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)
- Order-Items (Organizazdor-items)

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:
  - id (int, PK), full name (varchar), email (varchar, UNIQUE), shipping address (varchar), phone (varchar), registration date (date)

  Product:
  - id (int, PK), product name (varchar), description (text), price (decimal), quantity available (int), category id (int, FK $\rightarrow$ Category.id), seller id (int, FK $\rightarrow$ Seller.id)

  Category:
  - id (int, PK), category name (varchar), description (text)

  Order:
  - id (int, PK), total amount (decimal), order status (varchar), customer id (int, FK $\rightarrow$ Customer.id), payment method id (int, FK $\rightarrow$ PaymentMethod.id), shipping id (int, FK $\rightarrow$ Shipping.id)

  Shopping Cart:
  - id (int, PK), customer id (int, FK $\rightarrow$ Customer.id)

  Shopping Cart Product:
  - cart id (int, FK $\rightarrow$ ShoppingCart.id), product id (int, FK $\rightarrow$ Product.id), quantity (int), PK: (cart id, product id)

  Payment Method:
  - id (int, PK), payment type (varchar), customer id (int, FK $\rightarrow$ Customer.id)

  Review:
  - id (int, PK), rating (int), comment (text), review date (date), customer id (int, FK $\rightarrow$ Customer.id), product id (int, FK $\rightarrow$ Product.id)

  Shipping:
  - id (int, PK), shipping company (varchar), shipping date (date), estimated delivery (date), shipping cost (decimal)

  Offer:
  - id (int, PK), discount (decimal), start date (date), end date (date)

  Seller:

– id (int, PK), seller name (varchar), seller type (varchar), seller rating (decimal)

Search History:

– id (int, PK), search term (varchar), search date (date), customer id (int, FK → Customer.id)

Product Recommendations:

– id (int, PK), customer id (int, FK → Customer.id), recommended product id (int, FK → Product.id), recommendation date (date)

Returns:

– id (int, PK), return date (date), return reason (varchar), return status (varchar), order item id (int, FK → Order Items.id)

Coupons:

– id (int, PK), discount code (varchar), discount value (decimal), expiration date (date)

Order Items:

– id (int, PK), order id (int, FK → Order.id), product id (int, FK → Product.id), quantity (int), price at purchase (decimal), coupon id (int, FK → Coupons.id), offer id (int, FK → Offer.id)

- 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.

	Customer	Product	Category	Order	ShoppingCart	ShoppingCartProduct	PaymentMethod	Review	Shipping	Offer	Seller	SearchHistory	ProductRecommendations	Returns	Coupons	OrderItems
Customer				x	x		x	x				x	x			
Product			x			x		x		x	x		x		x	x
Category		x														
Order	x						x		x							x
ShoppingCart	x					x										
ShoppingCartProduct		x			x											
PaymentMethod	x			x												
Review	x	x														
Shipping				x												
Offer		x														
Seller		x														
SearchHistory	x															
ProductRecommendations	x	x														
Returns																x
Coupons		x														x
OrderItems		x		x										x	x	

Fig. 1. Entity Relationship Table

- Define types of relationships.

Entidad 1	Entidad 2	Tipo de Relación	Descripción
Customer	Order	1:N	Un cliente puede hacer múltiples pedidos, pero un pedido pertenece a un solo cliente.
Customer	Shopping Cart	1:1	Cada cliente tiene un único carrito de compras.
Customer	Payment Method	1:N	Un cliente puede tener múltiples métodos de pago, pero un método de pago pertenece a un solo cliente.
Customer	Review	1:N	Un cliente puede escribir múltiples reseñas, pero cada reseña pertenece a un solo cliente.
Customer	Search History	1:N	Un cliente puede tener múltiples búsquedas registradas, pero cada búsqueda pertenece a un solo cliente.
Customer	Product Recommendations	1:N	Un cliente puede recibir múltiples recomendaciones de productos, pero cada recomendación pertenece a un solo cliente.
Product	Category	N:1	Un producto pertenece a una sola categoría, pero una categoría puede tener múltiples productos.
Product	Review	1:N	Un producto puede tener múltiples reseñas, pero cada reseña pertenece a un solo producto.
Product	Shopping Cart Product	N:M	Un producto puede estar en múltiples carritos y un carrito puede contener múltiples productos.
Product	Order Items	N:M	Un producto puede estar en múltiples pedidos y un pedido puede contener múltiples productos.
Product	Coupons	N:1	Un cupón puede aplicarse a un solo producto, pero un producto puede tener múltiples cupones disponibles.
Category	Product	1:N	Una categoría puede tener múltiples productos, pero cada producto pertenece a una sola categoría.
Order	Customer	N:1	Un pedido pertenece a un solo cliente, pero un cliente puede hacer múltiples pedidos.
Order	Payment Method	N:1	Un pedido utiliza un solo método de pago, pero un método de pago puede usarse en múltiples pedidos.
Order	Shipping	1:1	Un pedido tiene un único envío asociado.
Order	Order Items	1:N	Un pedido puede contener múltiples productos.
Shopping Cart	Customer	1:1	Cada cliente tiene un único carrito de compras.
Shopping Cart	Shopping Cart Product	1:N	Un carrito de compras puede contener múltiples productos.
Shopping Cart Product	Product	N:M	Un producto puede estar en múltiples carritos, y un carrito puede contener múltiples productos.
Payment Method	Customer	N:1	Un cliente puede tener múltiples métodos de pago.
Payment Method	Order	1:N	Un pedido utiliza un solo método de pago, pero un método de pago puede usarse en múltiples pedidos.
Review	Customer	N:1	Una reseña pertenece a un solo cliente, pero un cliente puede hacer múltiples reseñas.
Review	Product	N:1	Una reseña pertenece a un solo producto, pero un producto puede tener múltiples reseñas.
Shipping	Order	1:1	Cada pedido tiene un único envío.
Offer	Product	1:N	Un producto puede tener múltiples ofertas, pero cada oferta está asociada a un solo producto.
Seller	Product	1:N	Un vendedor puede vender múltiples productos, pero un producto pertenece a un solo vendedor.
Search History	Customer	N:1	Una búsqueda pertenece a un solo cliente, pero un cliente puede tener múltiples búsquedas registradas.
Product Recommendations	Customer	N:1	Una recomendación pertenece a un solo cliente, pero un cliente puede recibir múltiples recomendaciones.
Returns	Order Items	1:1	Cada devolución está asociada a un único producto dentro de un pedido.
Coupons	Order Items	N:1	Un cupón puede aplicarse a un solo producto dentro de un pedido.
Order Items	Order	N:1	Un pedido puede contener múltiples productos, pero cada producto dentro del pedido pertenece a un solo pedido.
Order Items	Product	N:1	Un producto dentro de un pedido pertenece a un solo producto, pero un producto puede estar en múltiples pedidos.

