Gabriela Masak

D597 – Data Management

Task 1: Relational Database

9/4/2024

Design Document: Scenario 2

How businesses function is rapidly evolving in the modern digital age. Traditionally brick-and-mortar stores are expanding to online venues. If their database does not evolve with the business, they risk severe issues with data inconsistencies and security risks that could threaten the stability and reliability of the entire database infrastructure. As such, many retail businesses are focusing their efforts on building database management systems (DBMS) that can efficiently handle the scalability, flexibility, and efficiency necessary, while also maintaining their ethical and legal security standards. These DBMS can be defined by their data model, entities, attributes, data organization, scalability, security, and integration capabilities--each catering to the unique business problems they collectively solve.

EcoMart is a business facing exponential growth and moving performance goals. Data collected by EcoMart consists of product information, order information, and transaction financials. This information is frequently added to, modified, and manipulated, the means of which cannot hinder speed or reliability of the platform without disrupting the business. Addressing its transforming performance goals, EcoMart may seek to use this data to determine where to focus their efforts, pruning expensive or underperforming areas while growing their more profitable areas. Outside of analytics and business intelligence, security must also be considered to protect private and sensitive data from competitors, miscreants, or human error. This can be solved by implementing a new DBMS with the ability to scale while integrating with existing systems, removing issues like performance bottlenecks, security vulnerabilities, and data inconsistencies.

An online transaction processing (OLTP) database, also classified as a relational database, would allow for data to be stored and manipulated for tactical and strategic business purposes, while also furnishing the business with the ability to directly monitor inventory, sustainability certifications, and transaction data. For EcoMart, the relational database will be organized by the entity relationship diagram (ERD) seen in Figure 1.

A screenshot of a computer

Description automatically generated

Figure 1: EcoMart ERD

The ERD is comprised of four entities: items, orders, order details, and locality. Each of these four tables contains relevant data, organized by attributes. For items, these attributes are item identification numbers, item type, item price, and item cost. For orders, these attributes are order identification numbers, sales channel, order priority, order date, shipment date, and country identification number. For order details, these attributes are order identification numbers, item identification number, units sold, total revenue, total cost, and total profit. For locality, these attributes are country identification numbers, country name, and region cost.

Originally, there were fourteen attributes in the flat file. Adding identification numbers for countries and items increased this number. Choosing how to manage and organize these attributes into relevant entities was completed with data security and future analysis in mind. I defined the potential users as retail employees and executives, each using the data for different purposes. A retail employee would not need access to the full scope of data for their role and responsibilities—just general transaction information for processing, packing, and shipping orders. As such, the order entity was best served as containing solely basic transaction information, separate from data that would be used by executives to analyze financial metrics, whose related attributes form the order detail entity. Recognizing that data for specific items and countries are less liable to change compared to that of real-time transactions, the last two entities were respectively formed. Organizing the data into entities can make analysis of the data more efficient and accurate, presenting refined, smaller sets of data for queries, as well as aiding in explaining the relationships between the data.

The advantage of using a relational database over a spreadsheet or flat file is found in the ability to explore the relationships between the entities to perform complex queries and advanced data analysis with SQL (Structured Query Language). This capability is limited in spreadsheets, which are not designed for handling large-scale data manipulation or complex queries, and where analysis is much slower and time consuming. These relationships are stipulated by the primary and foreign keys of the entities. In Figure 1, relationships are visualized by crow’s foot notation and connecting lines. The crow’s foot notation represent the type of relationship: one-to-one, one-to-many, or many-to-many.

EcoMart, hoping to increase profits by decreasing focus on less strategic, expensive, or impractical areas, must first determine what these areas are. This can be achieved by a series of queries that weigh different areas of the business against each other to expose the best and worst performing of the group. A query into the item type that generates the least profit, a query into different regions’ shipping times and revenue, and a query into the financial differences of online and offline shopping. The business may use the results of these queries, included in Figures 9, 10, and 11 respectively, to decide where and how to optimize their business.

While the primary focus of many businesses may be extracting business intelligence from data, security is also a concern that is alleviated by the implementation of relational databases. Transactional databases support real-time data processing and ensure data integrity through Atomicity, Consistency, Isolation, and Durability (ACID) properties (Kaur, 2019). For example, the inclusion of automated identification numbers for items, orders, and countries allows for security and anonymity of proprietary information as well as the exclusion of data redundancy. Ensuring that transactions are either fully completed or not, refusing to accept partial completion, lends to atomicity and data accuracy. Segregating user permissions and access is also important. Retail employees, for example, would only need to access basic order details to fulfill their work requirements and would only be able to access the order entity within the database. Even good intentions can lead to costly errors, so effective security measures dictate that access should be supplied on an as-needed basis. Not only does this ensure greater accuracy, but also prevents bad actors from manipulating or stealing data.

