Задание (К.К. Рыжков)

**Требование к оформлению результата:**

все результаты должны быть оформлены **в едином файле** без указания внутри него ссылок на другие ресурсы. После выполнения задания скиньте контактному лицу ссылку на этот файл.

Исключение - если решите описывать REST API в сваггере, то это можно оформить и скинуть отдельным файлом.

1. Базы данных - тест

Для приведенных ниже вопросов выберите верные утверждения. Верным может быть одно или несколько утверждений.

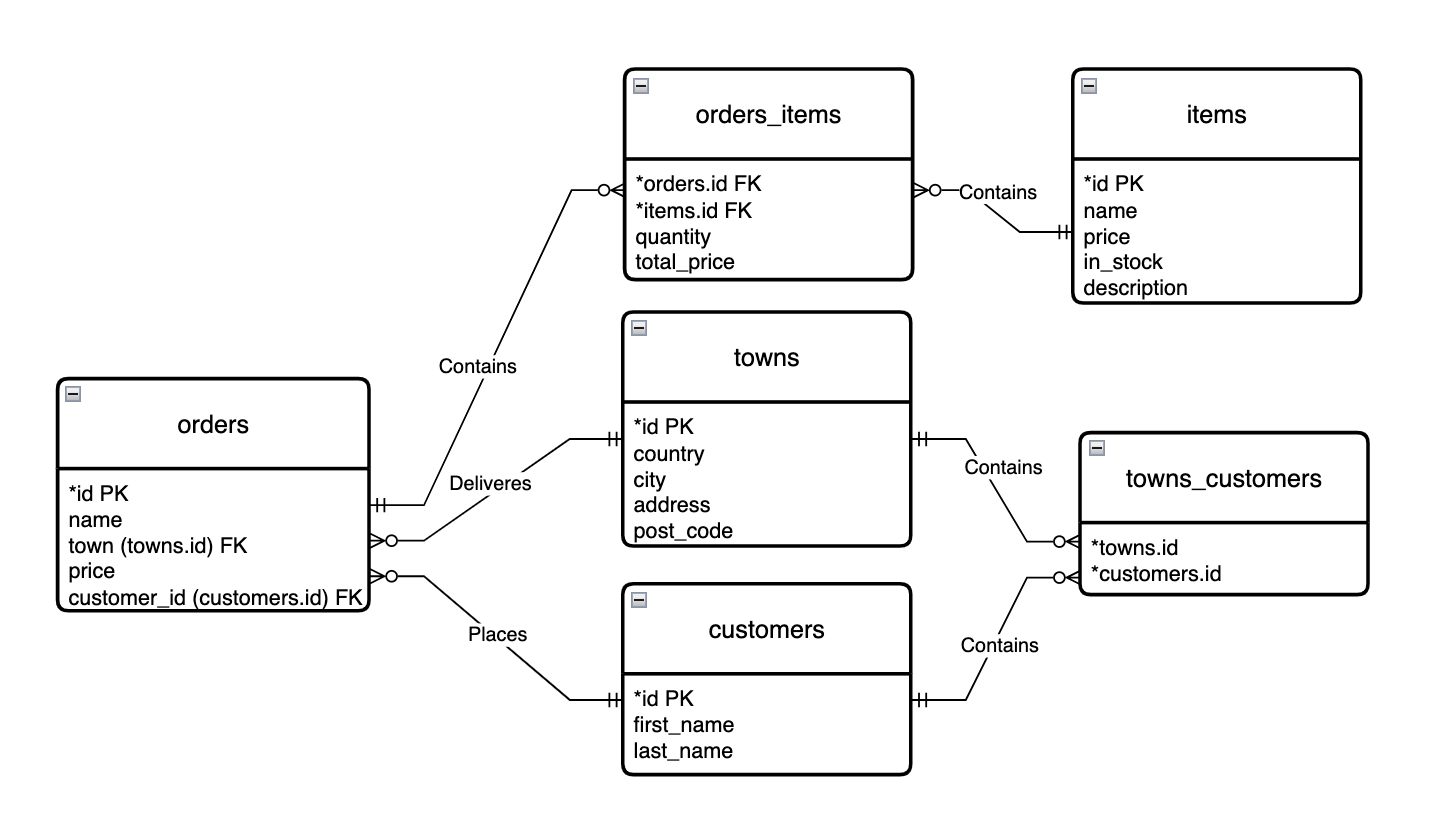
| 1 | Содержит ли какую-то информацию таблица, в которой нет полей? | 1. Содержит информацию о структуре БД  2. Не содержит никакой информации  3. Таблица без полей существовать не может  4. Содержит информацию о будущих записях |
| --- | --- | --- |
| 2 | В записи файла реляционной БД может содержаться: | 1. Исключительно однородная информация (данные только одного типа)  2. Только текстовая информация  3. Только логические величины  4. Неоднородная информация (данные разных типов)  5. Исключительно числовая информация |
| 3 | Чем первичный ключ отличается от внешнего ключа? | 1. Первичный ключ всегда состоит из множества столбцов, а внешний ключ состоит из одного столбца  2. Значения первичного ключа всегда должны быть уникальными и не могут быть null, значения внешнего ключа могут повторяться  3. Внешний ключ является идентификатором строки, а первичный ключ используется для связи между таблиц  4. Первичный ключ является идентификатором для строки, а внешний ключ используется для связывания таблиц |
| 4 | В какой нормальной форме говорится о том, что все атрибуты зависят от первичного ключа, а не от его части? | 1. 1НФ  2. 2НФ  3. 3НФ  4. 4НФ |
| 5 | В каком порядке в СУБД выполняются операторы SELECT, FROM, GROUP BY? | 1. Сначала SELECT, потом FROM и только потом GROUP BY  2. Сначала GROUP BY, потом SELECT и только потом FROM  3. Сначала FROM, потом SELECT и только потом GROUP BY  4. Сначала FROM, потом GROUP BY и только потом SELECT |