Fig. 2. Entity Relationship Table

- First view of the diagram.
- 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

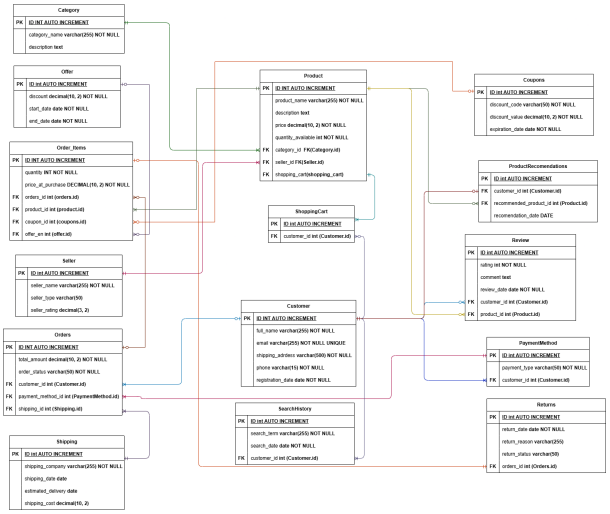

Fig. 3. Entity Relationship Table

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.

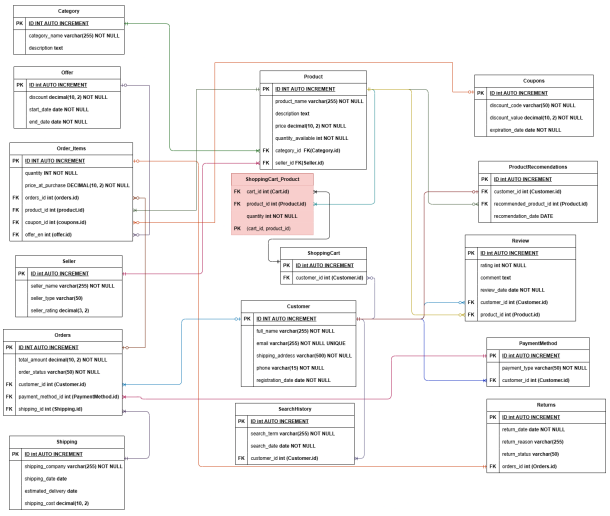

Fig. 4. Entity Relationship Table

- Obtain the data structure.

- **Customer** (Cliente):
  - id (int, PK), full_name (varchar), email (varchar, UNIQUE), 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), total_amount (decimal), order_status (varchar), customer_id (int, FK → Customer.id), 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)
- **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), recommendation_date (date)
- **Returns** (Devoluciones):
  - id (int, PK), return_date (date), return_reason (varchar), return_status (varchar), order_item_id (int, FK → Order_Items.id)
- **Coupons** (Cupones):
  - id (int, PK), discount_code (varchar), discount_value (decimal), expiration_date (date)
- **Order_Items** (Detalles del Pedido):
  - id (int, PK), order_id (int, FK → Order.id), product_id (int, FK → Product.id), quantity (int), price_at_purchase (decimal), coupon_id (int, FK → Coupons.id), offer_id (int, FK → Offer.id)

- IV. DEFINE DATA AND COMPONENT PROPERTIES

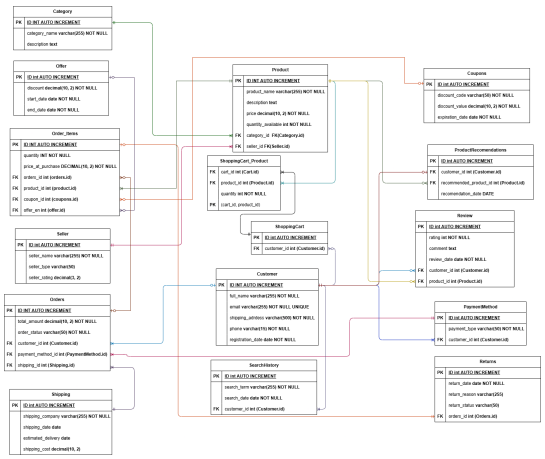

Fig. 5. Entity Relationship Table

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