Information Technology Institute

2024

Northwind Data Warehouse

Ahmed Hatem

Mariam Marcos

Mohamed Karam

Mostafa Mohamed

Yassa Rashad

Introduction

In today's data-centric landscape, businesses are embracing the power of data warehousing to drive informed decision-making. As we navigate this transformative era, we embark on the development journey of a Data Warehouse (DWH) built upon the esteemed NorthWind database. Our endeavor is fueled by the recognition of the pivotal role data plays in shaping strategic insights and driving operational efficiencies.

Drawing upon the robust capabilities of PostgreSQL as our chosen database management system, and wielding PGAdmin as our tool of choice for management and administration, we embark on a quest to engineer a DWH architecture that epitomizes performance, scalability, and versatility. Our mission is clear: to architect a solution that not only integrates seamlessly with existing data ecosystems but also empowers organizations to derive actionable insights from vast troves of data.

In this endeavor, we prioritize the development of a DWH architecture capable of accommodating the evolving needs of modern businesses. By harnessing the rich dataset provided by the NorthWind database, we endeavor to construct a foundational framework that fosters data-driven decision-making and fuels innovation across all facets of the organization.

As stewards of data, we recognize the inherent value encapsulated within the NorthWind database and the wealth of opportunities it presents for driving business growth and competitive advantage. Through meticulous planning, strategic implementation, and unwavering dedication to excellence, we aspire to craft a DWH architecture that not only meets the immediate analytical and reporting requirements but also lays the groundwork for future expansion and innovation.

Project Objective

Our objective is to construct a Data Warehouse (DWH) infrastructure using PostgreSQL on PGAdmin, built upon the NorthWind database. We aim to select dimension and fact tables, determine indexing strategies, and implement performance optimization techniques to create a highly responsive and scalable DWH capable of meeting complex analytical and reporting requirements.

The project taken steps

1. **Defining Business Processes:** We have comprehensively defined the core business processes that drive organizational operations and decision-making. This step lays the foundation for identifying key data requirements and determining the scope of our Data Warehouse (DWH) project.
2. **Defining Granularity for the Analysis Scope:** By establishing the granularity level for our analysis scope, we have clarified the level of detail at which data will be aggregated and analyzed within the DWH. This step ensures alignment between business needs and data representation.
3. **Defining Dimensions for Analysis Context:** We have identified and defined the dimensions that encapsulate the various attributes and characteristics relevant to our analysis context. These dimensions provide the context for organizing and categorizing data within the DWH.
4. **Defining Facts and Measurements:** Crucial to our DWH design, we have determined the facts and measurements that represent the quantitative aspects of our business processes. These metrics serve as the foundation for generating insights and making data-driven decisions.
5. **Data Warehouse Modeling (Defining the Schema):** Building upon the defined dimensions, facts, and measurements, we have created a comprehensive schema for our DWH. This schema outlines the structure and relationships between tables, laying the groundwork for data storage and retrieval.
6. **Physical Model Implementation:** With the schema in place, we have proceeded to implement the physical model of our DWH within the PostgreSQL environment using PGAdmin. This step involves creating tables, defining data types, and establishing relationships based on the schema design.

7. **Data Warehouse Indexing:** In order to optimize query performance and enhance data retrieval efficiency, we have implemented indexing strategies within our DWH. By strategically indexing columns frequently used in filtering, sorting, and joining operations, we aim to accelerate data access.
8. **Generating & Populating the Data:** We have generated and populated the DWH with relevant data extracted from the NorthWind database. This step involves data extraction, transformation, and loading (ETL), ensuring that the DWH contains accurate and up-to-date information for analysis.
9. **Writing SQL Queries to Gain Business Insights:** With the DWH populated, we have begun writing SQL queries to extract insights and derive meaningful analyses from the data. These queries are tailored to address specific business questions and objectives, enabling stakeholders to make informed decisions based on actionable insights.

Tables Overview

1. **Definition of Business Process:**
    - The customer orders products from our company, which procures them from suppliers and ships them to the customers.
    - Occasional promotions and discounts are also part of the business process.
2. **Definition of Granularity:**
    - The granularity of the fact table is defined as product per order.
3. **Dimensions:**
    - **DimCustomer:**

      Stores comprehensive information about customers, including customer key, ID, company details, contact information, and address.
    - **DimDate:**

      (Role Playing Dimension)

      Contains a wealth of date-related information such as full date, year, quarter, month, week number, day of the week, and flags for weekend and holiday.
    - **DimPromotion:**

      Captures promotion details, including promotion key, start and end dates, and discount percentage, enabling analysis of promotional effectiveness.
    - **DimShipper:**

      Houses information pertaining to shipping companies, encompassing shipper key, ID, company name, and contact details, facilitating tracking of shipping operations.
    - **DimSupplier:**

Encompasses comprehensive supplier information, including supplier key, ID, company name, contact details, and address, essential for supplier management and procurement analysis.

- **DimProduct:**

  Holds extensive product details such as product key, supplier key, product ID, name, category, description, and flags indicating quantity per unit and discontinued status, enabling product performance analysis and inventory management.

- **DimProductPrice:**

  (Mini Dimension)

  Contains historical product price information, facilitating price trend analysis and pricing strategy formulation.

- **DimTerritory:**

  Stores territory-related data, including territory key, ID, name, and region, crucial for sales territory analysis and regional performance evaluation.

- **DimEmployee:**

  (Slowly Changing Dimension_Type2)

  Houses detailed employee information such as employee key, ID, name, title, birthdate, hire date, contact information, and hierarchical relationships and tracks the change in employee’s title and manager overtime supporting workforce management and performance analysis.

- **DimEmpTerritory:**

  (Bridge Table)

  Establishes the many-to-many relationship between employees and territories, facilitating analysis of employee territory assignments and performance.

- **DimPayMethodSaleChannel:**

  (Junk Dimension)

Contains payment method and sales channel information, enabling analysis of sales channel effectiveness and payment method preferences.

- **OrderTime & IsReturned In Fact Table:**

  (Degenerated Dimension)

4. Fact Table:

Why Only One Fact Table?

We chose to work with only one fact table instead of creating separate cumulative and transactional fact tables. This decision was driven by our focus on enhancing performance. Introducing multiple fact tables would necessitate numerous joins between the largest tables in the Data Warehouse (DWH), potentially compromising performance. While opting for a single fact table may introduce some redundancy and increase storage requirements, our priority lies in optimizing performance rather than minimizing storage.

What is the Grain?

The grain of the order fact table is defined as every type of product per order. Therefore, the primary key (PK) for this fact table is a composite one consisting of OrderID and ProductID.

What are the Foreign Keys (FKs)?

Foreign keys (FKs) are used to add context to the measurements in the analysis. The FKs in the FactOrder table include ProductKey, CustomerKey, EmployeeKey, PromotionKey, ShipperKey, OrderDateKey, RequiredDateKey, ShippedDateKey, ReturnDate, and PayChanKey.

What are the Measurements?

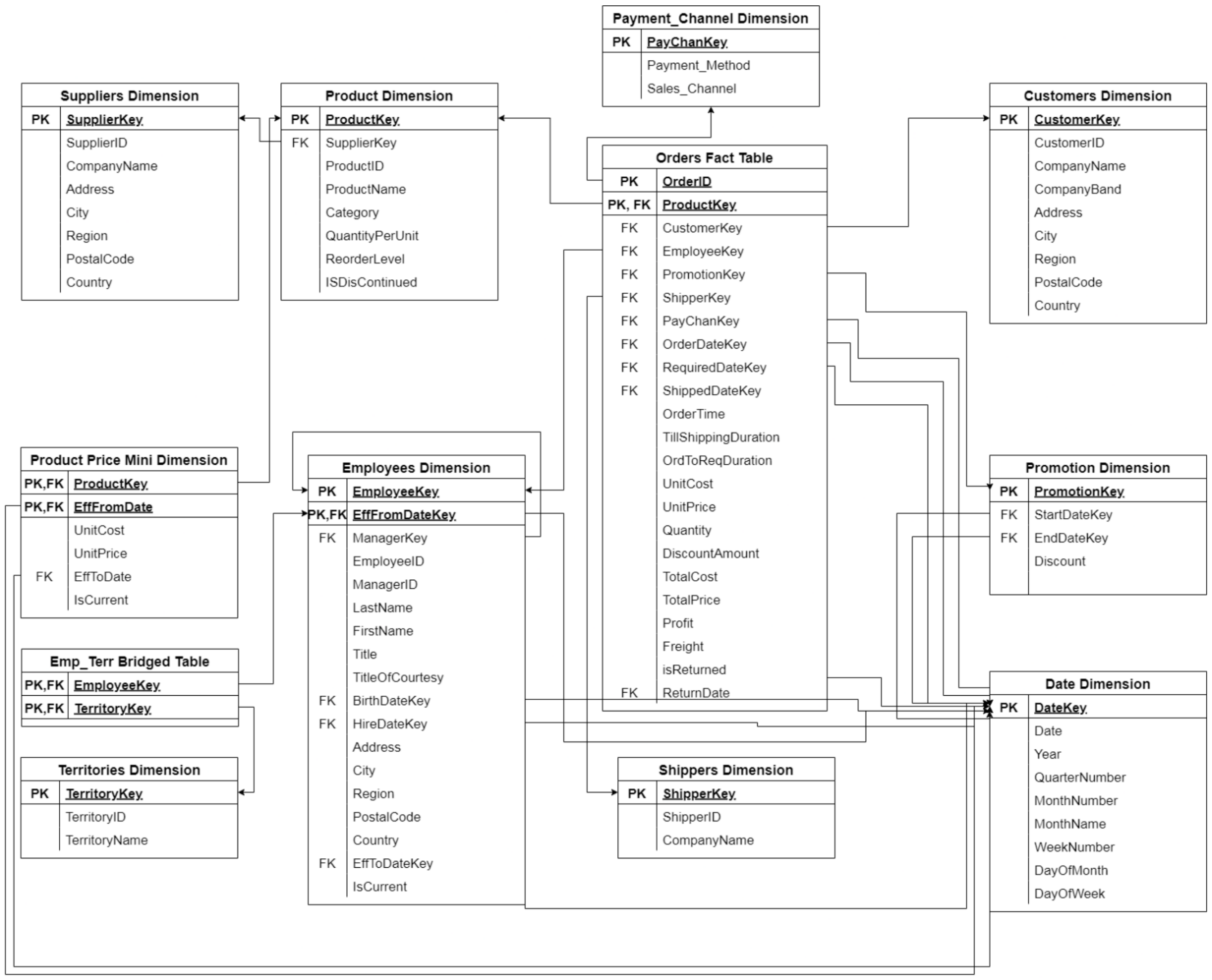
- **TillShippingDuration:**

    - Duration in days between the order placement and shipment.
- **OrdToReqDuration:**
    - Duration in days between the order placement and the required date.
- **UnitCost:**
    - The cost per unit of the product.
- **UnitPrice:**
    - The selling price per unit of the product.
- **Quantity:**
    - The quantity of the product in the order.
- **DiscountAmount:**
    - The amount of discount applied to the product of order.
- **TotalCost:**
    - The total cost of the whole product quantity in the order.
- **TotalPrice:**
    - The total price of the whole product quantity in the order.
- **Profit:**
    - The profit generated from the product within the specific order.
- **Freight:**
    - (allocated) as it was on the order level and we lowered it to the product level of granularity.
    - The shipping cost for the product within the specific order.
- **isReturned:**
    - Flag indicating whether the order was returned (1) or not (0).

- It was better to put it directly in the fact table as it will save storage as no need to add two additional columns of surrogate key as PK and FK, and this column would mostly be used in conjunction with every query, eliminating the need for a join and enhancing performance.

Data Warehouse Modeling

In our Data Warehouse Modeling, we adopted the snowflake schema to establish relationships between the dimension tables.


Payment_Channel Dimension
PK PayChanKey
Payment_Method
Sales_Channel
Suppliers Dimension
PK SupplierKey
SupplierID
CompanyName
Address
City
Region
PostalCode
Country
Product Dimension
PK ProductKey
FK SupplierKey
ProductID
ProductName
Category
QuantityPerUnit
ReorderLevel
ISDisContinued
Customers Dimension
PK CustomerKey
CustomerID
CompanyName
CompanyBand
Address
City
Region
PostalCode
Country
Orders Fact Table
PK OrderID
PK, FK ProductKey
FK CustomerKey
FK EmployeeKey
FK PromotionKey
FK ShipperKey
FK PayChanKey
FK OrderDateKey
FK RequiredDateKey
FK ShippedDateKey
OrderTime
TillShippingDuration
OrdToReqDuration
UnitCost
UnitPrice
Quantity
DiscountAmount
TotalCost
TotalPrice
Profit
Freight
isReturned
FK ReturnDate
Product Price Mini Dimension
PK,FK ProductKey
PK,FK EffFromDate
UnitCost
UnitPrice
FK EffToDate
IsCurrent
Employees Dimension
PK EmployeeKey
PK,FK EffFromDateKey
FK ManagerKey
EmployeeID
ManagerID
LastName
FirstName
Title
TitleOfCourtesy
FK BirthDateKey
FK HireDateKey
Address
City
Region
PostalCode
Country
FK EffToDateKey
IsCurrent
Promotion Dimension
PK PromotionKey
FK StartDateKey
FK EndDateKey
Discount
Emp_Terr Bridged Table
PK,FK EmployeeKey
PK,FK TerritoryKey
Territories Dimension
PK TerritoryKey
TerritoryID
TerritoryName
Shippers Dimension
PK ShipperKey
ShipperID
CompanyName
Date Dimension
PK DateKey
Date
Year
QuarterNumber
MonthNumber
MonthName
WeekNumber
DayOfMonth
DayOfWeek

Business Questions

1) Does order fulfillment speed (early vs. late deliveries) influence return rates among returned items?

2) How do Duels (Managers and their Employees (direct reports)) compare in terms of total profit generated and profit per day?

3) In each city we operate in, which product categories are the most profitable?

4) What are the preferred payment methods and sales channels?

5) In each city we operate in, what are the top 3 most frequently returned products?

6) Who are the top 5 customers, based on the total amount spent, in every city where we conduct business?

7)What Is The Impact Of Discounts On Profitability ? (Profit With Vs. Without Discounts)

8)Which suppliers are the most valuable in terms of product variety?

9) Which suppliers are the most valuable in terms of overall profitability?

10) Which products are the most profitable during holidays based on quantity sold, total sales price, and total profit?

11) Which suppliers have the highest number of returned products?

Indexes

1. Bitmap Index on City Column in DimCustomer Table

A bitmap index is chosen for the City column in the DimCustomer table due to its high cardinality nature. Cities often have numerous distinct values, making them suitable candidates for a bitmap index. This index type efficiently handles queries involving city-based filters or aggregations, such as counting customers by city, retrieving customers from specific cities, or analyzing customer distribution across different cities. By using a bitmap index, the database system can optimize query performance for city-related analytics and improve overall response times.

2. Bitmap Index on Category Column in DimProduct Table

Similar to the City column, the Category column in the DimProduct table is also expected to have high cardinality. Products are typically classified into various categories, resulting in a wide range of distinct category values. Utilizing a bitmap index on the Category column enables quick retrieval and analysis of products based on their categories. This index type is particularly beneficial for category-based reporting, analysis, and filtering operations. For example, it facilitates tasks such as analyzing sales performance by category, identifying top-selling categories, or comparing sales trends across different product categories.

3. Hash Index on ProductName Column in DimProduct Table

The ProductName column in the DimProduct table is often queried for exact matches, such as when users search for specific products by name. In such scenarios, a hash index on the ProductName column provides fast lookup performance for queries that require exact matching of product names. Hash indexes excel in handling equality searches, making them well-suited for this type of query pattern. By using a hash index, the database system can efficiently locate and retrieve products based on their names, enhancing user experience and query responsiveness.

4. Default B-Tree Indexes on FactOrder Table Columns

Several columns in the FactOrder table, including CustomerKey, EmployeeKey, PromotionKey, ShipperKey, and others, are indexed using default B-tree indexes. B-tree indexes are versatile and efficient for a wide range of queries, including range queries, equality searches, and sorting operations. These indexes enhance query performance for various order-related analytics, such as analyzing customer orders, evaluating employee performance, assessing promotional effectiveness, and optimizing shipping processes. By utilizing default B-tree indexes on these columns, the database system can handle diverse query requirements effectively and improve overall system performance.

5. Default B-Tree Index on SupplierKey Column in DimProduct Table

The SupplierKey column in the DimProduct table is indexed using a default B-tree index to optimize queries related to product suppliers. This index facilitates fast retrieval of products associated with specific suppliers, supplier performance analysis, supplier-related reporting, and supplier relationship management. By using a default B-tree index on the SupplierKey column, the database system can efficiently handle supplier-related queries and improve overall data retrieval performance.

6. Default B-Tree Indexes on DimEmployee Table Columns

Several columns in the DimEmployee table, including ManagerKey, BirthdateKey, HireDateKey, and EffectiveToDateKey, are indexed using default B-tree indexes. These indexes support efficient employee-related queries, including hierarchical queries, age-based analysis, tenure analysis, historical data retrieval, and workforce planning. By leveraging default B-tree indexes on these columns, the database system can optimize employee-related analytics, improve query performance, and facilitate data-driven decision-making in human resources management.

7. Default B-Tree Indexes on DimProductPriceMini Table Columns

The unitcost and unitprice columns in the DimProductPriceMini table are indexed using default B-tree indexes. These indexes aid in price analysis, cost comparisons, price trend analysis, and pricing strategy optimization for products over time. By using default B-tree indexes on these columns, the database system can efficiently handle price-related queries, support pricing decision-making processes, and improve overall data analysis capabilities.

Performance Comparison:

Query No	Before Indexes	After Indexes
1	Execution Time: 0.724 ms	Execution Time: 0.499 ms
2	Execution Time: 3.540 ms	Execution Time: 2.494 ms
3	Execution Time: 2.237 ms	Execution Time: 2.167 ms
4	Execution Time: 2.916 ms	Execution Time: 1.867 ms
5	Execution Time: 0.885 ms	Execution Time: 0.847 ms
6	Execution Time: 2.525 ms	Execution Time: 1.156 ms
7	Execution Time: 0.440 ms	Execution Time: 0.390 ms
8	Execution Time: 2.443 ms	Execution Time: 1.693 ms
9	Execution Time: 1.190 ms	Execution Time: 0.633 ms
10	Execution Time: 1.078 ms	Execution Time: 0.665 ms

Note:

The details of the queries are in the queries pdf file.