**Тестовое задание для VK. Тимур Гимадиев.**

**База данных.**

Для реализации интернет-магазина с перечисленными требованиями, я бы предложил использовать гибридный подход к выбору баз данных, комбинируя реляционные и документоориентированные (NoSQL) базы данных.

Среди реляционных БД использовал PostgreSQL, так как это надежная (удовлетворяет ACID), актуальная и достаточно простая в использовании система. Будем использовать ее для хранения структурированных данных, которые описываются четкой схемой и требуют строгих связей.   
  
Что будет храниться в Postgree:

- Каталог товаров и их категории.

- Информация о пользователях.

- Информация о складах и наличии товаров.

- Управление заказами и корзиной покупателей.

Примеры таблиц:

**Users:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User\_id (первичный ключ)** | **name** | **hash\_of\_password** | **email** | **city\_name** | **adress** |
| 1 | Тимур | 3v4h2nkf | Tmrgmdv2004@gmail.com | Казань | ул. Петербургская, 52 |
| 2 | Павел | Nf40nj32 | durov@vk.com | Дубай | Бульвар Шейха Мохаммеда бин Рашида, 1 |

city\_name эти 2 таблицы я предлагаю использовать индексирование.

**Warehouses:**

|  |  |  |
| --- | --- | --- |
| **warehouse\_id**  **(первичный ключ)** | **city\_name** | **warehouse\_name** |
| 1 | Казань | Склад\_1 |
| 2 | Москва | Склад\_2 |

Чтобы быстро найти id склада, можно использовать индексирование по столбцу city\_name.

**Categories:**

|  |  |  |
| --- | --- | --- |
| **category\_id**  **(первичный ключ)** | **category\_name** | **parent\_id**  **(references Categories.category\_id)** |
| 1 | Мужчинам | NULL |
| 2 | Женщинам | NULL |
| 3 | Одежда | 1 |
| 4 | Обувь | 1 |
| 5 | Украшения | 2 |
| 6 | Джемперы | 3 |
| 7 | Кеды | 4 |
| 8 | Бусы | 5 |

В таблице с категориями товаров используются самоссылающиеся строки. Это удобно, когда есть древовидная структура данных.

**Products:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **product\_id**  **(первичный ключ)** | **category\_id**  **(references Categories.category\_id)** | **product\_name** | **inventory\_id**  **(references Inventory.inventory\_id)** | **object\_storage**  **(refers to Object Storage)** |
| 1 | 6 | Джемпер мужской синий р44 | [1,2] | 1 |

Мы можем за константное время (3 операции) узнать к какой категории принадлежит определенный товар.

**Inventory (наличие на складах):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **inventory\_id**  **(первичный ключ)** | **product\_id**  **(references Products.product\_id)** | **warehouse\_id**  **(references Warehouses.warehouse\_id)** | **quantity** | **price** |
| 1 | 1 | 1 | 10 | 4000 |
| 2 | 1 | 2 | 20 | 4500 |

**Orders:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **order\_id (первичный ключ)** | **user\_id**  **(references Users.user\_id)** | **status** | **total\_price** | **created\_at** |
| 1 | 1 | В пути | 4200 | 01.06.2024 |

Перейдем к NoSQL БД. Так как данные о товаре могут быть структурированы не всегда по четким правилам и где требуется высокая гибкость и масштабируемость, будем использовать документоориентированную БД. Я слышал, что в PostgreSQL тоже можно хранить json-объекты, но до конца не уверен. Предлагаю использовать MongoDB, как один из самых популярных вариантов.

**ProductDetails:**

{

\_id

fields (ключ-значение пары для динамических характеристик: вес, длина, тип обложки и т.д.)  
}

Для хранения всего медиаконтента (фотографии, видео) вместо стандарной БД лучше использовать Object Storage, предназначенной для хранения файлов больших размеров.

**Схема**

Прототип API

**interface IUserRegistration {**

**public function register(string $email, string $password, array $additionalInfo): bool;}**

Возвращает true, если регистрация прошла успешно,

additionalInfo - Дополнительная информация о пользователе (например, имя, адрес и т. д.)

**interface IUserAuthentication {**

// Авторизация пользователя

//return string - Возвращает токен сессии, если авторизация прошла успешно

**public function login(string $email, string $password): string;**

// Выход из системы

// string $token - Токен текущей сессии пользователя

return bool - Возвращает true, если выход прошел успешно

**public function logout(string $token): bool;**

}

**interface ICatalogSearch {**

//Поиск товаров по запросу

// string $query - Поисковый запрос

// return array - Возвращает массив товаров, соответствующих запросу

**public function search(string $query): array;**

}

**interface ICatalogFilter {**

// Фильтрация товаров по параметрам

// array $filterParams - Массив параметров фильтрации (например, категория, цена и т. д.)

// return array - Возвращает массив отфильтрованных товаров

**public function filter(array $filterParams): array;**

}

**interface ICatalogSort** {

// Сортировка товаров по заданному критерию

// string $sortBy - Критерий сортировки (например, цена, популярность и т. д.)

// array - Возвращает массив отсортированных товаров

**public function sort(string $sortBy): array;**

**public function getItemByCategory(string $group, string $category, string $subcategory): array;**

// Получение деталей товара по его идентификатору

// int $itemId - Идентификатор товара

// return array - Во звращает массив с данными о товаре

**public function getItemDetails(int $itemId): array;**

}

// Классы для управления каталогом товаров

**class CatalogManagement implements ICatalogSearch, ICatalogFilter, ICatalogSort, ICatalogRetrieve {**

**public function search(string $query): array** {

// Реализация поиска товаров

}

**public function filter(array $filterParams): array** {

// Реализация фильтрации товаров

}

**public function sort(string $sortBy): array** {

// Реализация сортировки товаров

}

**public function getItemByCategory(string $group, string $category, string $subcategory): array** {

// Реализация получения товаров по категории

}

**public function getItemDetails(int $itemId): array** {

// Реализация получения деталей товара

}

}

// Интерфейс для управления корзиной

**interface ICartManagement {**

Добавление товара в корзину

//int $userId - Идентификатор пользователя

// int $itemId - Идентификатор товара

// int $quantity - Количество товара

// bool - Возвращает true, если товар успешно добавлен в корзину

**public function addItem(int $userId, int $itemId, int $quantity): bool;**

Удаление товара из корзины

// int $userId - Идентификатор пользователя

// int $itemId - Идентификатор товара

//return bool - Возвращает true, если товар успешно удален из корзины

**public function removeItem(int $userId, int $itemId): bool;**

Получение текущей корзины пользователя

//int $userId - Идентификатор пользователя

//array - Возвращает массив товаров в корзине

**public function getCart(int $userId): array;**

}

// Класс для управления корзиной

**class CartManagement implements ICartManagement {**

**public function addItem(int $userId, int $itemId, int $quantity): bool {**

// Реализация добавления товара в корзину

}

**public function removeItem(int $userId, int $itemId): bool {**

// Реализация удаления товара из корзины

}

**public function getCart(int $userId): array {**

// Реализация получения текущей корзины пользователя

}

}