GitHub Link for Article

Developing and Designing of Amazon's E-commerce

Name: S.U. Shiva Prasad

Subject: Data Concepts and Data Design

Mentor: Dr. Junaid Qazi

LinkedIn

GitHub Link for Article

Table of Contents

GitHub Link for Article

1 OVERVIEW

This case study explores the design of a scalable, secure database for Amazon's e-commerce platform. It efficiently manages millions of users, products, and transactions, enabling real-time processing, personalization, and data security. Key components include tables for Users, Products, Orders, and Vendors, with the system implemented using AWS RDS and Azure SQL for seamless operation.

2 INTRODUCTION

With millions of users and transactions daily, Amazon's e-commerce platform relies on a robust, scalable, and secure database to function seamlessly. This article explores the design and implementation of a database that supports real-time processing, personalisation, and efficient data management to deliver a smooth shopping experience.

3 MISSION

The goal of Amazon's database system is to offer a seamless and personalized shopping experience. The system is designed to handle the growing demands of millions of customers, products, and transactions while ensuring data security and smooth operations.

Key Objectives

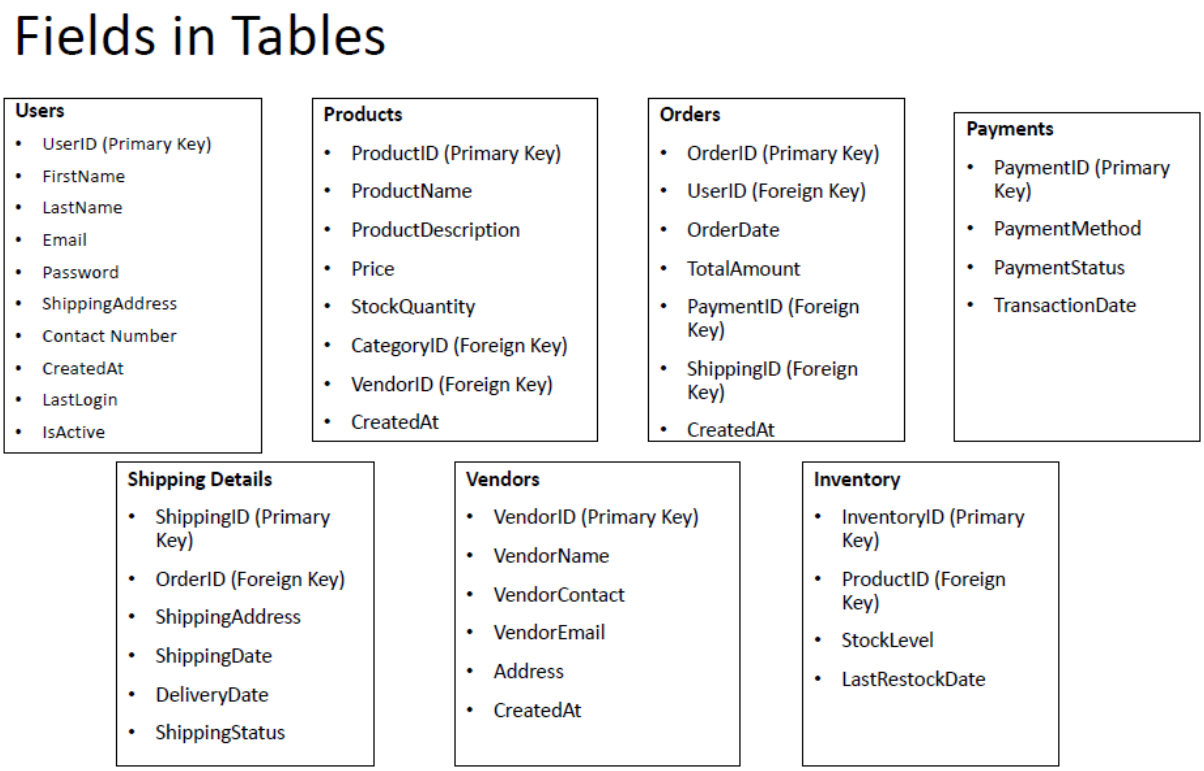
- Ease of Use: Simplify the shopping experience through intuitive navigation and personalised recommendations.
- Efficiency: Ensure real-time processing of millions of transactions.
- Security: Protect sensitive customer data, such as payment and personal information.
- Scalability: Accommodate global growth in users, products, and orders.
- Personalization: Tailor shopping experiences based on customer behaviour and preferences.