| 6 | Чем отличается оператор WHERE от HAVING | 1. Оператор WHERE применяется для фильтрации групп, а HAVING - для фильтрации отдельных строк  2. Оператор HAVING применяется для фильтрации групп, а WHERE - для фильтрации отдельных строк  3.HAVING работает только с агрегатными функциями, а WHERE может работать с любыми типами выражений  4. WHERE может использоваться для фильтрации по любому полю или выражению, а HAVING - только для фильтрации по выражению в списке выбора или агрегатной функции  5.HAVING всегда используется после GROUP BY, а WHERE может использоваться до или после GROUP BY |
| 7 | Какой результат покажет выполнение операторов SELECT COUNT (\*)? | 1. Число строк таблицы, указанной во FROM, не включая значение NULL  2. Число строк таблицы, указанной во FROM, где ячейка содержит символ  3. Число строк таблицы, указанной во FROM, включая значение NULL   * для PostgreSQL   4. Сумма строк таблицы, указанной во FROM, где ячейка содержит символ |
| 8 | В таблице «Animals» базы данных зоопарка содержится информация обо всех обитающих там животных, в том числе о лисах: red fox, grey fox, little fox. Напишите запрос, возвращающий информацию о возрасте лис | 1. SELECT age FROM Animals WHERE Animal LIKE “%fox”  2. SELECT age FROM %Fox.Animals  3. SELECT age  FROM Animals  WHERE Animal = fox  4. SELECT %fox age FROM Animals |
| 9 | Чем отличается DELETE от TRUNCATE? | 1. DELETE и TRUNCATE - это одно и то же  2. DELETE используется для удаления одной или нескольких строк из таблицы, а TRUNCATE используется для удаления всех строк из таблицы  3. DELETE может использовать условие WHERE, а TRUNCATE всегда удаляет все записи из таблицы  4. DELETE удаляет данные из таблицы, а TRUNCATE удаляет саму таблицу |
| 10 | Дана таблица:   | **COLOR** | | --- | | BLUE | | RED | | null | | RED |   Каким будет результат запроса?  SELECT COUNT (DISTINCT color) FROM Table | 1. BLUE,RED,NULL  2. 3 - в Postgre SQL  3. 1,2,4  4. 2 - для остальных |

