Задание

**Требование к оформлению результата:**

все результаты должны быть оформлены **в едином файле** без указания внутри него ссылок на другие ресурсы

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  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  3. 1,2,4  4. 2 |

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.

Ответ:

Составлена ER – диаграмма в 2-х вариантах.

1 вариант: таблица towns – отдельный, независимый справочник (предполагается, что адреса уже известны – например, от интеграции с сервисом адресов)

2 вариант: предполагалось, что пользователь может создать несколько адресов в таблице towns, после чего один из них сможет использовать при оформлении заказа

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Вариант 1

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Вариант 2

3. 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. Постройте Sequence UML диаграмму для этого сценария.

Ответ:

Метод и Endpoint: **GET** **/catalog**

Описание: Возвращает витрину интернет-магазина с кратким описанием товаров или ошибку.

Пример запроса: [http://localhost:8090/**catalog**](http://localhost:8090/catalog)

Пример учпешного ответа:

## [

## {

## "id": 75433,

## "name": "Корабль Джанго Фетта",

## "price": 2499,

## "series": "Star Wars"

## },

## {

## "id": 84521,

## "name": "Хогвартс Экспресс",

## "price": 5499,

## "series": "Harry Potter"

## }

## ]

Описание ответа:

id:

type: integer

example: 75433

description: Номер набора lego

name:

type: string

example: Корабль Джанго Фетта

description: Название набора

price:

type: number

example: 2499

series:

type: string

example: Star Wars

description: Название серии

Метод и Endpoint: **GET** **/catalog/{id}**

Описание: Возвращает карточку товара или ошибку.

Входные параметры:

Id - идентификатор набора lego конструктора

Пример запроса: [http://localhost:8090/**catalog**/75433](http://localhost:8090/catalog/75433)

Пример успешного ответа:

{

"id": 75433,

"name": "Корабль Джанго Фетта",

"price": 2499,

"series": "Star Wars",

"year": 2025,

"info": "3 минифигурки, 707 деталей, размеры в собранном виде 9 x 21 x 21 см"

}

Описание ответа:

id:

type: integer

example: 75433

description: Номер набора lego

name:

type: string

example: Корабль Джанго Фетта

description: Название набора

price:

type: number

example: 2499

description: Цена за 1 единицу набора

series:

type: string

example: Star Wars

description: Название серии

year:

type: integer

example: 2025

description: Год выпуска набора

info:

type: string

example: 3 минифигурки, 707 деталей, размеры в собранном виде 9 x 21 x 21 см

description: Описание набора

Метод и Endpoint: **POST /cart**

Описание: Добавляет товар в корзину или отображает ошибку

Входные параметры:

Id - идентификатор набора lego конструктора

Quantity – количество товара, добавляемое в корзину

Пример запроса: <http://localhost:8090/cart>

{

"id": 75433,

"quantity": 2

}

Пример успешного ответа:

{

"id": 75433,

"name": "Корабль Джанго Фетта",

"price": 2499,

"quantity": 2,

"total\_price": 4998

}

Описание ответа:

id:

type: integer

example: 75433

description: Номер набора lego

name:

type: string

example: Корабль Джанго Фетта

price:

type: number

example: 2499

description: Цена за 1 единицу набора

quantity:

type: integer

example: 2

description: Количество товара добавлено в корзину

total\_price:

type: number

example: 4998

description: общая сумма (price × quantity)

Коды ответов:

 **200 OK** — успешный ответ (возвращает данные).

 **400 Bad Request** — неверные данные (например, отрицательное количество).

 **404 Not Found** — товар не найден.

 **500 Internal Server Error** — внутренняя ошибка.

Пример ошибки:

{

"code": 404,

"name": "Товар не найден"

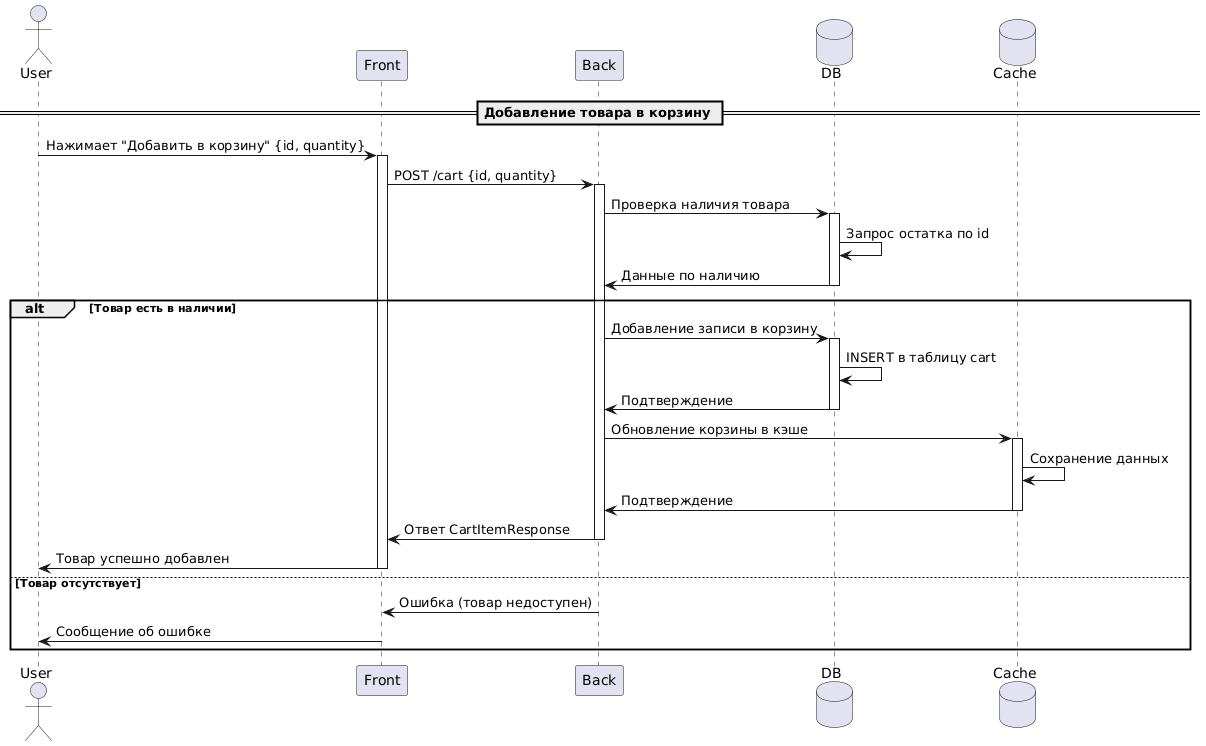
}

Изображение выглядит как текст, диаграмма, Параллельный, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, снимок экрана, Параллельный

Автоматически созданное описание



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.

Ответ: Составлена схема пополнения баланса своего мобильного телефона на 100 р., начиная с этапа включения мобильного телефона в нотации BPMN. Учтено несколько сценариев, когда баланс пополнить невозможно, к примеру проблемы с телефоном или недостаточно средств на счете в банке.

