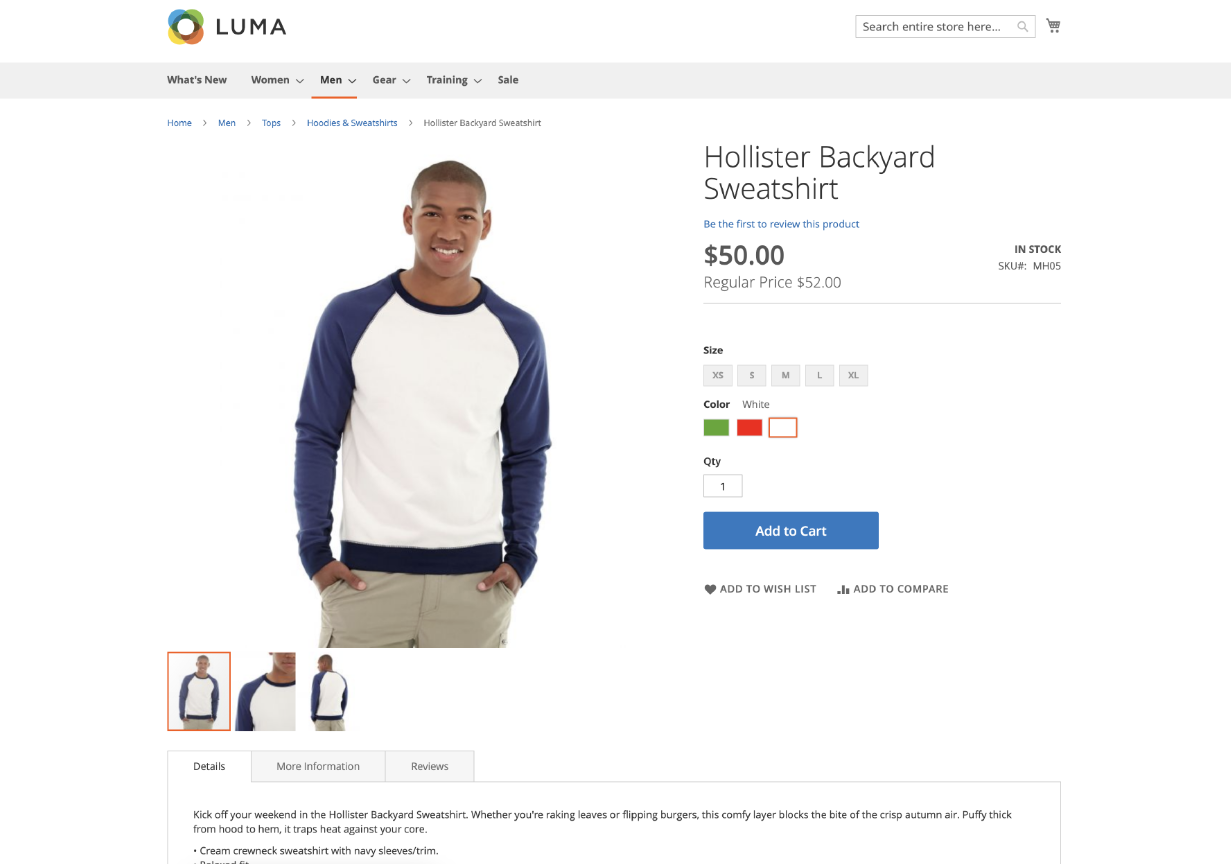


Рисунок 1.2 – Пример страницы товара в Magento

Было решено, что база данных должна хранить данные о покупателе и его корзине. Также в базе данных должны хранится товары, их атрибуты и категории. Обязательным является хранить историю заказов. Дополнительно можно сделать поддержку нескольких языков и валют. Для создания, редактирования и удаления товаров, категорий, языков и других сущностей должна быть предусмотрена возможность администрирования.

Для получения необходимой для работы информации я использовал следующие литературные источники:

1. Информация о системе управления интернет магазинами Magento.

2) Документация по реляционной системе управления базами данных MySQL.

3) Документация по инструменту для визуального проектирования базы данных MySQL Workbench.

# АНАЛИЗ ТРЕБОВАНИЙ И РАЗРАБОТКА ФУНКЦИОНАЛЬ- НЫХ ТРЕБОВАНИЙ

# 

# **2.1 Выбор СУБД**

В качестве СУБД была выбрана MySQL — реляционная система управления базами данных. Популярность MySQL обусловлена простотой СУБД, богатыми функциональными возможностями, встроенными системами безопасности, расширенными возможностями настройки и полной поддержкой SQL-языка. СУБД не перегружена функциями, поэтому работает быстрее и надёжнее многих аналогов.

MySQL позволяет осуществлять SQL-запросы SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT и WHERE частях запросов, работает с GROUP BY и ORDER BY, поддерживает агрегирующие функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать в запросах JOIN, поддерживает транзакции, работу с внешними ключами и каскадные операции на их основе, а также обеспечивает многие другие функциональные возможности.

