EAS 550 DATA MODEL QUERY LANGUAGE

PROJECT PHASE 1

Team Members: Hema Priya Balaji, Manimegalai Mamallan, Sandeep Pudi

Project: E-commerce & Retail Analytics

Dataset Link: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

Olist Brazilian E-Commerce OLTP Database

Introduction: The aim of this phase 1 was to design a clean, normalized and reliable OLTP data model for the Olist Brazilian E-commerce dataset. The dataset is initially stored across various CSV files, each representing different aspects of customer, order, products, sellers, payments and reviews.

To ensure data integrity and avoid redundancy, the schema was modelled using Entity Relation Diagram and normalized to third normal form(3NF). This report explains the design choices and depicts how the schema avoids common data anomalies such as insertion, updating and deletion anomalies.

Entity Identification and Design Rationale:

The goal of the schema designed is to model the Olist e-commerce data in a way that is logically clear, non-redundant and resistant to data anomalies. To achieve this, the data is decomposed into several related tables, each representing a single business concept and normalized.

- **Core entities**

  During analysis of the dataset, the following below mentioned are the entities that were identified:

ENTITY	RATIONALE
Customer	The customer table has these attributes: customer_id PK, customer_unique_id, customer_zip_code_prefix, customer_city and customer_state. Each row represents a single customer record keyed by customer_id, Demographic attributes such as city, state, ZIP prefix and it depend only on customer_id.
Sellers	The seller table has these attributes:seller_id PK, seller_zip_code_prefix, seller_city, and seller_state. Seller address info is stored once per seller,

	avoiding repetition across multiple orders or products.
Geolocation	The geolocation table has the attributes: geolocation_zip_code_prefix PK, geolocation_lat, geolocation_lng, geolocation_city, and geolocation_state. Geospatial attributes are separated so the same ZIP prefix and coordinates can be reused by both customers and sellers without duplication.
Product Category	The product category table has the attributes: category_id PK, product_category_name, and product_category_name_english. Product categories are modeled as a separate lookup table. This avoids repeating category names and translations for every product and allows consistent category management.
Product	The product table has the attributes: product_id PK, category_id FK, product_name_length, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, and product_width_cm. Each product is keyed by product_id and references its category via category_id. All other attributes describe the product itself and depend only on product_id.
Order	The order table has the attributes: order_id PK, customer_id FK, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, and order_estimated_delivery_date. Orders are separated from order items to support the natural 1:Nrelationship (one order, multiple items). All attributes depend on order_id.
OrderItem	The order item table has the attributes: order_id and order_item_id PK, product_id FK, seller_id FK, shipping_limit_date, price, and freight_value. This table captures each line item in an order. The composite key are order_id and order_item_id uniquely identifies a line, and

	all non-key attributes depend on the whole key (the specific item within an order).
Payment	The payment table has the attributes: order_id and payment_sequential PK, payment_type, payment_installments, and payment_value. Payments are modeled separately to support multiple payments per order (e.g., split payments). The composite key is order_id and payment_sequential ensures uniqueness and correctly models the 1:N relationship between orders and payments.
Review	The review table has the attributes: review_id PK, order_id FK, review_score, review_comment_title, review_comment_message, review_creation_date, and review_answer_timestamp. Reviews are separated from orders to allow a flexible link between an order and its review details, without mixing review text with transactional data.

Normalization to third normal form:

1. **First Normal form**

A table is in first normal form as it has each attribute that holds atomic values and no repeating groups.

- Every column stores atomic values such as price, review_score, city.
- No multivalued attributes as it has repeated information such as multiple payment attempts is modeled properly in the payment table using a composite key that is order_id, and payment_sequential.

2. **Second Normal Form**

A table is in second normal form if it is in first normal form and non-key attributes depend entirely on primary key.

Design Illustration:

Order Items: Order Items table has the primary key order_item_id and attributes like price, freight_value and shipping_limit_date depend on the entire composite key not just order_id.

Payments: Payments have payment_sequential as primary key. The attributes are payment_type and payment_value depend fully on entire composite key. No attribute in any composite-key table depends on only one part of the key which indicates that 2NF is satisfied.

3. **Third Normal Form**

A table is in 3NF if it is in 2NF. It has no transitive dependencies that non-key attributes must not depend on other non-key attributes.

How does this design satisfy 3NF and challenges solved?

Geolocation Normalization: In raw Olist data, the customers and sellers repeatedly store the below mentioned attributes zip_code_prefix, city, state, latitude and longitude. This creates more repetition and risk of inconsistencies

Solution: All geographic data on the geolocation table. Customers and sellers now store only the zip_code_prefix foreign key. The flow shown below removes the transitive dependency: Customer_id → zip_code_prefix → city/state/lat/long

Product Category Normalization: Product data included the category name. If category names change or translations are updated, repeating the name in every product row creates update anomalies.

Solution: A dedicated productcategory table with category_id as primary key and category names. The flow shown below removes the transitive dependency: Product_id → product_Category_id →category_name

Order, payment and review tables are isolated.

No table mixes information from various business concepts. For example:

- Orders store timestamps and status and there are no products, sellers, or review details.
- Reviews are stored separately and this reference only the order.

This separation avoids updating anomalies and simplifies ingestion.

Avoidance of Data Anomalies:

This design of the database helps to avoid the classic update, insert and delete anomalies:

Update Anomalies

- Category names are stored once in product_categories. Updating a category's English name requires changing only one row, instead of updating thousands of product rows.
- Customer and seller's location details are stored in their own tables, so changing a city or state is done once per entity.

Insert Anomalies

- Customer and seller's location details are stored in their own tables, so changing a city or state is done once per entity.
- New orders can be inserted into orders without requiring immediate payment or review rows, related data can be added later to order_payments or order_reviews.

Delete anomalies

- Deleting an order item(order_items) does not remove the master order record, the product, or the seller.
- Deleting a review from order_reviews does not remove the underlying order row, so historical transaction data is preserved.
- Category and geolocation data are kept in separate tables, so deleting an order or product does not accidentally delete shared reference data.

Summary

The final OLTP schema is normalized to third normal form completely by eliminating redundancy and ensuring reliable and anomaly free operations such as updates, inserts and deletes.

The entity relationship diagram and Structured Query Language schema adhere to best practices for relational modeling by ensuring the listed below:

- High data integrity
- Efficient storage
- Clean ingestion pipelines
- Reliable querying for future OLAP/Analytics phases

This design structures a strong foundation for phase-2. This ensures that schema is robust, easier to maintain and well suited for analytics on the Olist e-commerce dataset.