4 DATABASE DEVELOPMENT

The database structure supports key operational components of the platform, including users, products, orders, payments, and shipping details.

GitHub Link for Article

4.1 Core Component



4.2 Table Relationships

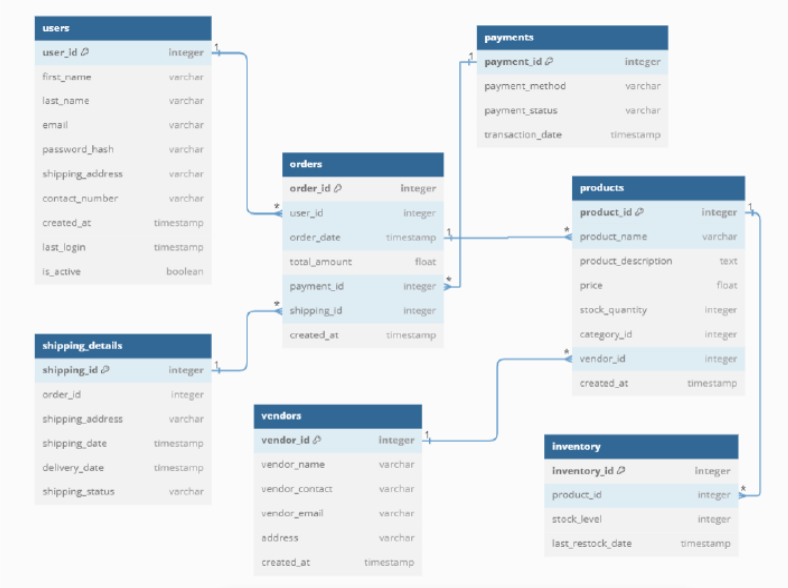
- **Users** ↔ **Orders**: One-to-many relationship (one user can place multiple orders).
- **Orders** ↔ **Order Items**: One-to-many relationship (one order contains multiple items).
- **Products** ↔ **Vendors**: Many-to-one relationship (each product has one vendor).
- **Orders** ↔ **Shipping Details**: One-to-one relationship (each order has associated shipping details).
- **Products** ↔ **Reviews**: One-to-many relationship (multiple reviews for a product).

4.3 Entity Relationship Diagram(ERD)

As part of the project, I worked with cloud platforms such as **Azure SQL** to set up databases for handling e-commerce data. I also explored relational database systems like **PostgreSQL** and **MySQL** to manage large datasets efficiently.

GitHub Link for Article

Entity-Relationship Diagram (ERD)



5 CONCLUSION

This case study highlights how a well-designed, scalable database can support the operations of a global e-commerce platform like Amazon. The database handles the complexities of user interactions, product listings, and order management and ensures security, personalisation, and future scalability. The design is ready to meet Amazon's growing needs in a rapidly expanding digital marketplace.

LinkedIn

6 APPENDIX: SQL QUERY COMPONENTS

Component	Description
Query Purpose	Retrieve detailed information about users' orders, including payment and shipping details.
Selected Fields	-Users.UserID -Users.FirstName -Users.LastName -Orders.OrderID -Orders.OrderDate -Orders.TotalAmount -Payments.PaymentMethod -Payments.PaymentStatus -ShippingDetails.ShippingAddress -ShippingDetails.ShippingDate -ShippingDetails.DeliveryDate - ShippingDetails.ShippingStatus
Main Tables	-Users -Orders -Payments - shipping details
Relationships	-Users to Orders: One-to-Many (One user can have multiple orders) -Orders to Payments: One-to-One (Each order has one payment) -Orders to ShippingDetails: One-to-One (Each order has one shipping detail)
Join Conditions	-Orders.UserID=Users.UserID -Orders.PaymentID=Payments.payment - Orders.OrderID = ShippingDetails.OrderID
SQL Clauses	-SELECT: Specifies the columns to be retrieved. -FROM: Indicates the primary table (Orders) to retrieve data from. - JOIN: Combines rows from two or more tables based on related columns.
Output Expected	A comprehensive list of orders, including user information, payment details, and shipping statuses.

