Ate Rose Grocery Store Database Project

Introduction

Background of Ate Rose Grocery Store

Ate Rose Grocery Store stands as a cornerstone of our local community, offering an extensive range of products to meet the daily needs of our valued customers. Over the years, this establishment has earned the trust and loyalty of the community by consistently delivering high-quality products and exceptional customer service. Ate Rose Grocery Store has become an integral part of our neighborhood, serving as not just a place to shop but also a hub for social interactions and community engagement.
In the face of ever-evolving market dynamics and customer expectations, Ate Rose Grocery Store faces several operational challenges. These challenges include optimizing inventory management, enhancing customer engagement, improving employee efficiency, and gaining valuable financial insights. It is evident that an organized and data-driven approach is required to address these challenges effectively and ensure the store's sustained growth and success.

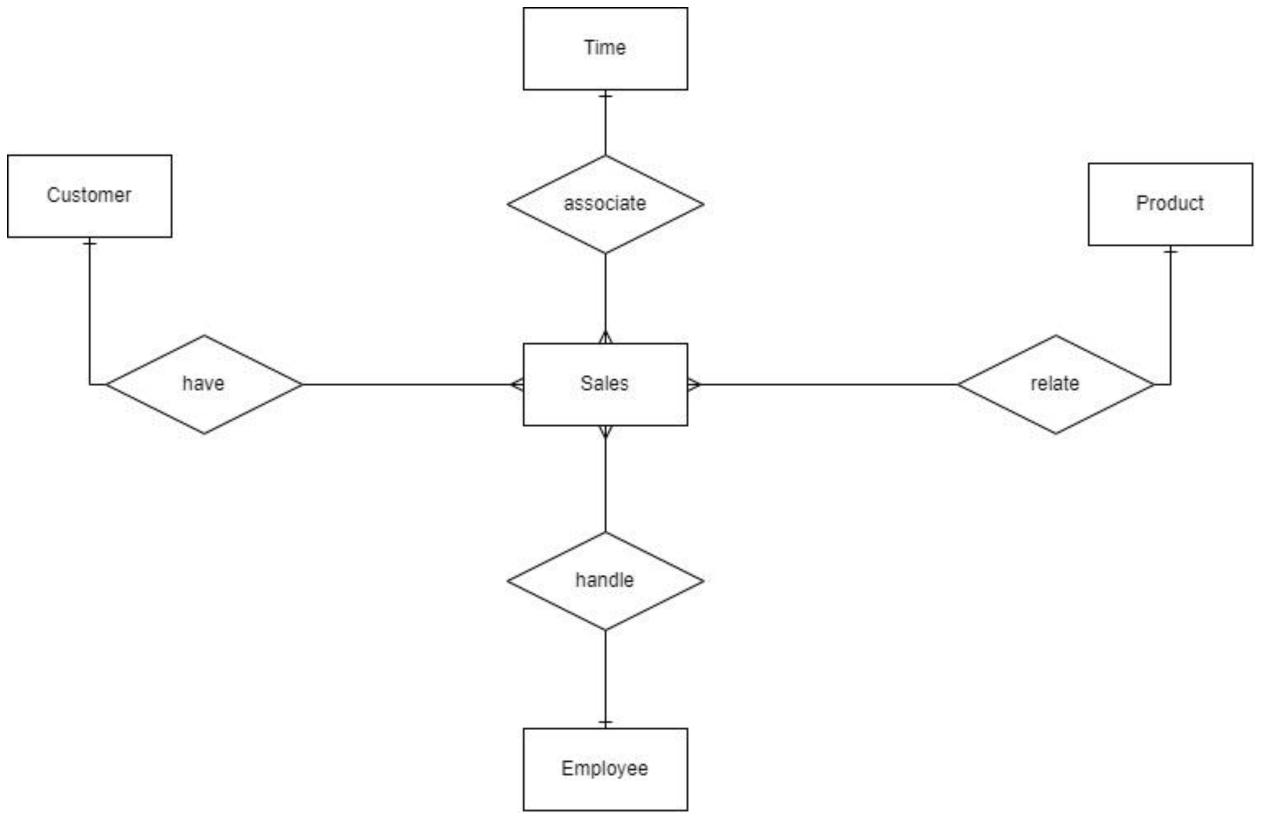
Intended Reports
To achieve these objectives, the Ate Rose Grocery Store Database Project will focus on generating a set of essential reports that the database system is designed to support:
Product Performance Report: This report will offer insights into product sales, category trends, pricing strategies, and stock quantity management. It will enable Ate Rose Grocery Store to identify top-selling products, optimize inventory levels, and streamline purchasing decisions.
Customer Behavior Analysis: By analyzing customer purchase history, preferences, and loyalty patterns, this report will help tailor promotions, provide personalized recommendations, and enhance overall customer satisfaction. A deeper understanding of shopping behavior will enable targeted marketing efforts.
Inventory Optimization Report: Efficiently managing stock levels and identifying slow-moving items are critical for minimizing costs and ensuring product availability. The Inventory Optimization Report will provide data-driven recommendations for inventory management and restocking strategies.
Employee Productivity Report: Assessing employee performance, monitoring work schedules, and quantifying sales contributions will empower Ate Rose Grocery Store to optimize staffing levels, motivate the team, and improve customer service.
Financial Performance Analysis: Timely financial reporting is crucial for evaluating profitability, identifying cost drivers, and making informed decisions. The Financial Performance Analysis report will offer insights into monthly revenue trends, profit margins, and expenditure breakdowns.

Business Questions:

1. How many products were sold during a specific time period?
2. What is the total revenue generated by each customer?
3. Which employees had the highest sales in a given month?
4. What are the most popular product categories among customers?
5. How does the stock quantity of products affect sales?

By implementing the database system and generating these reports, the Ate Rose Grocery Store aims to maintain its commitment to providing quality products and services to our community. This project signifies our dedication to addressing operational challenges proactively while fostering continued growth and success.

ERD - Entity Relationship Diagram



Normalized Tables and Relationships

1. Products Table:

- **ProductID (Primary Key):** A unique identifier for each product.
- **Name:** The name of the product.
- **Category:** The category to which the product belongs (e.g., dairy, produce).
- **Price:** The price of the product.
- **StockQuantity:** The quantity of the product in stock.

2. Customers Table:

- **CustomerID (Primary Key):** A unique identifier for each customer.
- **Name:** The name of the customer.
- **ContactNumber:** The contact number of the customer.
- **Email:** The email address of the customer.

3. Sales Table:

- **SaleID (Primary Key):** A unique identifier for each sale.
- **SaleDate:** The date of the sale.
- **TotalAmount:** The total amount of the sale.
- **CustomerID (Foreign Key):** A reference to the customer who made the purchase.

4. SalesItems Table:

- **SaleItemID (Primary Key):** A unique identifier for each sale item.
- **Quantity:** The quantity of the product sold in the sale.
- **Amount:** The total amount for the sale item.
- **SaleID (Foreign Key):** A reference to the sale to which this item belongs.
- **ProductID (Foreign Key):** A reference to the product sold.

5. Employees Table:

- **EmployeeID (Primary Key):** A unique identifier for each employee.
- **Name:** The name of the employee.
- **ContactNumber:** The contact number of the employee.
- **Position:** The position or role of the employee.
- **WorkSchedule:** The work schedule of the employee.

Relationships:

- Each sale in the Sales table is associated with a customer through the CustomerID foreign key. This represents a one-to-many relationship as one customer can make multiple purchases, but each sale belongs to one customer.

- The SalesItems table is linked to the Sales table through the SaleID foreign key, representing a one-to-many relationship. Each sale can have multiple items, but each item belongs to one sale.

- Each item in the SalesItems table is associated with a specific product through the ProductID foreign key. This establishes a one-to-one relationship between products and sales items.

- Employee data may be linked to sales if necessary for tracking which employee processed a particular sale. This can be represented as a foreign key in the Sales table, linking to the Employee table.

These normalized tables and relationships ensure data integrity, minimize redundancy, and support efficient querying for the Ate Rose Grocery Store database. They serve as the foundation for organizing and managing store data effectively.

Dimensional Normal Form (DNF) Methodology

The Dimensional Normal Form (DNF) methodology is a critical aspect of dimensional modeling in data warehousing. It focuses on structuring dimension tables efficiently to support analytical reporting and querying while minimizing data redundancy. In the context of the "Ate Rose Grocery Store" database project, we will apply DNF principles to organize dimension tables and fact tables effectively.

Dimension Families

Dimension families are groups of related dimensions that share common attributes or are closely related in terms of business semantics. For the "Ate Rose Grocery Store" database project, we will define the following dimension families:

1. Customer Dimensions:
- This dimension family includes dimensions related to customer information, such as customer demographics and preferences. It provides insights into customer behavior and shopping patterns.

2. Product Dimensions:
- Product dimensions encompass attributes related to the products sold in the store. This dimension family allows for detailed analysis of product performance and categorization.

3. Time Dimensions:
- Time dimensions provide a temporal context for analyzing data. They enable time-based analysis, such as sales trends over days, weeks, or months.

4. Employee Dimensions:
- Employee dimensions capture information about store employees, including their roles and schedules. This dimension family is valuable for assessing employee productivity and performance.

Fact Tables

Fact tables contain quantitative data (facts) and are associated with dimension tables through foreign keys. Fact tables are used to store measures and support the analysis of business events. In the "Ate Rose Grocery Store" database project, we will define the following fact table:

Sales Fact Table:
- The Sales Fact Table records key sales-related metrics, including total sales amounts and quantities. It is linked to dimension tables such as Customer Dimensions, Product Dimensions, and Time Dimensions, enabling comprehensive sales analysis.

By implementing dimension families and fact tables according to the DNF methodology, we can create a well-structured data warehouse for Ate Rose Grocery Store. This approach ensures that data is organized efficiently, supports complex queries, and facilitates meaningful insights into store operations and customer behavior.

Dimensional Model

Dimension Tables

1. Customer Dimensions:

DimCustomers:

- **CustomerKey (Surrogate Key):** A unique identifier for each customer.
- **CustomerID (Natural Key):** The original customer identifier.
- **Name:** The name of the customer.
- **ContactNumber:** The contact number of the customer.
- **Email:** The email address of the customer.

2. Product Dimensions:

DimProducts:

- **ProductKey (Surrogate Key):** A unique identifier for each product.
- **ProductID (Natural Key):** The original product identifier.
- **Name:** The name of the product.
- **Category:** The category to which the product belongs (e.g., dairy, produce).
- **Price:** The price of the product.
- **StockQuantity:** The quantity of the product in stock.

3. Time Dimensions:

DimTime:

- **TimeKey (Surrogate Key):** A unique identifier for each time period.
- **Date:** The date for the time period.
- **Day:** The day of the week.
- **Month:** The month.
- **Quarter:** The quarter of the year.
- **Year:** The year.

4. Employee Dimensions:

DimEmployees:

- **EmployeeKey (Surrogate Key): A** unique identifier for each employee.
- **EmployeeID (Natural Key):** The original employee identifier.
- **Name:** The name of the employee.
- **ContactNumber:** The contact number of the employee.
- **Position:** The position or role of the employee.
- **WorkSchedule:** The work schedule of the employee.

Fact Table

Sales Fact Table:

FactSales:

- **SalesKey (Surrogate Key):** A unique identifier for each sales transaction.
- **DateKey (Foreign Key):** A reference to the Date dimension (DimTime).
- **CustomerKey (Foreign Key):** A reference to the Customer dimension (DimCustomers).
- **ProductKey (Foreign Key):** A reference to the Product dimension (DimProducts).
- **EmployeeKey (Foreign Key):** A reference to the Employee dimension (DimEmployees).
- **QuantitySold:** The quantity of products sold in the transaction.
- **TotalAmount:** The total amount of the sale.

This dimensional model for the "Ate Rose Grocery Store" database project organizes data into dimension tables and a central fact table. It allows for comprehensive analysis of sales data with the context provided by customer, product, time, and employee dimensions. The surrogate keys ensure data consistency and support historical tracking when necessary.

SQL scripts for Creating and Inserting Data

SQL Script for Creating and Inserting in the Database (DB)

```
CREATE DATABASE "AteRoseGroceryStoreDB";

-- Create Customers Table
CREATE TABLE IF NOT EXISTS Customers (
   CustomerKey SERIAL PRIMARY KEY,
   CustomerID VARCHAR(10) NOT NULL,
   Name VARCHAR(255) NOT NULL,
   ContactNumber VARCHAR(15),
   Email VARCHAR(255)
);

-- Create Products Table
CREATE TABLE IF NOT EXISTS Products (
   ProductKey SERIAL PRIMARY KEY,
   ProductID VARCHAR(10) NOT NULL,
   Name VARCHAR(255) NOT NULL,
   Category VARCHAR(50),
   Price NUMERIC(10, 2) NOT NULL,
   StockQuantity INT NOT NULL
);

-- Create Time Table (DimTime)
CREATE TABLE IF NOT EXISTS DimTime (
   TimeKey SERIAL PRIMARY KEY,
   Date DATE NOT NULL,
   Day VARCHAR(10),
   Month VARCHAR(10),
   Quarter VARCHAR(10),
   Year INT
);

-- Create Employees Table
CREATE TABLE IF NOT EXISTS Employees (
   EmployeeKey SERIAL PRIMARY KEY,
   EmployeeID VARCHAR(10) NOT NULL,
   Name VARCHAR(255) NOT NULL,
   ContactNumber VARCHAR(15),
   Position VARCHAR(50),
   WorkSchedule VARCHAR(255)
);
```

```
-- Create Sales Fact Table
CREATE TABLE IF NOT EXISTS FactSales (
    SalesKey SERIAL PRIMARY KEY,
    DateKey INT,
    CustomerKey INT,
    ProductKey INT,
    EmployeeKey INT,
    QuantitySold INT,
    TotalAmount NUMERIC(10, 2)

);

-- Insert Dummy Data into Customers Table
INSERT INTO Customers (CustomerID, Name, ContactNumber, Email)
VALUES
    ('C001', 'John Smith', '555-1234', 'john@gmail.com'),
    ('C002', 'Alice Johnson', '555-5678', 'alice@gmail.com');


-- Insert Dummy Data into Products Table
INSERT INTO Products (ProductID, Name, Category, Price, StockQuantity)
VALUES
    ('P001', 'Milk', 'Dairy', 2.99, 100),
    ('P002', 'Apples', 'Produce', 1.49, 200);


-- Insert Dummy Data into DimTime Table (limited data for demonstration)
INSERT INTO DimTime (Date, Day, Month, Quarter, Year)
VALUES
    ('2023-01-01', 'Sunday', 'January', 'Q1', 2023),
    ('2023-01-02', 'Monday', 'January', 'Q1', 2023);


-- Insert Dummy Data into Employees Table
INSERT INTO Employees (EmployeeID, Name, ContactNumber, Position, WorkSchedule)
VALUES
    ('E001', 'Emily Davis', '555-9876', 'Cashier', '9 AM - 5 PM'),
    ('E002', 'Michael Wilson', '555-5432', 'Stock Clerk', '1 PM - 9 PM');


-- Insert Data into the Fact Table

-- Insert Dummy Data into FactSales Table
INSERT INTO FactSales (DateKey, CustomerKey, ProductKey, EmployeeKey, QuantitySold, TotalAmount)
VALUES
    (1, 1, 1, 1, 5, 14.95),
    (2, 2, 2, 2, 10, 14.90);
```

SQL Script for Creating and Inserting in the Datawarehouse(DW)

```
CREATE DATABASE "AteRoseGroceryStoreDW";

CREATE TABLE IF NOT EXISTS DimCustomers (
    CustomerKey SERIAL PRIMARY KEY,
    CustomerID VARCHAR(10) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    ContactNumber VARCHAR(15),
    Email VARCHAR(255)
);

-- Create DimProducts Table
CREATE TABLE IF NOT EXISTS DimProducts (
    ProductKey SERIAL PRIMARY KEY,
    ProductID VARCHAR(10) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Category VARCHAR(50),
    Price NUMERIC(10, 2) NOT NULL,
    StockQuantity INT NOT NULL
);

-- Create DimTime Table
CREATE TABLE IF NOT EXISTS DimTime (
    TimeKey SERIAL PRIMARY KEY,
    Date DATE NOT NULL,
    Day VARCHAR(10),
    Month VARCHAR(10),
    Quarter VARCHAR(10),
    Year INT
);

-- Create DimEmployees Table
CREATE TABLE IF NOT EXISTS DimEmployees (
    EmployeeKey SERIAL PRIMARY KEY,
    EmployeeID VARCHAR(10) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    ContactNumber VARCHAR(15),
    Position VARCHAR(50),
    WorkSchedule VARCHAR(255)
);

-- Create Fact Table

-- Create FactSales Table
CREATE TABLE IF NOT EXISTS FactSales (
    SalesKey SERIAL PRIMARY KEY,
    DateKey INT,
    CustomerKey INT,
    ProductKey INT,
    EmployeeKey INT,
    QuantitySold INT,
    TotalAmount NUMERIC(10, 2)
);


-- Insert Dummy Data into DimCustomers Table
INSERT INTO DimCustomers (CustomerID, Name, ContactNumber, Email)
VALUES
    ('C001', 'John Smith', '555-1234', 'john@gmail.com'),
    ('C002', 'Alice Johnson', '555-5678', 'alice@gmail.com');


-- Insert Dummy Data into DimProducts Table
INSERT INTO DimProducts (ProductID, Name, Category, Price, StockQuantity)
VALUES
    ('P001', 'Milk', 'Dairy', 2.99, 100),
    ('P002', 'Apples', 'Produce', 1.49, 200);


-- Insert Dummy Data into DimTime Table (limited data for demonstration)
INSERT INTO DimTime (Date, Day, Month, Quarter, Year)
VALUES
    ('2023-01-01', 'Sunday', 'January', 'Q1', 2023),
    ('2023-01-02', 'Monday', 'January', 'Q1', 2023);


-- Insert Dummy Data into DimEmployees Table
INSERT INTO DimEmployees (EmployeeID, Name, ContactNumber, Position, WorkSchedule)
VALUES
    ('E001', 'Emily Davis', '555-9876', 'Cashier', '9 AM - 5 PM'),
    ('E002', 'Michael Wilson', '555-5432', 'Stock Clerk', '1 PM - 9 PM');

-- Insert Data into the Fact Table

-- Insert Dummy Data into FactSales Table
INSERT INTO FactSales (DateKey, CustomerKey, ProductKey, EmployeeKey, QuantitySold, TotalAmount)
VALUES
    (1, 1, 1, 1, 5, 14.95),
    (2, 2, 2, 2, 10, 14.90);
```