The choice to use a relational database over a spreadsheet or flat file also provides solutions regarding scalability. Spreadsheets are limited in the number of data entries. These limits are almost nonexistent with a relational database, especially with the advent of cloud storage and computing, like Amazon RDS or Azure SQL Database. Users around the globe would be able to access and analyze the data, which is imperative for a growing international business. Cloud companies allow their clients to scale up or down almost instantaneously depending on business needs, a valuable set of options while EcoMart grows and transforms.

**Part 2: Implementation in WGU Virtual Lab**

1. Write script to create a database instance named “D597 Task 1” using the appropriate query language, based on the logical data model in part B. Provide a screenshot showing the script and the database instance in the platform.

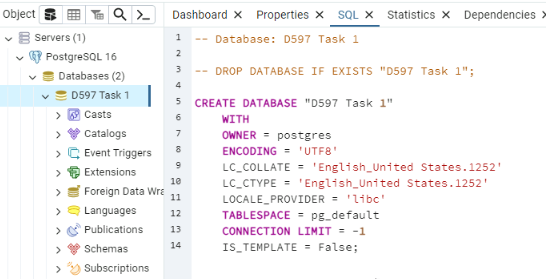


Figure 2: SQL Script to Create "D597 Task 1" Instance

1. Write script to import the data records from the chosen scenario CSV files into the database instance. Provide a screenshot showing the script and the data correctly inserted or mapped into the database.

A screenshot of a computer

Description automatically generated

Figure 3: PostgreSQL Screen to Import CVS File

After the data from the Sales\_Record.csv file is imported, the data can be viewed with a SELECT statement, as seen in Figure 4.

A screenshot of a computer

Description automatically generated

Figure 4: View of D597 Task 1 Table data

Data was then sorted into the four tables, based on the organization of the entities in the ERD in Figure 1.

This was accomplished by a series of SQL statements for each of the four tables, after which the relevant data could be viewed for each table.

INSERT INTO item(item\_type, unit\_price, unit\_cost) SELECT DISTINCT item\_type, unit\_price, unit\_cost FROM public."D597Task1";