## **2.2 Выбор инструмента для визуального проектирования**

Инструментом для визуального проектирования базы данных был выбран MySQL Workbench. Данный инструмент бесплатный и доступен на Windows, macOS, Linux и других платформах. К тому же большое количество качественных обучающих материалов по работе с MySQL Workbench находится в открытом доступе.

MySQL Workbench позволяет с помощью графического редактора создавать и редактировать ER-модели (Entity-Relationship), наглядно представляя сущности, связи и атрибуты базы данных. Также есть возможность генерировать SQL-код из модели и синхронизировать изменения. MySQL Workbench поддерживает сложные типы данных, индексы, триггеры, хранимые процедуры и другие элементы SQL.

## **2.3 Описание функциональных требований**

База данных должна отражать работу интернет магазина. На основании требований, выдвинутых в первом разделе, можно сформировать следующие функциональные требования:

– необходимо хранить данные о покупателе;

– необходимо хранить данные о корзине покупателя;

– необходимо хранить данные о товарах;

– необходимо хранить данные о категориях;

– необходимо хранить данные о атрибутах товаров;

– необходимо хранить историю заказов;

– необходимо хранить данные о языках;

– необходимо хранить данные о валютах;

– необходимо хранить данные о администраторах.

## **2.4 Спецификация функциональных требований**

Веб-сервис должен обеспечить функциональность для регистрации пользователя, просмотра товаров по категориям, поиска товаров, добавление товара в корзину, редактирования корзины с товарами, оформления заказа, редактирования и удаления категорий, товаров, языков и валют.

При регистрации указывается имя, фамилия, почта, дата рождения и страна. От выбранной страны будут зависеть показываемые товары, текущий язык и валюта.

Должно быть поддерживание неограниченного количества языков. Встроенным языком должен быть английский, он же является языком по-умолчанию. Должна быть возможность смены языка. При смене языка меняется текст страниц и описание товаров.

Также должно быть поддерживание неограниченного количества валют. Встроенной валютой должен быть доллар, он же является валютой по-умолчанию. Должна быть возможность смены валюты.

Категории представляют собой иерархическую структуру: одна родительская категория может иметь несколько вложенный категорий. Товары могут принадлежать сразу к нескольким категориям.

Сам товар содержит в себе медиа файлы, цену и набор атрибутов, задаваемых в зависимости от принадлежности к категории.

При выборе товара можно добавить его в корзину, указав положительное количество, меньшее или равное оставшимся на складе копиям товара. Из корзины можно оформить заказ на продукты из корзины.

Администратор, созданный вручную, может управлять категориями, товарами, языками, валютами и странами для продаж.

Администратор может добавлять, редактировать и удалять категории. При добавлении или редактировании можно указать родительскую категорию, изменять набор атрибутов и указать страну для показа категории.

Администратор может добавлять, редактировать и удалять товары. При добавлении или редактировании можно указать медиа файлы, категорию, заполнить атрибуты данной категории на всех доступных языках, указать цену во все доступных валютах и указать страну для показа товара.

Администратор может создавать и удалять языки, переводы содержимого страниц на разных языках, валюты и страны.

# ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ

Инфологическая модель базы данных находится в приложении А.

Таблица 3.1 – Отношение customers

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор покупателя | INT NOT NULL | PK |
| email | Email-адрес покупателя | VARCHAR(255) NOT NULL | UNIQUE |
| firstname | Имя покупателя | VARCHAR(45) NOT NULL |  |
| middlename | Отчество покупателя | VARCHAR(45) |  |
| lastname | Фамилия покупателя | VARCHAR(45) |  |
| password\_hash | Хеш пароля покупателя | VARCHAR(255) NOT NULL |  |
| is\_active | Флаг активности аккаунта покупателя | TINYINT |  |
| created\_at | Время создания покупателя | TIMESTAMP |  |
| updated\_at | Время обновления покупателя | TIMESTAMP |  |

Таблица 3.2 – Отношение customer\_addresses

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| customer\_id | Идентификатор покупателя | INT NOT NULL | FK (отношение customer) |
| city | Название города | VARCHAR(45) |  |
| country\_id | Идентификатор страны | INT NOT NULL | FK (отношение country) |
| postcode | Почтовый индекс | VARCHAR(6) |  |

Таблица 3.3 – Отношение countries

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор страны | INT NOT NULL | PK |
| name | Название страны | VARCHAR(45) NOT NULL |  |
| language\_id | Идентификатор языка | INT NOT NULL | FK (отношение language) |
| store\_id | Идентификатор виртуального магазина | INT NOT NULL | FK (отношение store) |
| currency\_id | Идентификатор валюты | INT NOT NULL | FK (отношение currency) |

Таблица 3.4 – Отношение languages

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор языка | INT NOT NULL | PK |
| name | Название языка | VARCHAR(45) NOT NULL |  |

