Design and Structure of Amazon's E-commerce Database

1st Juan David Quiroga
Universidad Distrital Francisco Jose de Caldas
Bogotá, Colombia

2nd Luis Alejandro Morales
Universidad Distrital Francisco Jose de Caldas
Bogotá, Colombia

***Abstract*—This document is a small way to see how one of the most recognized Amazon web pages is built using the entity-relationship model known as DER. Thus showing parts as it would be as they could work as purchases sales suppliers among other things that will see below**

***Index Terms*—Amazon, database, DER,**

I. Introduction

Amazon is one of the world's largest e-commerce platforms, founded in 1994 by Jeff Bezos. Over the years, Amazon has built a reputation based on convenience, speed of delivery, and a wide variety of products, which has positioned it as a global leader in online commerce. Amazon's success is based in large part on its ability to handle millions of daily transactions efficiently, which requires a robust and scalable database system. This system allows for the storage, organization, and access to large volumes of product, customer, and transaction information, ensuring that data is processed quickly and accurately. This paper aims to describe Amazon's database design in the context of its e-commerce operation, focusing on the main entities and relationships that allow managing the vast amounts of data handled by the platform.

II. Method and Materials

A. Taking a look at the DER

The method used to describe and model the Amazon database is the entity-relationship diagram (DER), a widely recognized technique in the field of software and database engineering. The DER is a tool that allows to graphically represent the logical structure of a database, showing the entities that compose it, the attributes of those entities and the relationships between them.

In an entity-relationship diagram, the entities represent the main objects to be stored in the database, such as "Customers", "Products" and "Orders". Each entity contains specific attributes that describe its properties, such as name, address or order date. The relationships between entities define how they interact with each other within the system, for example, a customer may place multiple orders, or an order may be associated with multiple products.

Using a DER to model Amazon's database makes it easier to understand its internal structure, ensuring that data is organized efficiently and that the relationships between entities reflect actual e-commerce operations. This is critical to optimizing both database queries and overall system performance.

III. The steps for a proper DER

The steps followed in the development of the model are described below:

1) Define components.
   - This step involves identifying the primary elements or modules that will make up the database. Components are typically larger structures that contain multiple entities and handle different aspects of the system.
2) Define entities.
   - Entities represent real-world objects or concepts that need to be stored in the database. Defining entities involves determining which data points are crucial for the system, such as Customers, Orders, or Products.
3) Define attributes for each entity.
   - Attributes describe the characteristics or properties of each entity. This step is about specifying the necessary details that need to be captured for each entity, such as customer names, product prices, and order dates.
4) Define relationships.
   - Relationships define how entities are connected to each other. In this step, you map out how the different entities interact, such as a customer placing multiple orders or a product being included in multiple orders.
5) Define types of relationships.
   - After defining the relationships, it's important to specify their types, such as one-to-one, one-to-many, or many-to-many. This ensures that the database accurately reflects the real-world interactions between entities.
6) First view of the diagram.
   - The initial version of the entity-relationship diagram (DER) is created, visually representing the entities, their attributes, and the relationships between them. This serves as a starting point for refining the design.
7) Divide many-to-many relationships.
   - Many-to-many relationships are complex and often need to be broken down into intermediary tables

or entities. This step involves dividing those relationships into simpler one-to-many relationships to make the data model more manageable.

8) Second view of the diagram.
   - After refining the relationships, a second version of the DER is created. This updated diagram includes the changes made during the optimization process, making the model more efficient and logical.

9) Obtain the data structure.
   - In this step, the structure of the data is obtained based on the final DER. This includes tables, columns, and their respective data types, forming the basis for how the data will be stored and managed in the system.

10) Define data and component properties.
    - Finally, the properties of the data (ex, data types, constraints) and components (ex, indexes, keys) are defined. This step ensures that the database will function correctly and efficiently by applying the appropriate rules and optimizations.