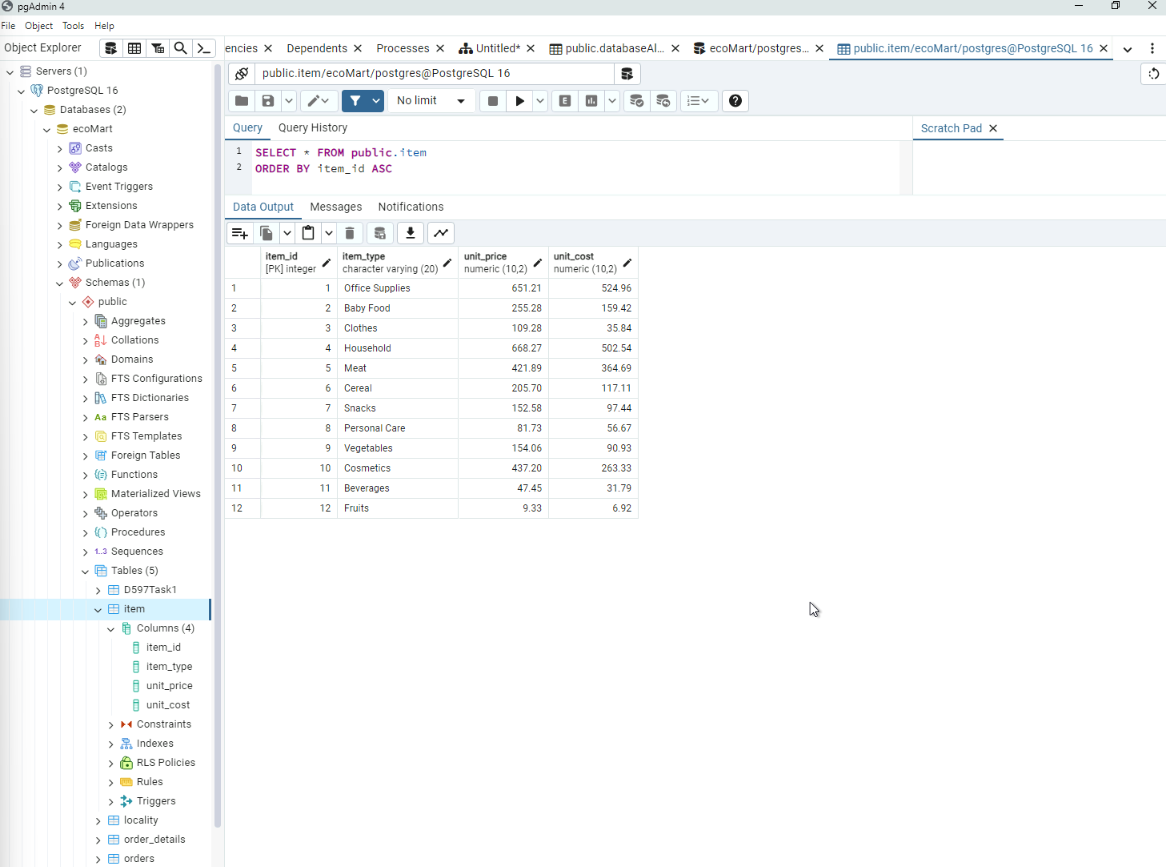


Figure 5: Item Table

INSERT INTO locality(country\_name, region\_name) SELECT DISTINCT country, region FROM public."D597Task1";

A screenshot of a computer

Description automatically generated

Figure 6: Locality Table

INSERT INTO orders(order\_id, sales\_channel, order\_priority, order\_date, ship\_date, country\_id) SELECT DISTINCT order\_id, sales\_channel, order\_priority, order\_date, ship\_date, country\_id FROM public."D597Task1" JOIN locality ON public."D597Task1".country = locality.country\_name;

A screenshot of a computer

Description automatically generated

Figure 7: Order Table

INSERT INTO order\_details(order\_id, item\_id, units\_sold, total\_revenue, total\_cost, total\_profit) SELECT DISTINCT order\_id, item\_id, units\_sold, total\_revenue, total\_cost, total\_profit FROM public."D597Task1" JOIN item ON public."D597Task1".item\_type = item.item\_type;

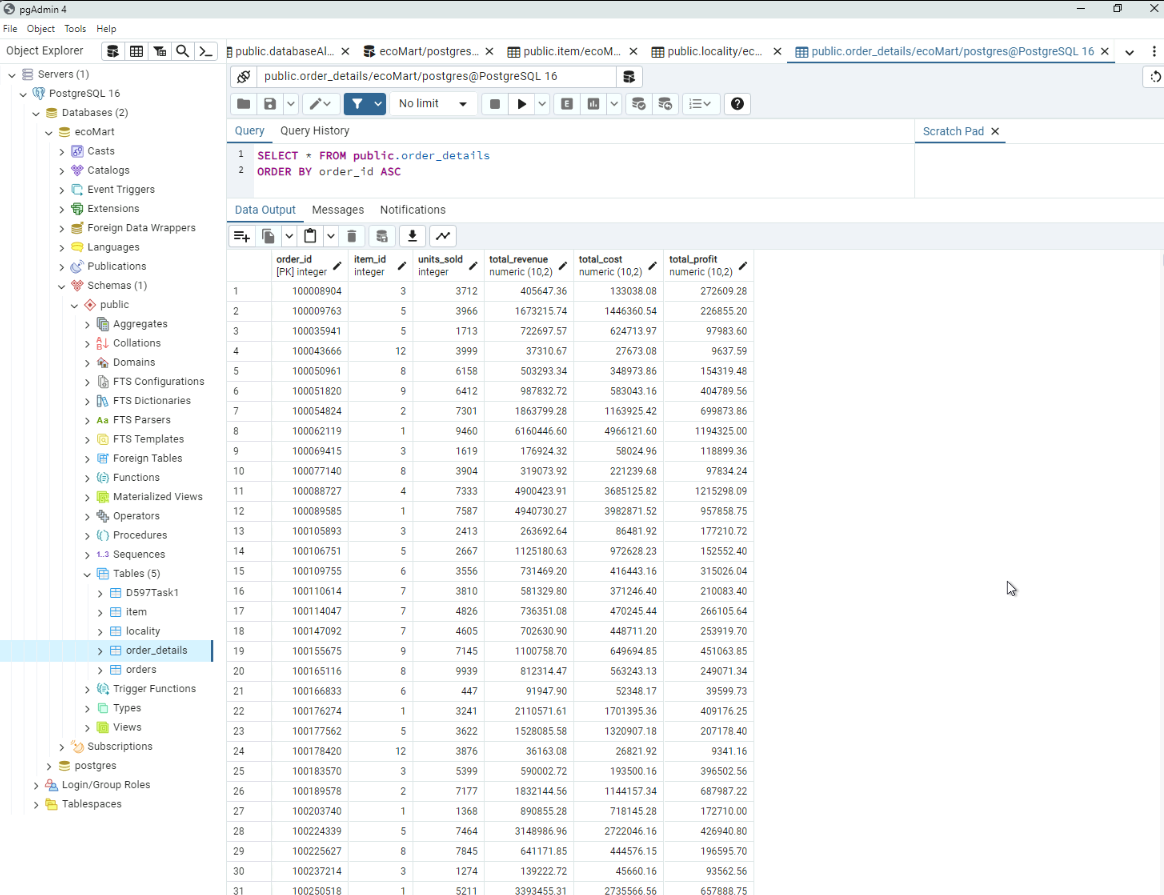


Figure 8: Order Details Table

1. Write script for **three** queries to retrieve specific information from the database that will help to solve the identified business problem. Provide a screenshot showing the script for *each* query and *each* query successfully executed.