Таблица 3.5 – Отношение page\_contents

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор контента страницы | INT NOT NULL | PK |
| name | Название контента | VARCHAR(45) NOT NULL |  |
| value | Значение контента | TEXT |  |
| language\_id | Идентификатор языка | INT NOT NULL | FK (отношение language) |

Таблица 3.6 – Отношение stores

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор виртуального магазина | INT NOT NULL | PK |
| name | Название виртуального магазина | VARCHAR(45) NOT NULL |  |
| is\_active | Флаг активности виртуального магазина | TINYINT |  |

Таблица 3.7 – Отношение m2m\_store\_product

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| store\_id | Идентификатор виртуального магазина | INT NOT NULL | FK (отношение store) |
| product\_id | Идентификатор товара | INT NOT NULL | FK (отношение product) |

Таблица 3.8 – Отношение m2m\_store\_category

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| store\_id | Идентификатор виртуального магазина | INT NOT NULL | FK (отношение store) |
| category\_id | Идентификатор категории | INT NOT NULL | FK (отношение category) |

Таблица 3.9 – Отношение products

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор товара | INT NOT NULL | PK |
| parent\_id | Идентификатор родительского товара | INT |  |
| sku | артикул | VARCHAR(45) NOT NULL | UNIQUE |
| created\_at | Время создания товара | TIMESTAMP |  |
| updated\_at | Время обновления товара | TIMESTAMP |  |
| count | Количество товара | INT NOT NULL |  |

Таблица 3.10 – Отношение product\_galleries

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор медиа файла | INT NOT NULL | PK |
| product\_id | Идентификатор товара | INT NOT NULL |  |
| product\_gallery\_type\_id | Идентификатор типа медиа файла | INT NOT NULL |  |
| value | Значение медиа файла | VARCHAR(255) NOT NULL |  |

Таблица 3.11 – Отношение product\_gallery\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор типа медиа файла | INT NOT NULL | PK |
| name | Название типа медиа файла | VARCHAR(45) NOT NULL |  |

Таблица 3.12 – Отношение m2m\_customer\_product

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| product\_id | Идентификатор товара | INT NOT NULL | FK (отношение product) |
| customer\_id | Идентификатор покупателя | INT NOT NULL | FK (отношение customer) |
| product\_count | Количество товаров | INT NOT NULL |  |

Таблица 3.13 – Отношение product\_prices

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор цены | INT NOT NULL | PK |
| product\_id | Идентификатор товара | INT NOT NULL | FK (отношение product) |
| currency\_id | Идентификатор валюты | INT NOT NULL | FK (отношение currency) |
| value | Значение цены | DECIMAL(6, 2) |  |

Таблица 3.14 – Отношение currencies

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор валюты | INT NOT NULL | PK |
| name | Название валюты | VARCHAR(45) NOT NULL |  |

Таблица 3.15 – Отношение categories

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор категории | INT NOT NULL | PK |
| parent\_id | Идентификатор родительской категории | INT |  |
| name | Название категории | VARCHAR(45) NOT NULL |  |
| child\_count | Количество дочерних категорий | INT NOT NULL |  |

Таблица 3.16 – Отношение m2m\_category\_product

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| product\_id | Идентификатор товара | INT NOT NULL | FK (отношение product) |
| category\_id | Идентификатор категории | INT NOT NULL | FK (отношение category) |

Таблица 3.17 – Отношение attribute\_sets

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор набора атрибутов | INT NOT NULL | PK |
| name | Название набора атрибутов | VARCHAR(45) NOT NULL |  |
| language\_id | Идентификатор языка | INT NOT NULL | FK (отношение language) |
| attribute\_template\_id | Идентификатор шаблона атрибута | INT NOT NULL | FK (отношение attribute\_template) |

Таблица 3.18 – Отношение m2m\_category\_attribute\_set

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| category\_id | Идентификатор категории | INT NOT NULL | FK (отношение category) |
| attribute\_set\_id | Идентификатор набора атрибутов | INT NOT NULL | FK (отношение attribute\_set) |

Таблица 3.19 – Отношение attribute\_templates

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор шаблона атрибута | INT NOT NULL | PK |
| name | Название шаблона атрибута | TEXT |  |

Таблица 3.20 – Отношение attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор атрибута | INT NOT NULL | PK |
| attribute\_set\_id | Идентификатор набора атрибутов | INT NOT NULL | FK (отношение attribute\_set) |
| product\_id | Идентификатор товара | INT NOT NULL | FK (отношение product) |
| value | Значение атрибута | VARCHAR(255) |  |

Таблица 3.21 – Отношение orders

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор заказа | INT NOT NULL | PK |
| customer\_id | Идентификатор покупателя | INT NOT NULL | FK (отношение customer) |
| sum | Сумма заказа | DECIMAL(7, 2) |  |
| created\_at | Время создания заказа | TIMESTAMP |  |

Таблица 3.22 – Отношение m2m\_order\_product

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор | INT NOT NULL | PK |
| order\_id | Идентификатор заказа | INT NOT NULL | FK (отношение order) |
| product\_id | Идентификатор товара | INT NOT NULL | FK (отношение product) |
| product\_count | Количество товаров | INT NOT NULL |  |

