Database in Practice - HW4

Mike Rabayda

April 3, 2024

1 3NF Normalized Schema for Relational Database

The PostgreSQL database design adheres to the principles of Third Normal Form (3NF) to ensure the elimination of redundancies and dependencies in the data structure. Achieving 3NF involves structuring the database so that:

- All tables are in the Second Normal Form (2NF).
- There are no transitive dependencies, so non-primary key columns do not depend on other non-primary key columns.

This normalization process enhances data integrity and consistency within the relational database by ensuring that each piece of information is stored only once, thus reducing the potential for anomalies in data modification operations.

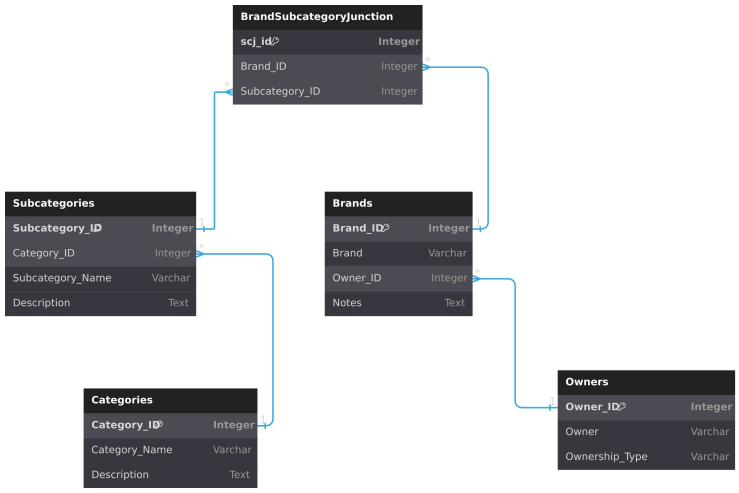

Figure 1: 3NF Normalized Schema for the Brand Information Database

Figure 1 below provides a visual representation of the normalized schema made for the storage and retrieval of Brand information within the PostgreSQL

database. This schema illustrates the relationships between Categories, Subcategories, Brands, Owners, and the BrandSubcategoryJunction, ensuring a clear and logical organization of data in compliance with 3NF principles.

2 PostgreSQL Setup

Cleaning

The PostgreSQL database was initialized with a dataset initially prepared for a MongoDB database consisting of approximately 600 rows, containing Brand, Owner, Ownership Type, Category (of product), and Notes. The key steps in the data cleaning process before organizing the information into 3NF included:

- Exclusion of entries without *Ownership Type* to maintain the relational database's integrity.
- Removal of problematic brands with missing owner information, specifically brands with IDs 145 (Dole), 226 (Guayaki), 410 (Rael), and 514 (Tabasco).
- Retention of duplicate brands to reflect their presence across multiple product categories, considering the potential for brands like Post to appear in various contexts.

Final Copy Commands

The data was subsequently cleaned in Microsoft Excel and organized based on the 3NF schema discussed earlier. The cleaned data was imported into the PostgreSQL database from various .csv's using the following commands in the PostgreSQL CLI:

```
\copy Categories FROM '/home/mrabayda/public_html/BrandOwnership/Databases HW2
- Categories.csv' DELIMITER ',' CSV HEADER;
\copy Subcategories FROM '/home/mrabayda/public_html/BrandOwnership/Databases HW2
- Subcategories.csv' DELIMITER ',' CSV HEADER;
\copy Owners FROM '/home/mrabayda/public_html/BrandOwnership/Databases HW2
- Brands - Databases HW5 - Owners-3.csv' DELIMITER ',' CSV HEADER;
\copy Brands FROM '/home/mrabayda/public_html/BrandOwnership/Databases HW2
- Brands - Databases HW5 - Brands.csv' DELIMITER ',' CSV HEADER;
\copy BrandSubcategoryJunction FROM '/home/mrabayda/public_html/BrandOwnership/
Databases HW2- SubCategoryJunction_fix.csv' DELIMITER ',' CSV HEADER;
```

Index Creation

To support efficient querying, particularly for brand-related searches, an index was created on the *Brand* column within the *Brands* table:

```
CREATE INDEX idx_brands_brand ON Brands(Brand);
```

3 MongoDB Setup

MongoDB Document Structure

The structure of a typical document within the `Brands` collection is represented in JSON format below:

```
{
  "_id": ObjectId("unique_id_generated_by_mongo"),
  "Brand": "Example Brand (indexed ascending)",
  "Owner": "Example Owner",
  "Ownership Type": "Megacorp, Private Equity, etc.",
  "Product Type": "Food",
  "Notes": "Some relevant notes about the brand or product."
}
```

Each document within the collection uniquely identifies a brand, characterized by attributes such as the brand's name, owner, ownership type, product type, and any notes. The `"Brand"` field is indexed in ascending order to optimize query performance, particularly for searches and retrievals based on brand names.