Query #1: Item Profitability

SELECT item\_id, SUM(total\_profit / units\_sold) AS item\_profit FROM order\_details GROUP BY item\_id ORDER BY item\_profit;

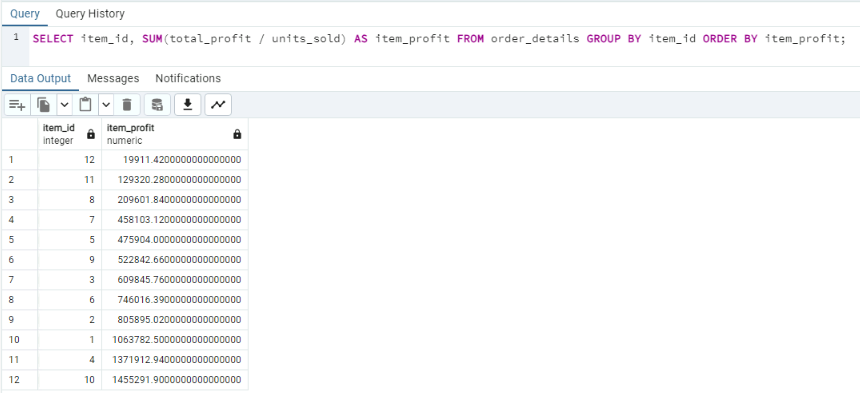


Figure 9: Results of Query 1

Query #2: Average Shipping Time by Region

SELECT region\_name, AVG(ship\_date - order\_date) AS averageTimeToShip FROM orders JOIN locality ON orders.country\_id=locality.country\_id GROUP BY region\_name;

A screenshot of a computer

Description automatically generated

Figure 10: Results of Query 2

Query #3: Online and Offline Business Comparison

SELECT sales\_channel, AVG(total\_profit) FROM public."orders" JOIN order\_details ON public."orders".order\_id = order\_details.order\_id GROUP BY sales\_channel

A screenshot of a computer

Description automatically generated

Figure 11: Results of Query 3

1. Apply optimization techniques to improve the run time of your queries from part F3, providing output results via a screenshot.

Query Optimization #1: Item Profitability

A screenshot of a computer

Description automatically generated

Figure 12: Runtime of Query 1 Before Optimization

SELECT item\_id, item\_profit

FROM (

SELECT item\_id, SUM(total\_profit / units\_sold) AS item\_profit

FROM order\_details

GROUP BY item\_id

) AS subquery

ORDER BY item\_profit;

A screenshot of a computer code

Description automatically generated

Figure 13: Runtime of Query 1 After Optimization

Query Optimization #2: Average Shipping Time by Region

A screenshot of a computer

Description automatically generated

Figure 14: Runtime of Query 2 Before Optimization

SELECT locality.region\_name, AVG(DATEDIFF(day, orders.order\_date, orders.ship\_date)) AS averageTimeToShip

FROM orders

JOIN locality ON orders.country\_id = locality.country\_id

GROUP BY locality.region\_name;

A screenshot of a computer program

Description automatically generated

Figure 15: Runtime of Query 2 After Optimization

Query Optimization #3: Online and Offline Business Comparison

A screenshot of a computer

Description automatically generated

Figure 16: Runtime of Query 3 Before Optimization

WITH sales\_data AS (

SELECT sales\_channel, total\_profit

FROM public."orders"

JOIN order\_details ON public."orders".order\_id = order\_details.order\_id

)

SELECT sales\_channel, AVG(total\_profit) AS avg\_profit

FROM sales\_data

GROUP BY sales\_channel;

A screenshot of a computer code

Description automatically generated

Figure 17: Runtime of Query 3 After Optimization

**Part 3: Presentation**

A link to the audiovisual presentation on Pantopo is provided here:

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=90e13344-83ef-4f01-9814-b1ee00faa963>

**Sources Cited**

1. Kaur, A. (2019, January 31). *ACID Properties in DBMS*. GeeksforGeeks;

GeeksforGeeks.com. <https://www.geeksforgeeks.org/acid-properties-in-dbms/>