Таблица 3.23 – Отношение admins

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор администратора | INT NOT NULL | PK |
| email | Email-адрес администратора | VARCHAR(255) NOT NULL | UNIQUE |
| username | Имя администратора | VARCHAR(45) NOT NULL |  |
| password\_hash | Хеш пароля администратора | VARCHAR(255) NOT NULL |  |
| is\_active | Флаг активности аккаунта администратора | TINYINT |  |
| created\_at | Время создания администратора | TIMESTAMP |  |
| updated\_at | Время обновления администратора | TIMESTAMP |  |

Таблица 3.24 – Отношение configs

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назначение  атрибута | Тип данных  атрибута | Примечание |
| id | Идентификатор конфигурации | INT NOT NULL | PK |
| name | Название конфигурации | VARCHAR(45) NOT NULL |  |
| value | Значение конфигурации | VARCHAR(255) |  |

1. ОПИСАНИЕ БИЗНЕС-ЛОГИКИ

# **4.1 Описание таблиц**

Структура базы данных полностью соответствует всем вышеописанным требованиям. Данная структура представлена в приложении Б.

Таблица customers хранит информацию о покупателе. Здесь хранятся электронная почта и пароль, которые используются для аутентификации пользователя. Она связана с таблицей customer\_addresses, которая через таблицу country определяет язык, валюту, а также доступный набор товаров для данного покупателя. Для хранения информации о корзине товаров покупателя используется таблица m2m\_customer\_product, а для отображения информации о заказах служит таблица orders.

Для хранения информации о товарах используется таблица products. Поле parent\_id служит для идентификации родительского товара (например, для товара «Деревянная кружка с подписью» родительским товаром может быть товар «Деревянная кружка»). Товар может иметь различные медиа файлы, информация о которых хранится в таблице product\_galleries. Для отображении цены товара в разных валютах используется таблица product\_prices. Цена представляет собой шестизначное число с двумя знаками после запятой. Принадлежность товара к виртуальному магазину и категории определяют таблицы m2m\_store\_product и m2m\_category\_product соответственно. Таблица attributes нужна для хранения информации о значении определённого атрибута данного товара.

Информация о категориях хранится в таблице categories. Чтобы представить категории как иерархическую структуру, в таблице categories есть поле parent\_id для указания родительской категории и поле child\_count для получения информации о наличии дочерних категорий (например, категория «Верхняя одежда» имеет родительскую категорию «Мужская одежда» и дочернюю категорию «Пальто»).

Таблица attribute\_sets используется для задания всех возможных атрибутов. Она связана с таблицей categories через таблицу m2m\_category\_attribute\_set. В таблице attribute\_sets также указан язык и шаблон для данного атрибута. Для перечисления всех возможных значений всех атрибутов каждого товара используется таблица attributes.

В таблице orders хранится информация о заказах. Она связана с таблицей customers. Для сопоставления заказа и заказанных товарах используется таблица order\_products.

Таблица admins служит для хранения информации о администраторе интернет магазина. Здесь хранятся электронная почта и пароль, которые используются для аутентификации пользователя.

Таблица configs представляет собой структуру, похожую на ассоциативный массив, то есть хранит название пункта конфигурации и его значение.

# **4.2 Процедуры и функции**

Хранимая процедура – это подпрограмма, предназначенная для выполнения ряда операций с данными и структурами базы данных, хранимая на стороне базы данных и доступная как для вызова из кода других процедур и триггеров, так и для непосредственного исполнения.

Хранимая функция – это подпрограмма, расширяющая возможности языка SQL и работающая аналогично встроенным в СУБД функциям, которая обязана возвращать значение.

Использование процедур и функций имеют следующие преимущества:

* улучшается читаемость кода;
* повышается производительность;
* обеспечивается безопасность.

В качестве примера можно рассмотреть процедуру для получения всех товаров их данной категории и её дочерних категорий:

DELIMITER $$

CREATE PROCEDURE get\_products\_by\_category\_id(IN initial\_category\_id INT)

BEGIN

WITH RECURSIVE all\_categories(id) AS (

SELECT id

FROM categories

WHERE id = initial\_category\_id

UNION ALL

SELECT c.id

FROM categories c

INNER JOIN all\_categories ac ON c.parent\_id = ac.id

)

SELECT DISTINCT p.\*

FROM products p

INNER JOIN m2m\_category\_product mcp ON p.id = mcp.product\_id

WHERE mcp.category\_id IN (SELECT id FROM all\_categories);

END $$

DELIMITER ;

Процедура get\_products\_by\_category\_id рекурсивно проходит по таблице categories и находит все дочерние категории для категории с идентификатором initial\_category\_id. Далее идёт выборка товаров без повторений, которые принадлежат полученным категориям.

# **4.3 Триггеры**

Триггер – это объект базы данных, описывающий перечень действий, которые необходимо автоматически выполнить при наступлении указанного события.

