База данных для интернет-магазина

Она будет предназначена для эффективного управления информацией о продуктах, заказах, клиентах, платежах, доставке и других аспектах розничной торговли в онлайн-среде. Бд необходима, чтобы хранить, организовывать и обрабатывать большое количество данных, связанных с продажами и обслуживанием клиентов.

Данная база данных будет обеспечивать возможность отслеживания состояния складских запасов, автоматизации процесса обработки заказов, хранения и управления информацией о клиентах и их заказах, анализа эффективности маркетинговых акций, отслеживания платежей и доставок, а также получения ценной аналитической информации для принятия управленческих решений.

Благодаря базе данных интернет-магазин может эффективно управлять своей деятельностью, обеспечивать качественное обслуживание клиентов, оптимизировать процессы закупок и поставок, а также проводить анализ торговых операций для улучшения бизнес-процессов.

База данных должна обеспечивать:

1. Управление товарами:

* Добавление, редактирование и удаление информации о продуктах.
* Возможность указания атрибутов товара, таких как название, описание, цена, наличие на складе, категория и изображения.

2. Управление заказами:

* Возможность просмотра и обработки оформленных заказов.
* Отслеживание статусов заказов (обработка, оплата, доставка).
* Создание отчетов о выполненных заказах.

3. Управление клиентской информацией:

* Хранение данных о клиентах, их личной информации, предпочтениях, истории заказов.
* Возможность регистрации новых клиентов.

4. Управление доставкой и оплатой:

* Хранение информации о способах доставки и оплаты.
* Возможность расчета стоимости доставки и работы с различными службами доставки.

5. Управление отзывами и оценками:

* Возможность просмотра, добавления и управления отзывами о товарах.
* Рейтинги и обзоры товаров от клиентов.

Также выдвигаются ограничения:

* Каждый товар должен иметь уникальный идентификатор, название не превышающее 100 символов, описание длиной не более 1000 символов, цену >= 0, категории могут иметь совпадающие названия.
* Каждый заказ должен содержать уникальный идентификатор, дату оформления, статус (например, "обработка", "доставка", "выполнен"), идентификатор клиента, общую сумму заказа >= 0.
* Каждый клиент должен иметь уникальный идентификатор, имя и фамилию не длиннее 50 символов, адрес доставки длиной не более 200 символов, контактную информацию (email, телефон) валидную и актуальную.
* Информация о доставке должна содержать уникальный идентификатор, способ доставки, дату доставки; информация об оплате должна включать способ оплаты, сумму оплаты >= 0, статус оплаты (например, "оплачено", "не оплачено").

Для создания Er-диаграммы я выделил следующие сущности:

Товар (ID, Наименование, Цена, Описание, Наличие на складе, Производитель)

Категория (ID, Наименование)

Заказ (ID, Дата заказа, Статус, Сумма)

Клиент (ID, Имя, Фамилия, Адрес, Контактная информация)

Доставка (ID, Дата доставки, Способ, Стоимость, Адрес, Статус)

Отзыв (ID, Оценка, Комментарий, Дата)

Связи:

Товары в Категориях (Многие-к-одному)

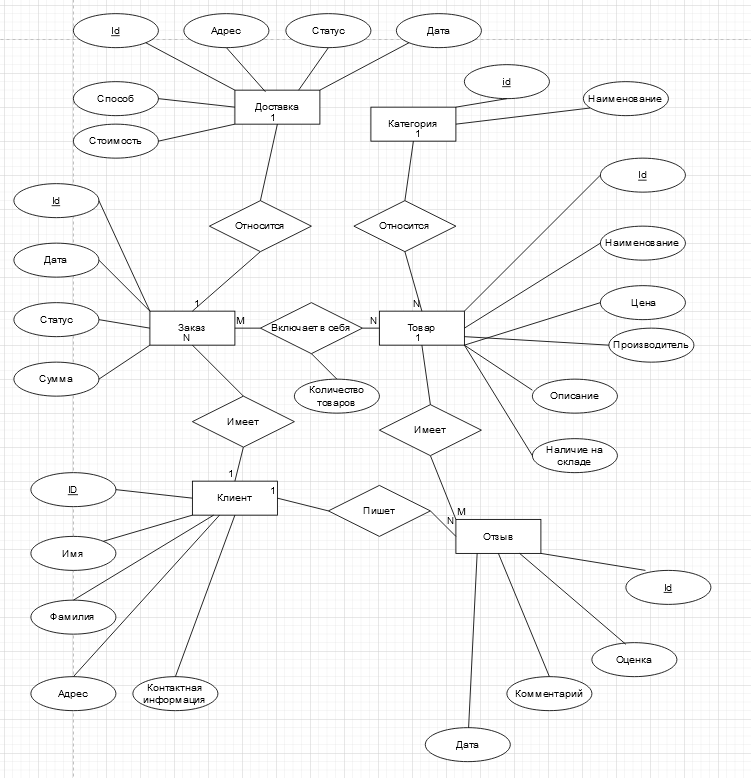
Заказы содержат Товары (Многие-ко-многим)

Заказы имеют Клиентов (Многие-к-одному)

Заказы связаны с Доставкой (Один-к-одному)

Отзывы относятся к Товарам (Многие-к-одному)

Отзывы написаны Клиентами (Многие-к-одному)

Er-диаграмма: 

Также в репозитории есть версия в расширении .drawio.

Далее предварительно составлю схему таблиц:

* Товары (ID товара (PRIMARY KEY), Наименование, Цена, Описание, Наличие на складе, Производитель, ID категории (FOREIGN KEY))
* Категории (ID категории (PRIMARY KEY), Наименование категории)
* Заказы (ID заказа (PRIMARY KEY), Дата оформления, Статус, Сумма, ID клиента (FOREIGN KEY), ID доставки (FOREIGN KEY))
* Клиенты (ID клиента (PRIMARY KEY), Имя, Фамилия, Адрес, Контактная информация)
* Доставки (ID доставки (PRIMARY KEY), Дата доставки, Способ доставки, Стоимость доставки, Адрес доставки, Статус доставки)
* Отзывы (ID отзыва (PRIMARY KEY), Оценка, Комментарий, Дата публикации, ID товара (FOREIGN KEY), ID клиента (FOREIGN KEY))
* Товары\_с\_заказом (Заказ\_ID, Товар\_ID, Количество)

Теперь опишу функциональные зависимости:

1. Таблица "Товары":

- ID товара → Наименование, Цена, Описание, Наличие на складе, Производитель, ID категории

- ID категории → Наименование категории (внешний ключ)

2. Таблица "Категории":

- ID категории → Наименование категории

3. Таблица "Заказы":

- ID заказа → Дата оформления, Статус, Сумма, ID клиента, ID доставки

- ID клиента → Имя, Фамилия, Адрес, Контактная информация (внешний ключ)

- ID доставки → Дата доставки, Способ доставки, Стоимость доставки, Адрес доставки, Статус доставки (внешний ключ)

4. Таблица "Клиенты":

- ID клиента → Имя, Фамилия, Адрес, Контактная информация

5. Таблица "Доставка":

- ID доставки → Дата доставки, Способ доставки, Стоимость доставки, Адрес доставки, Статус доставки

6. Таблица "Отзывы":

- ID отзыва → Оценка, Комментарий, Дата публикации, ID товара, ID клиента

- ID товара → Наименование, Цена, Описание, Наличие на складе, Производитель (внешний ключ)

- ID клиента → Имя, Фамилия, Адрес, Контактная информация (внешний ключ)

7. Таблица "Товары с заказом":

- Заказ\_ID, Товар\_ID → Количество

Схема соответствует 3 нормальной форме. Нет транзитивных зависимостей.

Далее напишу скрипт на sql ddl, создающий таблицы (в репозитории также есть файл shema\_ddl.sql):

CREATE TABLE Товары (

ID SERIAL PRIMARY KEY,

Наименование VARCHAR(255) NOT NULL,

Цена NUMERIC(10, 2) NOT NULL CHECK (Цена > 0),

Описание TEXT,

Наличие\_на\_складе INTEGER CHECK (Наличие\_на\_складе >= 0),

Производитель VARCHAR(255),

Категория\_ID INTEGER

);

CREATE TABLE Категории (

ID SERIAL PRIMARY KEY,

Наименование VARCHAR(255) NOT NULL

);

CREATE TABLE Заказы (

ID SERIAL PRIMARY KEY,

Дата\_оформления DATE NOT NULL,

Статус VARCHAR(50) NOT NULL,

Сумма NUMERIC(10, 2) NOT NULL,

Клиент\_ID INTEGER,

Доставка\_ID INTEGER,

ID\_товара INTEGER

);

CREATE TABLE Клиенты (

ID SERIAL PRIMARY KEY,

Имя VARCHAR(50) NOT NULL,

Фамилия VARCHAR(50) NOT NULL,

Адрес TEXT,

Контактная\_информация TEXT

);

CREATE TABLE Доставка (

ID SERIAL PRIMARY KEY,

Дата\_доставки DATE NOT NULL,

Способ\_доставки VARCHAR(100) NOT NULL,

Стоимость\_доставки NUMERIC(10, 2) NOT NULL CHECK (Стоимость\_доставки > 0),

Адрес\_доставки TEXT,

Статус\_доставки VARCHAR(50) NOT NULL

);

CREATE TABLE Отзывы (

ID SERIAL PRIMARY KEY,

Оценка INTEGER NOT NULL,

Комментарий TEXT,

Дата DATE NOT NULL,

Товар\_ID INTEGER,

Клиент\_ID INTEGER

);

CREATE TABLE Товары\_в\_заказе (

Заказ\_ID INTEGER,

Товар\_ID INTEGER,

Количество INTEGER,

FOREIGN KEY (Заказ\_ID) REFERENCES Заказы(ID),

FOREIGN KEY (Товар\_ID) REFERENCES Товары(ID)

);

ALTER TABLE Товары ADD FOREIGN KEY (Категория\_ID) REFERENCES Категории(ID);

ALTER TABLE Заказы ADD FOREIGN KEY (Клиент\_ID) REFERENCES Клиенты(ID);

ALTER TABLE Заказы ADD FOREIGN KEY (Доставка\_ID) REFERENCES Доставка(ID);

ALTER TABLE Отзывы ADD FOREIGN KEY (Товар\_ID) REFERENCES Товары(ID);

ALTER TABLE Отзывы ADD FOREIGN KEY (Клиент\_ID) REFERENCES Клиенты(ID);

Опишу теперь возможные запросы к базе данных, исходя из требований:

1. Вывести количество клиентов, оплативших заказ и количество ожидающих оплаты на текущий момент:

SELECT

(SELECT COUNT(DISTINCT Клиент\_ID) FROM Заказы WHERE Статус = 'оплачено') AS Количество\_оплаченных\_заказов,

(SELECT COUNT(DISTINCT Клиент\_ID) FROM Заказы WHERE Статус = 'ожидает оплаты') AS Количество\_ожидающих\_оплаты;

2. Вывести информацию о продуктах, которые есть в наличии на складе:

SELECT Наименование, Цена

FROM Товары

WHERE Наличие\_на\_складе > 0;

3. Вывести сумму ожидаемых доходов за определенный период времени, например, за последний месяц:

SELECT

SUM(Сумма) AS Ожидаемый\_доход\_за\_месяц

FROM Заказы

WHERE Дата\_оформления >= CURRENT\_DATE - INTERVAL '1 month';

4. Вывести средний рейтинг отзывов по каждому товару:

SELECT

Товар\_ID, AVG(Оценка) AS Средний\_рейтинг

FROM Отзывы

GROUP BY Товар\_ID;

5. Вывести список клиентов и дату последнего оформленного ими заказа:

SELECT

Клиенты.ID, Клиенты.Имя, Клиенты.Фамилия, MAX(Заказы.Дата\_оформления) AS Последний\_заказ

FROM

Клиенты

LEFT JOIN

Заказы ON Клиенты.ID = Заказы.Клиент\_ID

GROUP BY

Клиенты.ID, Клиенты.Имя, Клиенты.Фамилия;

6. Вывести список товаров, отсортированный по цене по убыванию:

SELECT

Наименование, Цена

FROM

Товары

ORDER BY Цена DESC;

7. Вывести ранжированный список товаров по их цене в пределах каждой категории

SELECT ID, Наименование, Цена, Категория\_ID,

RANK() OVER (PARTITION BY Категория\_ID ORDER BY Цена) AS Ранжирование\_по\_цене

FROM Товары; SELECT

ID,

Наименование,

Цена,

Категория\_ID,

RANK() OVER (PARTITION BY Категория\_ID ORDER BY Цена) AS Ранжирование\_по\_цене

FROM Товары;

Группировка запросов выше в транзакции будет логически неверным. Поэтому приведу пример еще запросов:

BEGIN;

UPDATE Заказы

SET Статус = 'доставлен'

WHERE ID = :id\_заказа;

UPDATE Доставка

SET Статус\_доставки = 'доставлено'

WHERE ID = :id\_доставки;

COMMIT;

В этой транзакции допустим, статус заказа изменился с ‘не доставлен, оплачен’ на ‘доставлен, оплачен’, а статус доставки изменился с ’В пути в грузовике’ на ‘доставлено’. Эти запросы правильно выполнять, используя транзакцию, чтобы обеспечить целостность данных.

Приведу пример проблем из-за недонормализованности.

Объединим таблицы товары и категории:

CREATE TABLE Товары\_и\_Категории (

ID SERIAL PRIMARY KEY,

Наименование VARCHAR(255) NOT NULL,

Цена NUMERIC(10, 2) NOT NULL,

Описание TEXT,

Наличие\_на\_складе INTEGER,

Производитель VARCHAR(255),

Категория\_ID INTEGER,

Наименование\_категории VARCHAR(255)

);

Теперь появилась функциональная зависимость Категория\_ID → Наименование\_категории, где нарушается 3 нормальная форма. (Храним id ввиду не уникальности наименования категории)

Такой запрос из-за наличия новой зависимости может привести к ошибкам, например, когда имелось ввиду поменять полностью категорию для товара, т.е. и id и ее название. Он может вызвать нарушение целостности данных. Также при необходимости поменять название категории, придется менять все данные в таблице, где встречался id этой категории:

UPDATE Товары\_и\_Категории

SET Наименование\_категории = 'Техника'

WHERE ID = 1;