GitHub Link for Article

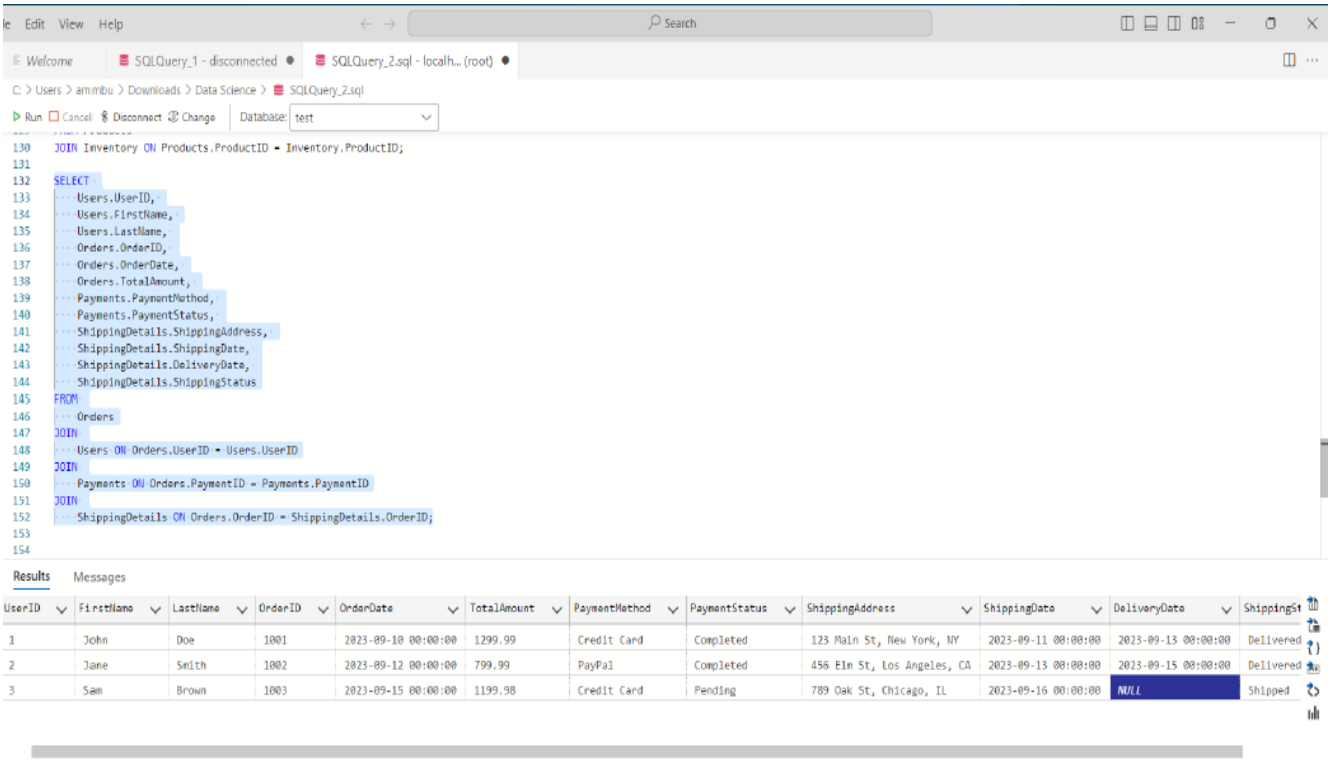
Component	Description
Potential Use Cases	-Analyse user purchasing behaviour Monitor payment and shipping statuses - Generate reports for customer service inquiries

7 APPENDIX TABLE

Section	Content
Tables Used	Users, Products, Orders, Vendors, Order Items, Shipping Details, Reviews
Key Relationships	Users ↔ Orders (One-to-Many), Orders ↔ Order Items (One-to-Many), Products ↔ Vendors (Many-to-One), Orders ↔ Shipping Details (One-to-One), Products ↔ Reviews (One-to-Many)
Database Platforms	AWS RDS, Azure SQL, PostgreSQL, MySQL
Key Objectives	Ease of Use, Efficiency, Security, Scalability, Personalization
Core Components	Users, Products, Orders, Payments, Shipping Details
ERD Tools Used	Azure SQL, PostgreSQL, MySQL
Security Measures	Data encryption, Secure payment processing, Privacy Protocols
Personalization Methods	Recommendations based on browsing history, purchase history, and user preferences
Scalability	Designed for millions of users and products, with the ability to grow alongside the platform's needs
Technologies Involved	Cloud Platforms (AWS, Azure), Relational Databases (PostgreSQL, MySQL), Real-time Data Processing
Conclusion Summary	A scalable, secure, and efficient database designed to handle Amazon's e-commerce platform growth and needs.

LinkedIn

GitHub Link for Article



LinkedIn