Триггеров используются для решения следующих задач:

* реализация сложной логики каскадных операций;
* обновление данных кэширующих или агрегирующих полей и таблиц;
* контроль и изменение таких значений полей, которые находятся в строгой зависимости от значений других полей или иных условий.

Примером является триггер, который запрещает вставку в таблицу product\_prices отрицательного значения для поля value (то есть цены товара):

DELIMITER $$

CREATE TRIGGER check\_product\_price\_before\_insert

BEFORE INSERT ON product\_prices

FOR EACH ROW

BEGIN

IF NEW.value < 0

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Price is incorrect';

END IF;

END $$

DELIMITER ;

Триггер check\_product\_price\_before\_insert проверяется каждую новую строку, вставляемую в таблицу product\_prices, и при нахождении отрицательного значения стоимости (поле value), генерирует ошибку.

Триггер check\_product\_price\_before\_update выполняет те же действия, только срабатывает при модификации данных.

# 5. ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ

Для выявления ошибок на этапе конструирования программного средства было проведено полное тестирование. Специфики тестирования и результаты отражены в следующей таблице.

Таблица 5.1 – Тестирование базы данных

|  |  |  |
| --- | --- | --- |
| Специфика тестирования | Ожидаемый результат | Полученный результат |
| Добавление записи в таблицу | Запись добавлена | Тест пройден |
| Изменение данных записи | Обновление данных соответствующей записи | Тест пройден |
| Удаление записи из таблицы | Удаление соответствующей записи | Тест пройден |
| Осуществление выборки данных из таблицы | Отображение списка записей с указанными данными из выбранной таблицы | Тест пройден |
| Добавление записи с некорректным значением | Запись не выполнится | Тест пройден |
| Использование хранимой процедуры get\_products\_by\_category\_ id() | Отображение списка записей товаров из данной категории и её дочерних категорий | Тест пройден |
| Каскадное удаление записей после удаления записей родительской таблицы | Удаление записей, у которых стоит каскадное удаление | Тест пройден |

Проведённые тесты показали корректное выполнение всех требуемых действий. В ходе итогового тестирования не было выявлено каких-либо ошибок или некорректной работы базы данных.

# ЗАКЛЮЧЕНИЕ

В пояснительно записке я отразил все этапы разработки моего программного средства. На каждом из этих этапов я постарался показать ход моих мыслей и объяснить причины принятых решений. В результате работы над курсовым проектом была спроектирована база данных для веб-сервиса управления интернет магазином. Спроектированная база данных позволяет веб-сервису в полной мере выполнять все операции, заданные в условии курсового проекта.

При разработке базы данных использовалась система управления реляционными базами данных MySQL и среда визуального проектирования MySQL Workbench.

После анализа приведённого аналога и выявления необходимых требований, было спроектирована рассмотренная база данных. Для её создания было проведено углублённое ознакомление с предметной областью и были выбраны наиболее оптимальные решения. В ходе тестирования было показано, что база данных работает корректно относительно поставленных требований.

К сожалению, из-за кратких сроков пришлось отложить реализацию логики оплаты по частям, а также формирования купонов и скидок. Но несмотря на это, проделанная работа принесла мне много опыта, необходимого в моей будущей профессии.