1) Define components.
   Naturally, the basis for this is born between
   - Users Customers: Interact with the site to search, evaluate, and purchase products. They can leave reviews and ratings. Sellers: Offer products on the platform.
   - Shopping Cart System User Interaction: Allows users to add products and view their selection before proceeding to checkout.
   - Payment System Payment Method Interaction: Communicates with various payment gateways (such as credit cards, PayPal) to process transactions.
   - Order Management Status Update: Allows users to track the status of their orders (processing, shipped, delivered).
   - Recommendation System Personalized Suggestions: Offers product recommendations based on the user's browsing and purchasing history.
   - Review and Rating System Review Collection: Allows users to leave comments and ratings on products.
   - Logistics and Shipping Inventory Management: Ensures products are available and coordinates shipping.
   - Promotions Promotions that encourage customer loyalty, such as discounts and coupons.

Define entities.

- Customer (Usuario)
- Product (Producto)
- Category (Categoría de Producto)
- Order (Pedido)
- ShoppingCart (Carrito de Compras)
- PaymentMethod (Método de Pago)
- Review (Reseña)
- Shipping (Envío)
- Offer (Oferta)
- Seller (Vendedor)
- SearchHistory (Historial de Búsquedas)
- ProductRecommendations (Recomendaciones de Productos)
- Returns (Devoluciones)
- Coupons (Cupones)

Define attributes for each entity.

- Each entity is associated with specific attributes that define its characteristics. The key attributes for each entity are as follows:

  Customer:
  - User ID (PK), Full Name, Email, Shipping Address, Phone, Registration Date

  Product:
  - Product ID (PK), Product Name, Description, Price, Quantity Available, Category ID (FK), Shopping Cart ID (FK), Seller ID (FK)

  Category:
  - Category ID (PK), Category Name, Description

  Order:
  - Order ID (PK), Order Date, Customer ID (FK), Total Amount, Order Status, Payment Method ID (FK), Shipping ID (FK)

  Shopping Cart:
  - Cart ID (PK), Customer ID (FK)

  Payment Method:
  - Payment Method ID (PK), Payment Type, Customer ID (FK)

  Review:
  - Review ID (PK), Rating, Comment, Review Date, Customer ID (FK), Product ID (FK)

  Shipping:
  - Shipping ID (PK), Shipping Company, Shipping Date, Estimated Delivery Date, Shipping Cost

  Offer:
  - Offer ID (PK), Discount, Start Date, End Date, Product ID (FK)

  Seller:
  - Seller ID (PK), Seller Name, Seller Type, Seller Rating

  Search History:
  - Search ID (PK), Search Term, Search Date, Customer ID (FK)

  Product Recommendations:
  - Recommendation ID (PK), Customer ID (FK), Recommended Product ID (FK)

  Returns:
  - Return ID (PK), Return Date, Return Reason, Return Status, Order ID (FK)

Coupons:

- Coupon ID (PK), Discount Code, Discount Value, Expiration Date, Applicable Product ID (FK)

- Define relationships.
  The following table shows the relationships that the entities have with each other so we can then see what type they are related to.

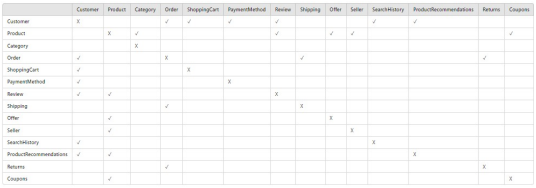

Fig. 1. Entity Relationship Table

- Define types of relationships.

Relationship	Type
Customer - order	1 to many
Customer - ShoppingCart	1 to 1
Customer - PaymentMethod	1 to many
Customer - Review	1 to many
Customer - SearchHistory	1 to many
Customer - ProductRecomendations	1 to many
Product - Category	Many to 1
Product - ShopingCart_Product	Many to many
Product - Review	1 to many
Product - Ofter	1 to many
Product - Cupons	1 to many
Seller - Product	1 to many
Order - PaymentMethod	1 to 1
Order - Shipping	1 to 1
Order - Returns	1 to many

Fig. 2. Entity Relationship Table

- First view of the diagram.