2. Базы данных - ER

В базе данных есть таблица заказов - orders. В ней есть поля: id (идентификатор заказа), name (название заказа), town (адрес доставки заказа), price (цена заказа), customer\_id (идентификатор покупателя). Также есть таблицы: towns (справочник адресов), items (товары), customers (покупатели). Известно, что между orders и items предполагается связь многие-ко-многим.

**Что нужно сделать:**

Спроектируйте ER-диаграмму с учетом этих вводных. Состав полей таблиц укажите на свое усмотрение с учетом условия, но для таблицы orders учтите те поля, что указаны.



2. Интеграции

Представим, что Вы работаете аналитиком и проектируете работу приложения интернет-магазина. Вам нужно заложить и спроектировать следующий сценарий: отображение витрины товаров (список товаров с кратким описанием), переход с витрины на экран с детальным описанием конкретного товара, добавление товара в корзину.

**Что нужно сделать:**

1. Спроектируйте REST API, которые нужны для реализации описанного сценария. Решение должно включать описание запросов и описание или пример ответа для каждого из запросов в формате JSON. Способ описания - на ваше усмотрение. По составу полей товара можете ориентироваться на любой известный интернет-магазин.
2. **Метод получения списка продуктов:**

* **Протокол:** HTTPS
* **Базовый URL:** https://api.shop.com/
* **URL:** /products
* **Метод:** GET
* **Параметры:**

| № | Название, обязательность | Описание | Тип | Комментарий |
| --- | --- | --- | --- | --- |
| 1 | limit\* | Количество элементов на странице | Integer |  |
| 2 | offset | Количество пропускаемых элементов при генерации страницы | Integer |  |
| 3 | searchText | Текст из поля для поиска по списку | String | Min кол-во символов - 1 Max кол-во символов - 25 |

* **Пример запроса:**

| https |
| --- |
| GET /products HTTPS/1.1  Host: https://api.shop.com/ |

* **Пример ответа с кодом 200:**

| Json |
| --- |
| {  "products": [  {  "id": "1",  "name": "Pomidor",  "icon": "<https://example.com/pomidor.jpg>",  "price": 99,99,  "short\_description": "Pomidor from Krasnodar"  },  {  "id": "2",  "name": "Ogurets",  "icon": "[https://example.com/](https://example.com/pomidor.jpg)ogurets.jpg",  "price": 69.99,  "short\_description": "Ogurets from Astrakhan"  },  {  "id": "3",  "name": "Arbuz",  "icon": "[https://example.com/](https://example.com/pomidor.jpg)arbuz.jpg",  "price": 129,99  "short\_description": "Arbuz from Anapa"  },  {  "id": "4",  "name": "Vinograd",  "icon": "[https://example.com/](https://example.com/pomidor.jpg)vinograd.jpg",  "price": 149,99  "short\_description": "Vinograd from Krym",  }  ]  } |

| № | Название | Описание | Тип | БД |
| --- | --- | --- | --- | --- |
| 1 | id | Идентификатор продукта | Integer | products.id |
| 2 | name | Название продукта | String | products.name |
| 3 | icon | Иконка продукта | String | products.icon |
| 4 | price | Цена продукта | Integer | products.price |
| 5 | short\_description | Краткое описание продукта | Float | products.short\_description |

* **Ошибки:**
  + 500 Server Error - Ошибка сервера.
  + 400 Bad Request - Некорректные параметры запроса.
  + 404 Not Found - Страница не найдена.

1. **Метод получения информации о продукте:**

* **Протокол:** HTTPS
* **Базовый URL:** https://api.shop.com/
* **URL:** /products/{id}
* **Метод:** GET
* **Параметры:**

| № | Название, обязательность | Описание | Тип | Комментарий |
| --- | --- | --- | --- | --- |
| 1 | id\* | Уникальный идентификатор продукта | Integer | Целое положительное число |

* **Пример запроса:**

| https |
| --- |
| GET /products/{id} HTTPS/1.1  Host: https://api.shop.com/ |

* **Пример ответа с кодом 200:**

| Json |
| --- |
| {  "id": "1",  "name": "Pomidor",  "icon": "[https://example.com/](https://example.com/pomidor.jpg)pomidor.jpg",  "price": 69,99,  "description": "Sochniy pomidor rodom iz Krasnodarskogo Kraya",  "in\_stock": true,  "rating": 5  } |

| № | Название | Описание | Тип | БД |
| --- | --- | --- | --- | --- |
| 1 | id | Идентификатор продукта | Integer | products.id |
| 2 | name | Название продукта | String | products.name |
| 3 | icon | Иконка продукта | String | products.icon |
| 4 | price | Цена продукта | float | products.price |
| 5 | description | Описание продукта | String | products.description |
| 6 | in\_stock | Наличие продукта | Boolean | products.in\_stock |
| 7 | rating | Рейтинг продукта | Float | AVG (rating.value) где products.id = id нужного продукта |

* **Ошибки:**
  + 500 Server Error - Ошибка сервера.
  + 400 Bad Request - Некорректные параметры запроса.
  + 404 Not Found - Страница не найдена.