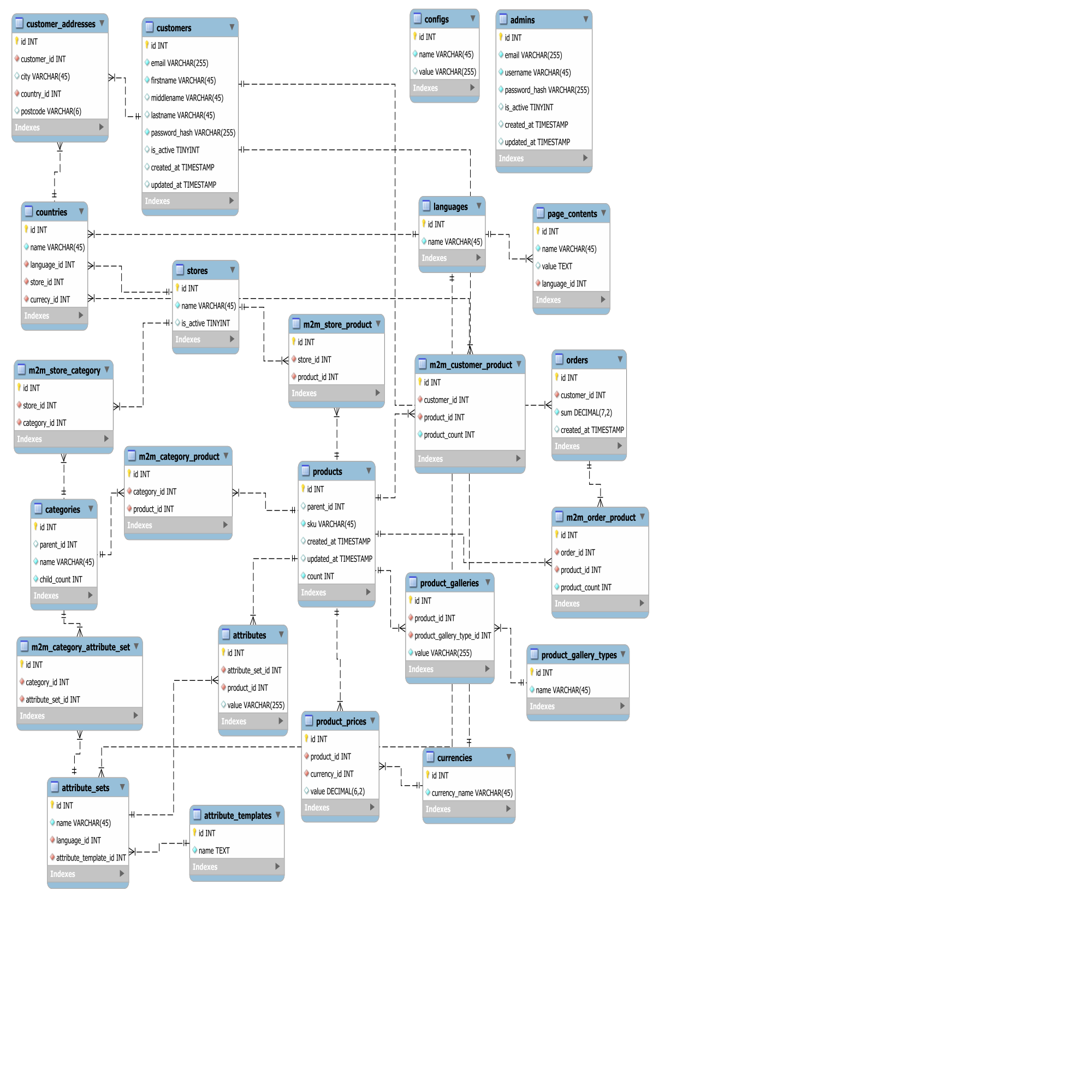
Работа была разделена на этапы, такие как анализ существующих решений, постановка требований к проектируемой базе данных, составление инфологической модели на основе полученной бизнес-логики и тестирование. После последовательного выполнения вышеперечисленных этапов разработки была получена база данных для веб-сервиса управления интернет магазином.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://business.adobe.com/products/magento/magento-commerce.html [Электронный ресурс] – Информация о системе управления интернет магазинами Magento.
2. https://dev.mysql.com/doc/ [Электронный ресурс] – Документация по реляционной системе управления базами данных MySQL.
3. https://dev.mysql.com/doc/workbench/en/ [Электронный ресурс] – Документация по инструменту для визуального проектирования базы данных MySQL Workbench.

# ПРИЛОЖЕНИЕ А

**Инфологическая модель базы данных**



# ПРИЛОЖЕНИЕ Б

**Дамп структуры базы данных**

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema cool\_shop

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema cool\_shop

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `cool\_shop` DEFAULT CHARACTER SET utf8 ;

USE `cool\_shop` ;

-- -----------------------------------------------------

-- Table `cool\_shop`.`customers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`customers` (

`id` INT NOT NULL AUTO\_INCREMENT,

`email` VARCHAR(255) NOT NULL,

`firstname` VARCHAR(45) NOT NULL,

`middlename` VARCHAR(45) NULL,

`lastname` VARCHAR(45) NULL,

`password\_hash` VARCHAR(255) NOT NULL,

`is\_active` TINYINT NULL,

`created\_at` TIMESTAMP NULL,

`updated\_at` TIMESTAMP NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE,

UNIQUE INDEX `password\_hash\_UNIQUE` (`password\_hash` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`stores`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`stores` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`is\_active` TINYINT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`currencies`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`currencies` (

`id` INT NOT NULL AUTO\_INCREMENT,

`currency\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `currency\_name\_UNIQUE` (`currency\_name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`languages`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`languages` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`countries`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`countries` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`language\_id` INT NOT NULL,

`store\_id` INT NOT NULL,

`currecy\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_store\_id\_idx` (`store\_id` ASC) VISIBLE,

INDEX `fk\_currency\_id\_idx` (`currecy\_id` ASC) VISIBLE,

INDEX `fk\_language\_id\_idx` (`language\_id` ASC) VISIBLE,

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE,

CONSTRAINT `fk\_country\_store\_id`

FOREIGN KEY (`store\_id`)

REFERENCES `cool\_shop`.`stores` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_country\_currency\_id`

FOREIGN KEY (`currecy\_id`)

REFERENCES `cool\_shop`.`currencies` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_country\_language\_id`

FOREIGN KEY (`language\_id`)

REFERENCES `cool\_shop`.`languages` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`customer\_addresses`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`customer\_addresses` (

`id` INT NOT NULL AUTO\_INCREMENT,

`customer\_id` INT NOT NULL,

`city` VARCHAR(45) NULL,

`country\_id` INT NOT NULL,

`postcode` VARCHAR(6) NULL,

PRIMARY KEY (`id`),

INDEX `fk\_country\_id\_idx` (`country\_id` ASC) VISIBLE,

INDEX `fk\_customer\_id\_idx` (`customer\_id` ASC) VISIBLE,

CONSTRAINT `fk\_customer\_address\_country\_id`

FOREIGN KEY (`country\_id`)

REFERENCES `cool\_shop`.`countries` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_customer\_address\_customer\_id`

FOREIGN KEY (`customer\_id`)

REFERENCES `cool\_shop`.`customers` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`admins`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`admins` (

`id` INT NOT NULL AUTO\_INCREMENT,

`email` VARCHAR(255) NOT NULL,

`username` VARCHAR(45) NOT NULL,

`password\_hash` VARCHAR(255) NOT NULL,

`is\_active` TINYINT NULL,

`created\_at` TIMESTAMP NULL,

`updated\_at` TIMESTAMP NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE,

UNIQUE INDEX `password\_hash\_UNIQUE` (`password\_hash` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`categories`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`categories` (

`id` INT NOT NULL AUTO\_INCREMENT,

`parent\_id` INT NULL,

`name` VARCHAR(45) NOT NULL,

`child\_count` INT NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`products`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`products` (

`id` INT NOT NULL AUTO\_INCREMENT,

`parent\_id` INT NULL,

`sku` VARCHAR(45) NOT NULL,

`created\_at` TIMESTAMP NULL,

`updated\_at` TIMESTAMP NULL,

`count` INT NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`m2m\_category\_product`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`m2m\_category\_product` (

`id` INT NOT NULL AUTO\_INCREMENT,

`category\_id` INT NOT NULL,

`product\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_category\_id\_idx` (`category\_id` ASC) VISIBLE,

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

CONSTRAINT `fk\_m2m\_category\_product\_category\_id`

FOREIGN KEY (`category\_id`)

REFERENCES `cool\_shop`.`categories` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_m2m\_category\_product\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`attribute\_templates`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`attribute\_templates` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` TEXT NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`attribute\_sets`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`attribute\_sets` (

`id` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

`language\_id` INT NOT NULL,

`attribute\_template\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_language\_id\_idx` (`language\_id` ASC) VISIBLE,

INDEX `fk\_attribute\_template\_id\_idx` (`attribute\_template\_id` ASC) VISIBLE,

CONSTRAINT `fk\_attribute\_set\_language\_id`

FOREIGN KEY (`language\_id`)

REFERENCES `cool\_shop`.`languages` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_attribute\_set\_attribute\_template\_id`

FOREIGN KEY (`attribute\_template\_id`)

REFERENCES `cool\_shop`.`attribute\_templates` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`attributes`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`attributes` (

`id` INT NOT NULL AUTO\_INCREMENT,

`attribute\_set\_id` INT NOT NULL,

`product\_id` INT NOT NULL,

`value` VARCHAR(255) NULL,

PRIMARY KEY (`id`),

INDEX `fk\_attribute\_set\_id\_idx` (`attribute\_set\_id` ASC) VISIBLE,

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

CONSTRAINT `fk\_attribute\_attribute\_set\_id`

FOREIGN KEY (`attribute\_set\_id`)

REFERENCES `cool\_shop`.`attribute\_sets` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_attribute\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`product\_gallery\_types`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`product\_gallery\_types` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`product\_galleries`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`product\_galleries` (

`id` INT NOT NULL AUTO\_INCREMENT,

`product\_id` INT NOT NULL,

`product\_gallery\_type\_id` INT NOT NULL,

`value` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

INDEX `fk\_product\_gallery\_type\_id\_idx` (`product\_gallery\_type\_id` ASC) VISIBLE,

CONSTRAINT `fk\_product\_gallery\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_product\_gallery\_product\_gallery\_type\_id`

FOREIGN KEY (`product\_gallery\_type\_id`)

REFERENCES `cool\_shop`.`product\_gallery\_types` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`m2m\_customer\_product`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`m2m\_customer\_product` (

`id` INT NOT NULL,

`customer\_id` INT NOT NULL,

`product\_id` INT NOT NULL,

`product\_count` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_customer\_id\_idx` (`customer\_id` ASC) VISIBLE,

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

CONSTRAINT `fk\_m2m\_customer\_product\_customer\_id`

FOREIGN KEY (`customer\_id`)

REFERENCES `cool\_shop`.`customers` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_m2m\_customer\_product\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`m2m\_store\_category`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`m2m\_store\_category` (

`id` INT NOT NULL AUTO\_INCREMENT,

`store\_id` INT NOT NULL,

`category\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_store\_id\_idx` (`store\_id` ASC) VISIBLE,

INDEX `fk\_category\_id\_idx` (`category\_id` ASC) VISIBLE,

CONSTRAINT `fk\_m2m\_store\_category\_store\_id`

FOREIGN KEY (`store\_id`)

REFERENCES `cool\_shop`.`stores` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_m2m\_store\_category\_category\_id`

FOREIGN KEY (`category\_id`)

REFERENCES `cool\_shop`.`categories` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`m2m\_store\_product`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`m2m\_store\_product` (

`id` INT NOT NULL AUTO\_INCREMENT,

`store\_id` INT NOT NULL,

`product\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_store\_id\_idx` (`store\_id` ASC) VISIBLE,

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

CONSTRAINT `fk\_m2m\_store\_product\_store\_id`

FOREIGN KEY (`store\_id`)

REFERENCES `cool\_shop`.`stores` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_m2m\_store\_product\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`product\_prices`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`product\_prices` (

`id` INT NOT NULL AUTO\_INCREMENT,

`product\_id` INT NOT NULL,

`currency\_id` INT NOT NULL,

`value` DECIMAL(6,2) NULL,

PRIMARY KEY (`id`),

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

INDEX `fk\_currency\_id\_idx` (`currency\_id` ASC) VISIBLE,

CONSTRAINT `fk\_product\_price\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_product\_price\_currency\_id`

FOREIGN KEY (`currency\_id`)

REFERENCES `cool\_shop`.`currencies` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`page\_contents`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`page\_contents` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`value` TEXT NULL,

`language\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_language\_id\_idx` (`language\_id` ASC) VISIBLE,

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE,

CONSTRAINT `fk\_page\_content\_language\_id`

FOREIGN KEY (`language\_id`)

REFERENCES `cool\_shop`.`languages` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`configs`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`configs` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`value` VARCHAR(255) NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`orders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`orders` (

`id` INT NOT NULL AUTO\_INCREMENT,

`customer\_id` INT NOT NULL,

`sum` DECIMAL(7,2) NOT NULL,

`created\_at` TIMESTAMP NULL,

PRIMARY KEY (`id`),

INDEX `fk\_customer\_id\_idx` (`customer\_id` ASC) VISIBLE,

CONSTRAINT `fk\_order\_customer\_id`

FOREIGN KEY (`customer\_id`)

REFERENCES `cool\_shop`.`customers` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`m2m\_order\_product`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`m2m\_order\_product` (

`id` INT NOT NULL AUTO\_INCREMENT,

`order\_id` INT NOT NULL,

`product\_id` INT NOT NULL,

`product\_count` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_order\_id\_idx` (`order\_id` ASC) VISIBLE,

INDEX `fk\_product\_id\_idx` (`product\_id` ASC) VISIBLE,

CONSTRAINT `fk\_m2m\_order\_product\_order\_id`

FOREIGN KEY (`order\_id`)

REFERENCES `cool\_shop`.`orders` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_m2m\_order\_product\_product\_id`

FOREIGN KEY (`product\_id`)

REFERENCES `cool\_shop`.`products` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `cool\_shop`.`m2m\_category\_attribute\_set`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `cool\_shop`.`m2m\_category\_attribute\_set` (

`id` INT NOT NULL AUTO\_INCREMENT,

`category\_id` INT NOT NULL,

`attribute\_set\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_category\_id\_idx` (`category\_id` ASC) VISIBLE,

INDEX `fk\_attribute\_set\_id\_idx` (`attribute\_set\_id` ASC) VISIBLE,

CONSTRAINT `fk\_m2m\_category\_attribute\_set\_category\_id`

FOREIGN KEY (`category\_id`)

REFERENCES `cool\_shop`.`categories` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_m2m\_category\_attribute\_set\_attribute\_set\_id`

FOREIGN KEY (`attribute\_set\_id`)

REFERENCES `cool\_shop`.`attribute\_sets` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

DROP PROCEDURE IF EXISTS get\_products\_by\_category\_id;

DELIMITER $$

CREATE PROCEDURE get\_products\_by\_category\_id(IN initial\_category\_id INT)

BEGIN

WITH RECURSIVE all\_categories(id) AS (

SELECT id

FROM categories

WHERE id = initial\_category\_id

UNION ALL

SELECT c.id

FROM categories c

INNER JOIN all\_categories ac ON c.parent\_id = ac.id

)

SELECT DISTINCT p.\*

FROM products p

INNER JOIN m2m\_category\_product mcp ON p.id = mcp.product\_id

WHERE mcp.category\_id IN (SELECT id FROM all\_categories);

END $$

DELIMITER ;

DROP TRIGGER IF EXISTS check\_product\_price\_before\_insert;

DELIMITER $$

CREATE TRIGGER check\_product\_price\_before\_insert

BEFORE INSERT ON product\_prices

FOR EACH ROW

BEGIN

IF NEW.value < 0

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Price is incorrect';

END IF;

END $$

DELIMITER ;

DROP TRIGGER IF EXISTS check\_product\_price\_before\_update;

DELIMITER $$

CREATE TRIGGER check\_product\_price\_before\_update

BEFORE UPDATE ON product\_prices

FOR EACH ROW

BEGIN

IF NEW.value < 0

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Price is incorrect';

END IF;

END $$

DELIMITER ;