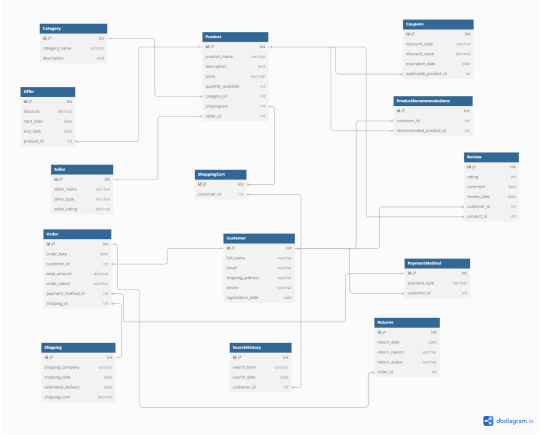

Fig. 3. Entity Relationship Table

- Divide many-to-many relationships.
  Within our first design (DER), we find that between the entities ”Products” and ”ShoppingCart” there is a many-to-many relationship since a shopping cart can contain many products and a product can be in many carts. We must eliminate this relationship since it can generate problems in the queries. That is why we implemented a new entity called ”ShoppingCart_Product” with the following attributes:
  - **cart_id**: A foreign key referencing the ShoppingCart entity to identify which shopping cart the product belongs to.
  - **product_id**: A foreign key referencing the Product entity to identify which product is being added to the shopping cart.
  - **quantity**: An integer attribute indicating the number of units of the product that the customer wants to purchase.

  In this way, the relationships will be one-to-many.
- Second view of the diagram.

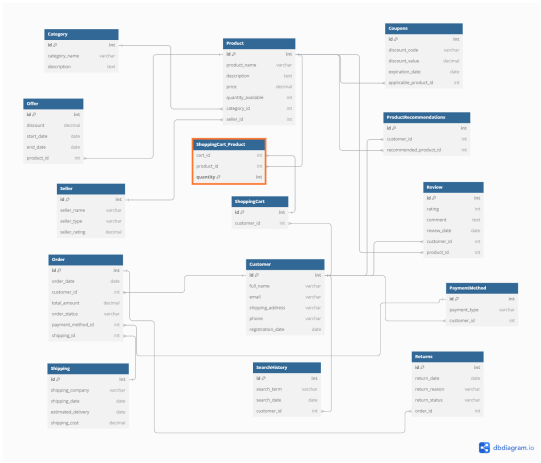

Fig. 4. Entity Relationship Table

- Obtain the data structure.
  - **Customer** (Cliente):
    - id (int, PK), full_name (varchar), email (varchar), shipping_address (varchar), phone (varchar), registration_date (date)
  - **Product** (Producto):
    - id (int, PK), product_name (varchar), description (text), price (decimal), quantity_available (int), category_id (int, FK → Category.id), seller_id (int, FK → Seller.id)
  - **Category** (Categoría):
    - id (int, PK), category_name (varchar), description (text)
  - **Order** (Pedido):
    - id (int, PK), order_date (date), customer_id (int, FK → Customer.id), total_amount (decimal), order_status (varchar), payment_method_id (int, FK

→ PaymentMethod.id), shipping_id (int, FK → Shipping.id)

- **ShoppingCart** (Carrito de Compras):
  - id (int, PK), customer_id (int, FK → Customer.id)
- **ShoppingCart_Product**:
  - cart_id (int, FK → ShoppingCart.id), product_id (int, FK → Product.id), quantity (int), PK: (cart_id, product_id)
- **PaymentMethod** (Método de Pago):
  - id (int, PK), payment_type (varchar), customer_id (int, FK → Customer.id)
- **Review** (Opinión):
  - id (int, PK), rating (int), comment (text), review_date (date), customer_id (int, FK → Customer.id), product_id (int, FK → Product.id)
- **Shipping** (Envío):
  - id (int, PK), shipping_company (varchar), shipping_date (date), estimated_delivery (date), shipping_cost (decimal)
- **Offer** (Oferta):
  - id (int, PK), discount (decimal), start_date (date), end_date (date), product_id (int, FK → Product.id)
- **Seller** (Vendedor):
  - id (int, PK), seller_name (varchar), seller_type (varchar), seller_rating (decimal)
- **SearchHistory** (Historial de Búsquedas):
  - id (int, PK), search_term (varchar), search_date (date), customer_id (int, FK → Customer.id)
- **ProductRecommendations** (Recomendaciones de Productos):
  - id (int, PK), customer_id (int, FK → Customer.id), recommended_product_id (int, FK → Product.id)
- **Returns** (Devoluciones):
  - id (int, PK), return_date (date), return_reason (varchar), return_status (varchar), order_id (int, FK → Order.id)
- **Coupons** (Cupones):
  - id (int, PK), discount_code (varchar), discount_value (decimal), expiration_date (date), applicable_product_id (int, FK → Product.id)

