Table of Contents

Data Types

Store

Attribute	Data Type	Nullable
StoreNumber	String	Not Null
PhoneNumber	String	Not Null
StreetAddress	String	Not Null
ZipCode	String	Not Null

City

Attribute	Data Type	Nullable
CityName	String	Not Null
State	String	Not Null
Population	Integer	Not Null
CitySizeCategory	String	Not Null

Membership

Attribute	Data Type	Nullable
MemberID	String	Not Null
SignupDate	Date	Not Null

Membership Type

Attribute	Data Type	Nullable
TypeName	String	Not Null

Product

Attribute	Data Type	Nullable
ProductID	String	Not Null
ProductName	String	Not Null
RetailPrice	Float	Not Null

Manufacturer

Attribute	Data Type	Nullable
ManufacturerID	String	Not Null
ManufacturerName	String	Not Null
MaximumDiscount	Float	Null

Table of Contents

Category

Attribute	Data Type	Nullable
CategoryName	String	Not Null

OnSale

Attribute	Data Type	Nullable
OnSaleDate	Date	Not Null
OnSalePrice	Float	Not Null

Calendar

Attribute	Data Type	Nullable
Date	Date	Not Null

Holiday

Attribute	Data Type	Nullable
HolidayName	String	Not Null

Table of Contents

Business Logic Constraints

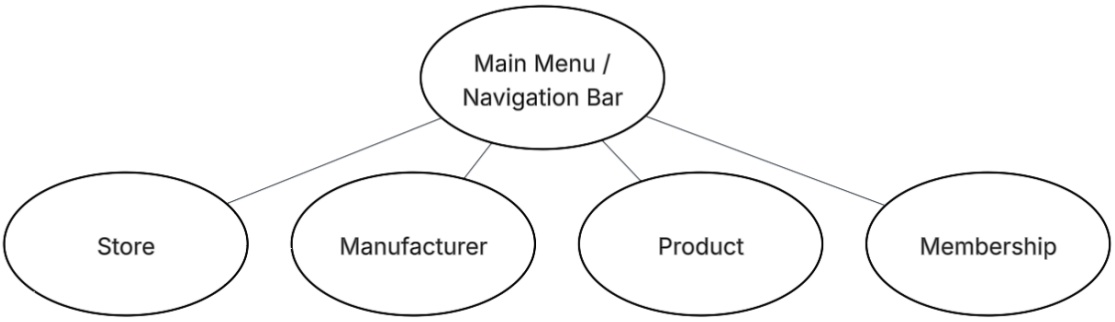
- All products are available and sold at all stores, that is, there is no need to specify that a certain product is only available at a certain store.
- The retail price is in effect unless there is a sale.
- If a product goes on sale on a certain day, it is on sale at the same price on this day in all stores, that is saying stores are not allowed to hold sales independently or have store-specific sale prices.
- The PSCDW disallows sale prices that are higher than retail prices.
- Some manufacturers put a cap on the maximum discount that any retailer can apply to any of the manufacturer's products in terms of a percentage.
- Even if a manufacturer does not specify a maximum discount, Peachtree Savings Club's general rule is that no product may be discounted more than 90% off the retail price.
- Savings Club administration does not allow any product to be on sale for more than 25 days in any calendar month.
- That as with any retail store, pricing errors may occur for a variety of reasons, and such errors in historical data should not be corrected.
- We assume that our company does not need to deal with any sales tax.
- The data warehouse is not required to store which products were purchased together during a single sales transaction.

Table of Contents

Task Decomposition with Abstract Code

Main Menu / Navigation Bar

Task Decomposition



Lock Types: 4 Read-only lookups on Store, Manufacturer, Product, Membership.
Number of Locks: 4
Enabling Conditions: None.
Frequency: Around 50 visits per day and all subtasks have the same frequency.
Consistency (ACID): is not critical.
Subtasks: Mother Task is needed since all subtasks must be done but order is not critical.

