Cianna Grama
Professor Peitzsch
COM 303: Database Systems
10 May 2025

Zara Database Write-Up

Introduction to Zara

Zara Overview

Zara is an international fashion retailer that follows the fast-fashion model of bringing fashion trends from runways to stores quickly. Zara was founded in 1975 by Amancio Ortega and Rosalia Mera in Galicia, Spain. Zara sells clothing, accessories, and shoes for men, women, and children that are stylish, affordable, and updated often for their customers. New items appear in Zara stores as often as twice a week. Zara is a large brand with thousands of stores in over 90 countries. Among the physical stores, Zara also has a large online selection of items.

Structures of Zara: Informing the Database Design

Zara has some unique structures that we had to keep in mind when designing the database. First, Zara is an international enterprise, so we needed to ensure the database could account for stores in different countries. Second, Zara has one online store for all countries, with slightly different interfaces for each country. There is technically only one website, so the online stores have the same backend for each country. The differences in the interfaces are in prices, currency types, and languages to account for those differences in different countries. The online stores use the same website and function the same in terms of usage and structure. Third, Zara has multiple locations in certain cities, so we ensured the database could handle that. Lastly, Zara only carries items that are their brand, so "brand" was not a part of our E-R model.

Logical Structures of the Database

First, the online store inventory and physical store inventories are functionally the same in the database. The online store inventory has much higher quantities of items, but they act the same. In our database, there is technically a location that holds the online store inventory. It is technically a warehouse, but functionally a store, so it is logically considered a store in our database. This location is where all online orders are shipped out from. Second, the product ID in our database is functionally the same as a universal product code (UPC), but the UPC is externally associated outside of the database. This reduces the complications of UPCs in the database and allows the product ID to be incremented within the database.
The logic of purchases, restocks, and returns is as follows in our database: When a purchase is made, the quantity is an attribute in itemsPurchased, and that quantity is subtracted from the

storeInventory quantity. Returns are handled by a decrease of the quantity value in itemsPurchased and an increase in the storeInventory quantity value of the store the item was purchased from. Store restocks are handled by a quantity checker. When the quantity in a store goes below 15 items, then a restock request is sent to the warehouse that supplies said store, and 100 new items are added to the inventory of the store and removed from the inventory of the warehouse.

Business Decisions

First, each product has a company-wide set price that is reflected in the price attribute of the products table. However, there are price differences for an item in different stores. This is the result of sales, which are an attribute in storeInventory. When an item is purchased, the price in product is multiplied by the amount on sale, and that purchase price is recorded in itemsPurchased under the attribute price_at_purchase. This ensures that different stores can have different product prices, and ensures that the purchase price is recorded if that data needs to be accessed. Second, physical store purchases and online purchases can be returned to any physical store. Once that item is returned, the store where the item was bought is found through a query and sent to that location. However, physical purchases cannot be returned online. Third, shipping is done through online orders only. Even if a customer orders a product online in a physical store, that goes into the system as an online order and is functionally the same. Fourth, our Zara database does not identify non-members to perform business analytic queries on specific people. Members have individualized data analytics because they agree to that as a part of our membership program, but non-members are not subject to the same analytics. Non-member purchases as a whole are still used in our analytics, but not individual analytics. Fifth, every warehouse carries every product, as previously discussed. And lastly, when a new store is created, they are automatically assigned a warehouse to supply them.

Data Generation

We generated data by using the python Faker library to randomly generate entity data using realistic attribute values. Then, we used the entity data to populate the relationships, adding on some attributes with the Faker library generation when necessary. Then, we tested queries and altered our tables slightly based on our queries. The main alterations were removing derived attributes and removing unnecessary attributes. For example, we removed the total order price because it is derived and we removed warehouse_name as an attribute of warehouse because it is unnecessary.

We also altered the data generation to ensure logical accuracy. In customerPayment, we made sure the data populates so that every payment has a customer, and every member and online guest customer has assigned payments in the database. In orderPayment, we generated data so that order, payment, and customer are correctly corresponding. Each customer needs to have an order, each order needs to have a payment, but the payment of the order needs to be one

that the customer has. We ensured this by populating all the data generation except for orderPayment, committing that data population, querying into the database to find the customer associated with each order, finding the payment associated with that customer, and then placing the associated payment and order into the orderPayment table. Lastly, in supplies, we ensured that every store is supplied by a warehouse, but that warehouses supply more than one store.

Significant Problems and Solutions

One major difficulty was determining how to create the logical structure of the E-R. We had many drafts and attempts at relating the data and went through different versions before landing on our official E-R. I discussed the problems within each entity and relationship in the E-R section. We solved this problem by asking lots of questions, working together, brainstorming separately, and then coming together to work to ensure our ideas were fresh.

Another major difficulty was data generation. Ensuring that the data populated correctly was vital to being able to test our database, and we ran into difficulties populating the relationship data. The entity data was easier to generate because we used the python Fake library to simulate real data, but the relationships could not be so easily generated. We solved this problem by making lists of the created IDs of the entities and then using those lists to populate the relationship data. Additionally, we had different approaches to hierarchy data populations, and it resulted in some confusion. I took the approach of generating bottom-up from the hierarchy to make sure that all the tables were evenly populated, and Carly took the approach of populating from top-down and randomly assigning tables in the hierarchy the generated IDs would fall under. We discovered that her method worked best for populating huge hierarchies, like products, and that my method worked best for populating smaller hierarchies, like the members hierarchy.