1. **Метод добавление продукта в корзину:**

* **Протокол:** HTTPS
* **Базовый URL:** https://api.shop.com/
* **URL:** /products/cart
* **Метод: POST**
* **Параметры:**

| № | Название, обязательность | Описание | Тип | Комментарий |
| --- | --- | --- | --- | --- |
| 1 | id\* | Уникальный идентификатор продукта | Integer | Целое положительное число |
| 2 | \*quantity | Количество продукта для покупки | Integer | Целое положительное число |

* **Пример запроса:**

| https |
| --- |
| {  "id": 1,  "quantity": 1  } |

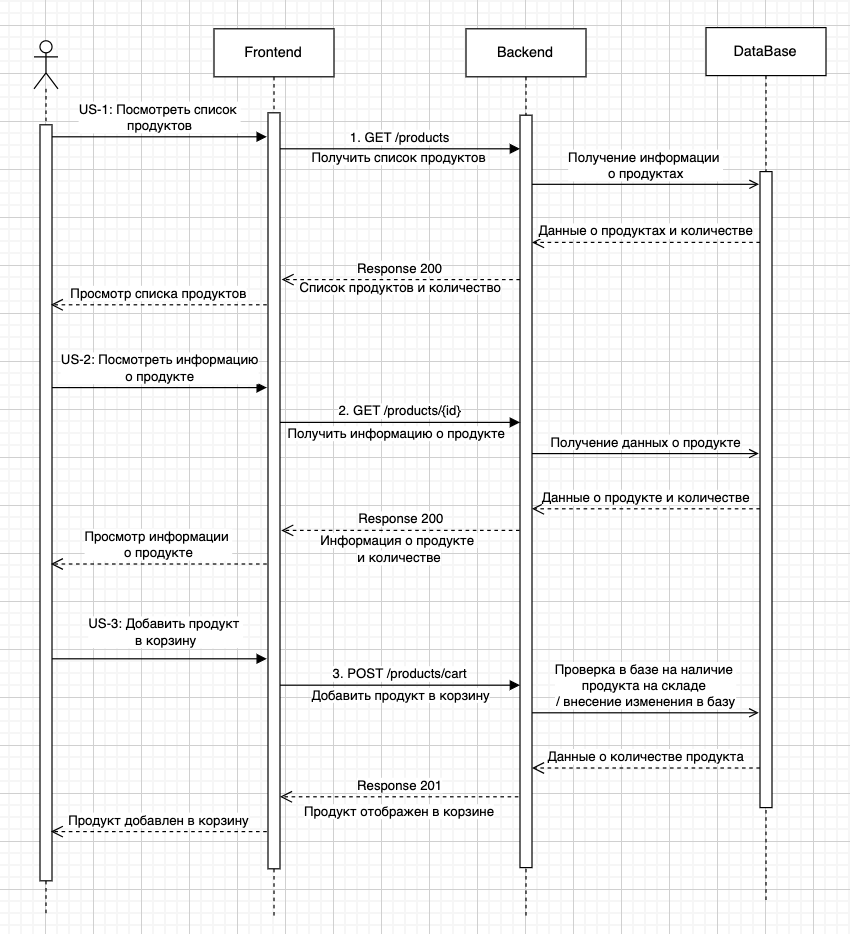
* **Пример ответа с кодом 201:**

| Json |
| --- |
| {  "id": "1",  "products\_id": "1",  "name": "Pomidor",  "quantity": 1  "icon": "[https://example.com/](https://example.com/pomidor.jpg)pomidor.jpg",  "total\_price": 69,99  } |

| № | Название | Описание | Тип | БД |
| --- | --- | --- | --- | --- |
| 1 | id | Идентификатор корзины | Integer | cart.id |
| 1 | products\_id | Идентификатор продукта | Integer | products.id |
| 2 | name | Название продукта | String | products.name |
| 3 | quantity | Количество продукта | Integer | products\_cart.quantity |
| 3 | icon | Иконка продукта | String | products.icon |
| 4 | total\_price | Общая цена за продукт в отношении к количеству | float | products\_cart.total\_price |

* **Ошибки:**
  + 500 Server Error - Ошибка сервера.
  + 400 Bad Request - Некорректные параметры запроса.
  + 404 Not Found - Страница не найдена.

1. Постройте Sequence UML диаграмму для этого сценария.



4. Алгоритмическое мышление

Возьмем в качестве примера банковское мобильное приложение.

Исходные условия: у вас в руке смартфон, на котором установлено банковское приложение (телефон выключен).

**Что нужно сделать:**

Используя любую нотацию, опишите в виде диаграммы процесс, в рамках которого Вы, используя банковское приложение, пополните баланс своего телефона на 100Р. В качестве примера можете использовать любое мобильное банковское приложение, что у Вас есть.

В данном случае лучше всего подходит BPMN нотация, разделил промежуточными событиями для удобства чтения:

