**Database Documentation**

The database is designed to support the e-commerce platform by managing products and orders. The schema includes three key tables: Products, Orders, and OrderDetails, which are related to each other to model customer transactions.

**1. Products Table:**

Description: This table contains information about the products available in the store. Each product has a unique identifier (Id), a name, a price, and the amount of stock available.

Columns:

Id: Auto-incremented integer, serves as the primary key.

Name: Name of the product, with a maximum length of 100 characters.

Price: Price of the product, stored as DECIMAL to handle monetary values ​​with two decimal places.

Stock: Number of units available for sale.

**2. Orders Table:**

Description: Stores information related to customer orders. Each order has a unique identifier, a field to store the customer ID, the date the order was placed, and the total amount.

Columns:

Id: Auto-incremented integer, serves as the primary key.

CustomerId: Identifier of the customer who placed the order.

OrderDate: Date and time the order was placed.

TotalAmount: Total amount of the order, calculated as the sum of the products ordered.

**3. OrderDetails Table:**

Description: This table links products to orders through a composite primary key consisting of OrderId and ProductId. It stores information about the quantity of each product ordered and the unit price at the time of the order.

Columns:

OrderId: Relationship to the Orders table, indicating which order the product belongs to.

ProductId: Relationship to the Products table, identifying the product in the order.

Quantity: Number of units of the product in the order.

UnitPrice: Unit price of the product at the time of purchase.

**Relationships between Tables**

**Orders and OrderDetails**: One-to-many relationship. An order can have multiple products associated with it via the OrderDetails table, but each product in OrderDetails belongs to a single order.

**Products and OrderDetails:** One-to-many relationship. A product can be present in multiple order details, but each order detail contains only one product.

**Indexes**

Indexes have been created on the OrderId and ProductId columns of the OrderDetails table to optimize queries that search for order details by order or product.

Scalability Strategy

This design is prepared to scale horizontally by using indexes on high-volume tables, such as OrderDetails. Additionally, separating order details into a specific table allows for efficient management of products and orders, ensuring that the system can scale as the number of transactions grows.

**Data Access and Management**

Data Access: SQL statements are used to perform CRUD (Create, Read, Update, Delete) operations on the tables. Queries can be simple or complex, depending on the need to retrieve specific information, such as all products, details of a specific order, etc.

Data Management: Data is managed using stored procedures, functions, and views to encapsulate business logic. This helps keep the code clean and makes database maintenance easier. Additionally, indexes are implemented on key columns to optimize query performance.

**Considerations for Non-Relational Databases**

If at some point it were decided to use a non-relational database (e.g. MongoDB), the data structure might look different, as it would be document-based. In this case, products, orders, and order details could be stored as JSON documents, allowing for greater flexibility in data structure. However, this would also entail a re-evaluation of how data is accessed and managed, since the typical foreign key relationships of a relational database would not exist.