Abstract Code

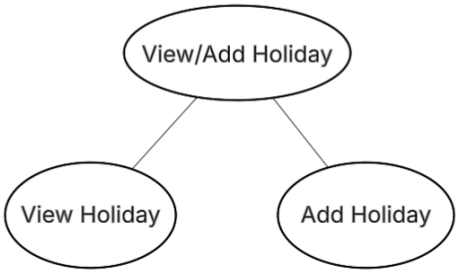
- When user successfully land on or navigate to **Main Menu** form:
  - Find Store and display the count of stores.
  - Find Manufacturer and display the count of manufacturers.
  - Find Product and display the count of products sold.
  - Find Membership and display the count of memberships sold.
- Show "***View/Add Holiday***", "***Update City Population***", "***Manufacturer's Product Report***", "***Category Report***", "***Actual vs. Predicted Revenue for Speaker Units***", "***Store Revenue by Year by State***", "***Air Conditioners on Groundhog Day***", "***State with Highest Volume for each Category***", "***Revenue by Population***", and **"*Membership Trends*"** buttons.
- Upon:
  - Click ***View/Add Holiday*** button – Jump to the **View/Add Holiday** task.
  - Click ***Update City Population*** button – Jump to the **Update City Population** task.
  - Click ***Manufacturer's Product Report*** button – Jump to the **Top 100 Manufacturers and Drill-down** task.
  - Click ***Category Report*** button – Jump to the **View Category Details** task.

Table of Contents

- Click ***Actual vs. Predicted Revenue for Speaker Units*** button – Jump to the **Revenue Gap Analysis** task.
- Click ***Store Revenue by Year by State*** button – Jump to the **Yearly Store Revenue Filter by State** task.
- Click ***Air Conditioners on Groundhog Day*** button – Jump to the **View Air Conditioners Sales on Groundhog Day** task.
- Click ***State with Highest Volume for each Category*** button – Jump to the **Top State for each Category Filter by Year-Month** task.
- Click ***Revenue by Population*** button – Jump to the **View Revenue by Population** task.
- Click ***Membership Trends*** button – Jump to the **View Memberships Sold per Year** task.

View/Add Holiday

Task Decomposition



Lock Types: 2 Read-only lookups on Calendar and Holiday and 1 Write lock for insert data into Holiday.
Number of Locks: 3
Enabling Conditions: None.
Frequency: Around 50 visits per day for **View Holiday** task and around 5 times per month for **Add Holiday** task. They have different frequency, usually **Add Holiday** is less than **View Holiday**.
Consistency (ACID): is not critical. Even if the Holiday is edited by some user, you can still get holiday information up-to-date.
Subtasks: Mother task is needed to coordinate these two subtasks. The order is critical as lookups need be done first followed by any number of edits and lookups.

Abstract Code

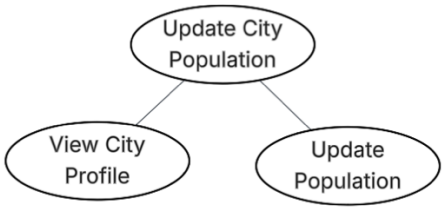
- User click ***View Holiday*** button from **Main Menu** form or successfully submit add holiday requests.
- Run the **View Holiday** task: Find each Holiday using Calendar.Date; Display Calendar.Date and Holiday.HolidayName.

Table of Contents

- If user click ***Add Holiday*** button from **Holidays Information List** form, then:
    - User enter *date* ('$date') and *holiday* ('$holiday') input fields.
        - If data validation is successful for *date* and *holiday* input fields, then:
            - When ***Save*** button is clicked:
                - If Holiday record is found:
                    - Go back to **Add Holiday** form, with error message.
                - Else:
                    - Insert Holiday record.
                    - Go to **Holidays Information List** form.
        - Else *date* and *holiday* input fields are invalid, display **Add Holiday** form, with error message.
- Else no action is taken on **Holiday Information List** form then do nothing.
- When ready, user can select next action from Navigation Bar.

Update City Population

Task Decomposition



Lock Types: Read and Write locks to update population of City.
Number of Locks: 2
Enabling Conditions: None.
Frequency: Around 20 times per day, also the frequency is depending on the number of cities need an update.
Consistency (ACID): is not critical. Even if the city is edited by some user, you can still get city information up-to-date.
Subtasks: Mother task is needed to coordinate these two subtasks. The order is critical as lookups need be done first followed by any number of edits and lookups.