The MongoDB database was populated with data sourced from three separate .csv files, encompassing Food Product Brands, Personal Care Products, and Fitness Products. This data was amalgamated into a single collection with an additional *Product Type* column, which serves as the *Category* within the Mongo dataset.

Cleaning

The cleaning process primarily focused on standardizing the *Ownership Type* field by:

- Consolidating various versions of 'mega corporation' into "Megacorp".
- Maintaining the distinction between different ownership types

Import Command

The following command was used to import the consolidated CSV data into the MongoDB collection:

```
mongoimport --db=mrabayda --collection=Brands --type=csv --file=
"path/to/Mongo CSV Brands - Master_Master.csv" --headerline
```

Index Creation

An ascending index was created on the *Brand* field to enhance query performance:

```
db.Brands.createIndex({Brand: 1});
```

4 Sample Queries

This section details three sample queries executed against both the PostgreSQL and MongoDB databases. These queries illustrate the retrieval of data based on specific criteria: brands owned by "Clorox", products related to "Fitness", and products under the "Dove" brand.

4.1 Querying Brands Owned by Clorox

This query retrieves all brand entries where the owner is specified as "Clorox". In MongoDB, this is achieved with a simple find operation on the "Owner" field. In PostgreSQL, the query involves joining the Brands table with the Owners table to filter based on the owner's name.

MongoDB:

```
db.Brands.find({"Owner": "Clorox"});
```

PostgreSQL:

```
SELECT B.Brand, O.Owner, O.Ownership_Type, C.Category_Name AS Category, S.Subcategory_Name A
FROM Brands B
INNER JOIN Owners O ON B.Owner_ID = O.Owner_ID
INNER JOIN BrandSubcategoryJunction BSJ ON B.Brand_ID = BSJ.Brand_ID
INNER JOIN Subcategories S ON BSJ.Subcategory_ID = S.Subcategory_ID
INNER JOIN Categories C ON S.Category_ID = C.Category_ID
WHERE O.Owner = 'Clorox';
```

4.2 Querying Fitness Products

This query focuses on retrieving all entries categorized under "Fitness" products. In MongoDB, this involves filtering documents based on the "Product Type". In PostgreSQL, the query requires a join operation across multiple tables to reach the category information.

MongoDB:

```
db.Brands.find({"Product Type": "Fitness"});
```

PostgreSQL:

```
SELECT B.Brand, O.Owner, O.Ownership_Type, C.Category_Name AS Category, S.Subcategory_Name A
FROM Categories C
INNER JOIN Subcategories S ON C.Category_ID = S.Category_ID
INNER JOIN BrandSubcategoryJunction BSJ ON S.Subcategory_ID = BSJ.Subcategory_ID
INNER JOIN Brands B ON BSJ.Brand_ID = B.Brand_ID
INNER JOIN Owners O ON B.Owner_ID = O.Owner_ID
WHERE C.Category_Name = 'Fitness Products'
LIMIT 5;
```

4.3 Querying Dove Brand

This query retrieves all entries for the "Dove" brand. The MongoDB query filters documents by the "Brand" field, while the PostgreSQL query involves a condition in the WHERE clause to filter the results by the brand name.

MongoDB:

```
db.Brands.find({"Brand": "Dove"});
```

PostgreSQL:

```
SELECT B.Brand, O.Owner, O.Ownership_Type, C.Category_Name AS Category, S.Subcategory_Name A
FROM Brands B
INNER JOIN Owners O ON B.Owner_ID = O.Owner_ID
INNER JOIN BrandSubcategoryJunction BSJ ON B.Brand_ID = BSJ.Brand_ID
INNER JOIN Subcategories S ON BSJ.Subcategory_ID = S.Subcategory_ID
INNER JOIN Categories C ON S.Category_ID = C.Category_ID
WHERE B.Brand = 'Dove';
```

Each of these queries demonstrates the flexibility and capabilities of querying in both MongoDB and PostgreSQL, showcasing how similar data retrieval objectives are achieved in a document-based system versus a relational database system.