Entity	Attribute	Data Type	Constraints
Customer	id	int	PK, Auto Increment
	full_name	varchar(255)	NOT NULL
	email	varchar(255)	NOT NULL, UNIQUE
	shipping_address	varchar(500)	NOT NULL
	phone	varchar(15)	NOT NULL
	registration_date	date	NOT NULL
Product	id	int	PK, Auto Increment
	product_name	varchar(255)	NOT NULL
	description	text	
	price	decimal(10, 2)	NOT NULL
	quantity_available	int	NOT NULL
	category_id	int	FK (Category.id)
	seller_id	int	FK (Seller.id)
Category	id	int	PK, Auto Increment
	category_name	varchar(255)	NOT NULL
	description	text	
Order	id	int	PK, Auto Increment
	order_date	date	NOT NULL
	customer_id	int	FK (Customer.id)
	total_amount	decimal(10, 2)	NOT NULL
	order_status	varchar(50)	NOT NULL
	payment_method_id	int	FK (PaymentMethod.id)
	shipping_id	int	FK (Shipping.id)
ShoppingCart	id	int	PK, Auto Increment
	customer_id	int	FK (Customer.id)
ShoppingCart_Product	cart_id	int	FK (ShoppingCart.id)
	product_id	int	FK (Product.id)
	quantity	int	
PaymentMethod	id	int	PK, Auto Increment
	payment_type	varchar(50)	NOT NULL
	customer_id	int	FK (Customer.id)
Review	id	int	PK, Auto Increment
	rating	int	NOT NULL
	comment	text	
	review_date	date	NOT NULL
	customer_id	int	FK (Customer.id)
	product_id	int	FK (Product.id)
Shipping	id	int	PK, Auto Increment
	shipping_company	varchar(255)	NOT NULL
	shipping_date	date	
	estimated_delivery	date	
	shipping_cost	decimal(10, 2)	
Offer	id	int	PK, Auto Increment
	discount	decimal(10, 2)	NOT NULL
	start_date	date	NOT NULL
	end_date	date	NOT NULL
	product_id	int	FK (Product.id)
Seller	id	int	PK, Auto Increment
	seller_name	varchar(255)	NOT NULL
	seller_type	varchar(50)	
	seller_rating	decimal(3, 2)	
SearchHistory	id	int	PK, Auto Increment
	search_term	varchar(255)	NOT NULL
	search_date	date	NOT NULL
	customer_id	int	FK (Customer.id)
ProductRecommendations	id	int	PK, Auto Increment
	customer_id	int	FK (Customer.id)
	commended_product_	int	FK (Product.id)
Returns	id	int	PK, Auto Increment
	return_date	date	NOT NULL
	return_reason	varchar(255)	
	return_status	varchar(50)	
	order_id	int	FK (Order.id)
Coupons	id	int	PK, Auto Increment
	discount_code	varchar(50)	NOT NULL
	discount_value	decimal(10, 2)	NOT NULL
	expiration_date	date	NOT NULL
	applicable_product_id	int	FK (Product.id)

Fig. 5. Entity Relationship Table

- IV. Define data and component properties

Conclusions

The implementation of entity-relationship diagrams (DER) is essential for effective database design, especially in e-commerce platforms such as Amazon. DERs provide a clear visual representation of the entities involved in the system, as well as their relationships and attributes. This facilitates the understanding of the database structure and ensures that all components are interconnected in a logical and coherent manner.

In the case of Amazon, the complexity of the data ecosystem demands a well-planned and structured approach. The proposal of a database that includes key entities such as “Customers”, “Products”, “Orders” and “Shopping Carts” allows to efficiently manage daily operations and optimize the user experience. By identifying relationships, such as those between products and shopping carts, common redundancy issues can be resolved and referential integrity maintained.

References

- Edraw Software. (n.d.). *ER Diagram*. Recuperado de https://www.edrawsoft.com/es/er-diagram/
- Universidad de Buenos Aires. (n.d.). *Diagrama Entidad Relación*. Recuperado de https://repositorio.ub.edu.ar/handle/123456789/5155