Overview

The Amazon Order Reviews System is designed to efficiently manage and analyze customer reviews and orders from Amazon. The system integrates various data concerning customers, items, orders, and reviews, thereby facilitating a comprehensive understanding of customer feedback and purchasing patterns. This report outlines the system's architecture, including its data model, data population approach, implemented triggers for maintaining data integrity, and the challenges encountered during dataset integration.

Data Model

The system's data model is constructed around several core entities: Person, Customer, Reviewers, Item, Order, review_item, Log_order, and Log_Review. These entities are interconnected through various relationships to represent the system's domain accurately. For instance, the Customer and Reviewers entities are related to Person, indicating a design that accommodates both customer details and their reviewing activities. Items and orders are managed through the Item and Order tables, respectively, with the review_item table connecting reviews to specific items. Log_order and Log_Review tables serve as log entities to track order and review metrics.

Data Population Approach and Challenges

The initial step in populating the database involved defining the schema through Data Definition Language (DDL) scripts. Following this, data was imported from CSV files containing Amazon order and review data, which involved a meticulous process of mapping CSV columns to the database schema. This process unveiled significant challenges, primarily due to the schema being defined prior to the acquisition of appropriate datasets. The complexity lay in aligning the real-world, often messy data with the predefined schema, necessitating adjustments and transformations to ensure compatibility.

The difficulty was particularly pronounced in cases where the dataset lacked direct correspondence to the schema's entities and attributes. For example, integrating product and customer information required a careful extraction of relevant details from the datasets, followed by a process to avoid duplicate entries and maintain data integrity. The challenges were compounded by the necessity to link reviews with both customers and items, requiring an intricate understanding of the data's structure and content.

Data model Enhancement

Views have been implemented to simplify complex queries and present the data in a more accessible format. For instance, the Customer_Orders_View provides a snapshot of order details alongside customer names, facilitating an easier analysis of sales data. The Item_Reviews_View consolidates item details with their respective reviews, offering immediate insight into customer feedback. Furthermore, the Low_Quantity_Items view serves as an alert system for inventory management, highlighting items that may require restocking.

In addition to these views, domain constraints were introduced to enforce data integrity at the database level. Constraints such as chk_item_quantity_non_negative and chk_order_price_positive ensure that item quantities do not fall below zero and that prices are always above zero, respectively. These constraints act as business rules, embedding the system's logic directly within the database structure.

MySQL's Approach to Domains and Types

MySQL's treatment of domains and types diverges from traditional implementations found in other RDBMS such as PostgreSQL, which supports the DOMAIN keyword explicitly. In MySQL, domain-like constraints are applied directly to table columns through data types and CHECK constraints. These constraints are responsible for validating data before it is committed to the database, providing a guarantee that data adheres to predefined rules. For example, the CHECK constraint chk_review_title_not_empty ensures that no review is submitted with an empty title, thereby maintaining the quality and usefulness of customer feedback.

Implemented Triggers

To maintain data integrity and automate the system's workflow, several triggers were implemented. These include:

update_log_review: Triggered after inserting a new review, it updates the review count for the related item in the Log_Review table.

update_log_order: Activated after an order is placed, it increments the total orders placed by a customer in the Log_Order table.

remove_log_review and remove_log_order: These triggers handle the deletion of related logs in the Log_Review and Log_Order tables when items or customer records are removed, ensuring the database remains consistent and accurate.

Conclusion

The Amazon Order Reviews System exemplifies a robust data management solution that addresses the complex requirements of handling and analyzing ecommerce data. Despite the challenges encountered, particularly in dataset integration, the system successfully facilitates a detailed analysis of customer orders and reviews. Future enhancements could include refining the data model to accommodate more nuanced relationships between entities, implementing more sophisticated data cleaning processes to handle diverse datasets, and exploring advanced data analytics capabilities to extract deeper insights from the data. Through continuous improvement, the system can evolve to meet the dynamic needs of ecommerce analytics more effectively.