Abstract Code

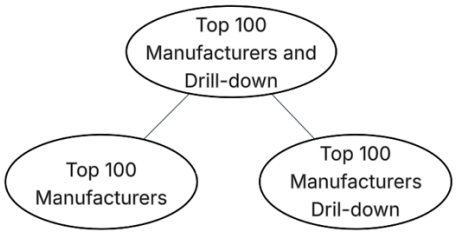
- User click ***Update City Population*** button from **Main Menu** form.
- Run **View City Profile** task: Find each City; Display CityName, State, and Population.
- If user click ***Edit*** button on **City Information List** form, then all city population fields become editable.
    - If user edit *population* ('$population') input field, then:
        - If data validation is successful for *population* input field, then:

- When ***Save*** button is clicked:
    - If City record is found and City.Population != '$population':
        - Update City record.
        - Display **City Information List** form.
    - Else:
        - Display **City Information List** form, with error message.
- Else *population* input field is invalid, display **City Information List** form, with error message.
- Else user click ***Cancel*** button, then go back to **City Information List** form.

- Else no action is taken on **City Information List** form then do nothing.
- When ready, user can select next action from Navigation Bar.

Top 100 Manufacturers and Drill-down

Task Decomposition



Lock Types: 3 Read-only lookups on Manufacturer, Product, and Category.
Number of Locks: 3
Enabling Conditions: None.
Frequency: Around 50 visits per day on main report and around 100 times per day on drill-down report. They have different frequency and usually drill-down is more than main report.
Consistency (ACID): is not critical. User click drill-down doesn't affect any other subtasks since it's read-only lookups.
Subtasks: Mother task is needed to coordinate subtasks. Order is critical, we probably more often have general query comes first and then drill down.

Abstract Code

- User click ***Manufacturer's Product Report*** button from **Main Menu** form.
- Run the **Top 100 Manufacturers** task:
    - Find all Manufactuer and retrieve ManufacturerID and ManufacturerName.
    - For each Manufacturer, find all Product produced by that Manufacturer and retrieve ManufacturerID, ManufacturerName, ProductID and RetailPrice.
        - Count total number of products offered by the manufacturer using ProductID.

Table of Contents

        - Calculate Average Retail Price, Minimum Retail Price, Maximum Retail Price by aggregating ManufacturerID and ManufacturerName.
        - Calculate Retail Price Range as the difference of maximum retail price and minimum retail price.
    - Dispaly ManufacturerID, ManufacturerName, Total number of products offered by the manfuacturer, Average Retail Price, Minimum Retail Price, Maximum Retail Price, and Retail Price Range.
    - Sort by average retail price in descending order.
    - Limit results to top 100 manufacturers based on average price.
- When ready, user can click drill-down or select next action from Navigation Bar.

- If user click ***drill-down*** button on a ManufacturerName, then run **Top 100 Manufacturers Drill-down** task:
    - Query for information about selected Manufacturer where $manufactuerid is stored in the web session.
    - Find the current Manufacturer using ManufactuerID;
    - For each Product offered by the current Manufacturer:
        - Find all Category associated with that product and concatenate all CategoryName in alphabetical order, separated by forward slashes.
    - Display ManufacturerID, ManufacturerName, and MaximumDiscount in the report header; And for each of manufacturer's products, display ProductID, ProductName, CategoryName(s) and Retail Price.
    - Sort Products first by RetailPrice descending then by ProductID ascending.
- Else no action is taken on **Manufacturer's Product Report** form then do nothing.
- When ready, user can select next action from Navigation Bar.

View Category Details

Task Decomposition

Lock Types: 5 Read-only lookups on Category, Product, Manufacturer, OnSale, and Calendar.
Number of Locks: 5
Enabling Conditions: None.
Frequency: Around 50 times per day.
Consistency (ACID): is not critical.
Subtasks: Mother task is not needed. No task decomposition.

Abstract Code

- User click ***Category Report*** button from **Main Menu** form.
- