Тема: Разработка системы управления запасами на складе

Задание: Разработка базы данных для управления запасами на складе.

**Цель задания:**Разработать базу данных для управления запасами на складе, которая будет включать несколько таблиц, поддерживать операции с данными, индексацию, транзакции, функции, процедуры и триггеры. База данных должна обеспечивать целостность данных и поддержку различных операций с ними.

Часть 1: Управление таблицами

Создание таблиц

Создайте следующие таблицы:

* products (товары)
* warehouses (склады)
* suppliers (поставщики)
* stock (инвентарь)
* orders (заказы)
* order\_items (позиции заказов)

Описание таблиц

**Таблица products**

* product\_id (INTEGER, PRIMARY KEY) — уникальный идентификатор товара.
* product\_name (TEXT) — название товара.
* product\_description (TEXT) — описание товара.
* category (TEXT) — категория товара (например, "электроника", "одежда").
* unit\_price (NUMERIC) — цена за единицу товара.
* created\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время создания записи о товаре.
* updated\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время последнего обновления записи о товаре.

**Таблица warehouses**

* warehouse\_id (INTEGER, PRIMARY KEY) — уникальный идентификатор склада.
* warehouse\_name (TEXT) — название склада.
* location (TEXT) — местоположение склада.
* capacity (INTEGER) — вместимость склада (количество единиц товара).
* created\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время создания записи о складе.
* updated\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время последнего обновления записи о складе.

**Таблица suppliers**

* supplier\_id (INTEGER, PRIMARY KEY) — уникальный идентификатор поставщика.
* supplier\_name (TEXT) — название поставщика.
* contact\_person (TEXT) — контактное лицо.
* phone\_number (TEXT) — номер телефона.
* email (TEXT) — электронная почта.
* created\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время создания записи о поставщике.
* updated\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время последнего обновления записи о поставщике.

**Таблица stock**

* stock\_id (INTEGER, PRIMARY KEY) — уникальный идентификатор записи инвентаря.
* product\_id (INTEGER, FOREIGN KEY) — внешний ключ, ссылающийся на таблицу products.
* warehouse\_id (INTEGER, FOREIGN KEY) — внешний ключ, ссылающийся на таблицу warehouses.
* quantity (INTEGER) — количество товара на складе.
* last\_restocked (DATE) — дата последнего пополнения запасов.
* created\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время создания записи.
* updated\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время последнего обновления записи.

**Таблица orders**

* order\_id (INTEGER, PRIMARY KEY) — уникальный идентификатор заказа.
* order\_date (DATE) — дата заказа.
* supplier\_id (INTEGER, FOREIGN KEY) — внешний ключ, ссылающийся на таблицу suppliers.
* total\_amount (NUMERIC) — общая сумма заказа.
* status (TEXT) — статус заказа (например, "в обработке", "доставлен", "отменен").
* created\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время создания заказа.
* updated\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время последнего обновления заказа.

**Таблица order\_items**

* order\_item\_id (INTEGER, PRIMARY KEY) — уникальный идентификатор позиции заказа.
* order\_id (INTEGER, FOREIGN KEY) — внешний ключ, ссылающийся на таблицу orders.
* product\_id (INTEGER, FOREIGN KEY) — внешний ключ, ссылающийся на таблицу products.
* quantity (INTEGER) — количество товара в заказе.
* unit\_price (NUMERIC) — цена за единицу товара.
* total\_price (NUMERIC) — общая стоимость позиции заказа.
* created\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время создания записи.
* updated\_at (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP) — дата и время последнего обновления записи.

Часть 2: Работа с данными

Добавление данных

Заполните таблицы следующими данными:

* Добавьте минимум 50 поставщиков.
* Добавьте минимум 30 складов.
* Добавьте минимум 100 товаров.
* Добавьте минимум 50 заказов с позициями.

Запросы к данным

Напишите запросы для:

1. Выбора всех товаров, которые находятся на определенном складе.
2. Выбора всех заказов, которые были доставлены.
3. Вывода всех товаров с их количеством на каждом складе.
4. Вывода заказов, которые были отменены.

Часть 3: Индексы

Создание индексов

1. Создайте индекс на поле product\_id в таблице stock для ускорения запросов, связанных с поиском товаров.
2. Создайте индекс на поле warehouse\_id в таблице stock для ускорения поиска товаров по складу.

Часть 4: Транзакции

Реализация транзакций

Реализуйте транзакцию для добавления нового товара и его размещения на складе:

1. Добавьте товар в таблицу products.
2. Добавьте запись о товаре в таблицу stock. Если хотя бы одна из операций неудачна, транзакция должна быть отменена.

Напишите запрос с использованием BEGIN, COMMIT и ROLLBACK для контроля транзакций.

Часть 5: Функции (PL/pgSQL)

1. **Функция для подсчета среднего количества товаров на складе:**Напишите функцию, которая будет принимать warehouse\_id и возвращать среднее количество товаров на складе.
2. **Функция для обновления информации о товаре:**Напишите функцию для обновления цены товара по product\_id. Функция должна проверять, существует ли товар.
3. **Функция для подсчета количества заказов на поставщика:**Напишите функцию, которая будет принимать supplier\_id и возвращать количество заказов, сделанных у этого поставщика.

Часть 6: Процедуры

1. **Процедура для обновления статуса заказа:**Напишите процедуру, которая будет обновлять статус заказа (например, "в обработке", "доставлен", "отменен") в зависимости от даты заказа.
2. **Процедура для массового обновления цен:**Напишите процедуру, которая будет увеличивать цену всех товаров в определенной категории на 10%.

Часть 7: Триггеры

1. **Триггер для автоматического обновления даты последнего обновления товара:**Напишите триггер, который будет автоматически обновлять поле updated\_at в таблице products, каждый раз когда запись изменяется.
2. **Триггер для проверки уникальности записи:**Напишите триггер, который будет проверять, что один и тот же товар не может быть добавлен на склад дважды.

Часть 8: Дополнительные темы

1. **Использование RAISE для обработки ошибок:**Используйте RAISE EXCEPTION в одной из функций или процедур для вывода ошибок, если данные не удовлетворяют определенным критериям (например, если товар уже существует на складе).
2. **DO-блок для выполнения анонимного кода:**Напишите DO-блок, который будет создавать новый товар и добавлять его на склад.

Часть 9: CTE (Common Table Expressions)

Используйте CTE для выбора всех товаров, которые находятся на складе с количеством меньше 10.

Часть 10: Оконные функции

Используйте оконную функцию для расчета ранга каждого товара по количеству на складе среди всех товаров.

Часть 11: Продвинутые индексы

Создайте индекс GIN для оптимизации поиска по текстовым данным в таблице products.

Часть 12: Полнотекстовый поиск

Используйте полнотекстовый поиск для анализа описания товаров и поиска товаров по ключевым словам в их описаниях.

Часть 13: Партиционирование таблиц

1. **Партиционирование по диапазону (Range Partitioning):**Реализуйте партиционирование таблицы orders по диапазону дат для улучшения производительности запросов по времени.
2. **Партиционирование по списку (List Partitioning):**Реализуйте партиционирование таблицы products по категориям с использованием списка значений.
3. **Партиционирование по хэшированию (Hash Partitioning):**Реализуйте партиционирование таблицы stock по хэшированию для равномерного распределения данных.

Часть 14: Временные таблицы

1. **Использование временных таблиц (TEMP):**Создайте временную таблицу для хранения промежуточных результатов выполнения сложных запросов, например, товаров с низким запасом.

Часть 15: Оптимизация UPDATE в PostgreSQL

1. **Использование HOT (Heap-Only Tuple) Updates:**Напишите запрос для выполнения обновления записи в таблице products, которое должно быть выполнено с использованием HOT обновлений.

Часть 16: Уровни изоляции и проблемы аномалий

1. **Пример транзакции с уровнем изоляции SERIALIZABLE:**Напишите пример транзакции, которая использует уровень изоляции SERIALIZABLE, и объясните, как он предотвращает аномалии.

Часть 17: Кастомные агрегатные функции

1. **Создание кастомной агрегатной функции:**Напишите свою кастомную агрегатную функцию в PL/pgSQL, которая будет рассчитывать медиану количества товаров на складе.
2. **Пример использования агрегатной функции:**Вычисление медианы для количества товаров из таблицы stock.
3. **Создание агрегатных функций для сложных вычислений:**Создайте кастомную агрегатную функцию для подсчета средней цены товаров, но с дополнительной логикой (например, если товар находится в категории "электроника", его цена увеличивается на 15%).

Часть 18: Кастомные операторы в PostgreSQL

1. **Создание кастомного оператора:**Напишите кастомный оператор для выполнения операции "слияния строк", который будет объединять название товара и его категорию в одну строку.
2. **Использование кастомного оператора в запросах:**Напишите запрос, который использует созданный кастомный оператор для формирования информации о товаре и его категории.

Часть 19: Пользовательские типы данных

1. **Создание пользовательского типа данных:**Создайте новый тип данных product\_status, который будет содержать поля для различных статусов товара (например, "в наличии", "нет в наличии", "под заказ").
2. **Использование пользовательского типа данных в запросах:**Напишите запрос, который извлекает товары с определенным статусом из таблицы products, используя пользовательский тип данных.

Часть 20: Sequences в PostgreSQL

1. **Создание и использование Sequence:**Создайте sequence для генерации уникальных идентификаторов товаров в таблице products. Напишите запрос, который автоматически присваивает уникальный ID товарам при добавлении новых записей.
2. **Использование Sequence в запросах:**Напишите запрос, который будет вставлять новые записи в таблицу products с использованием значения из sequence.

Часть 21: Правила (Rules) в PostgreSQL

1. **Создание правила (Rule) для автоматического обновления:**Создайте правило, которое автоматически обновляет поле updated\_at в таблице products каждый раз, когда товар изменяется.
2. **Использование правил для автоматических операций:**Напишите правило, которое будет автоматически удалять запись из таблицы stock, если количество товара на складе меньше 0.

Часть 22: Collations в PostgreSQL

1. **Использование Collations для сортировки данных:**Напишите запрос, который сортирует товары по названию в зависимости от локали (например, для английской или русской локали).

Часть 23: Коэффициент корреляции

1. **Вычисление коэффициента корреляции:**Напишите запрос, который вычисляет коэффициент корреляции между ценой товара и количеством на складе для различных категорий товаров с использованием функции CORR().

Часть 24: Casts в PostgreSQL

1. **Использование кастов (Casts) для преобразования типов данных:**Напишите запрос, который преобразует строку с датой (например, "2023-05-10") в тип DATE. Используйте функцию кастинга для преобразования данных из одного типа в другой в контексте таблиц orders и order\_items.
2. **Создание кастов для пользовательских типов данных:**Напишите каст для преобразования пользовательского типа product\_status в строковое представление (например, формат "в наличии", "нет в наличии").

Часть 25: Дополнительные задачи

1. **Обобщенная задача по оптимизации запросов:**Используйте методы оптимизации запросов (например, создание индексов, использование EXPLAIN, оптимизация JOIN'ов и т.д.), чтобы ускорить выполнение запроса на выборку товаров, которые находятся на складе с количеством меньше 10.
2. **Использование временных таблиц и UNLOGGED таблиц:**Создайте временную таблицу для хранения промежуточных данных о товарах с низким запасом, и используйте UNLOGGED таблицу для более быстрой обработки этих данных.

Часть 26: Представления (Views)

1. **Создание представления:**Создайте представление warehouse\_summary, которое будет показывать список товаров для каждого склада с их количеством и статусом.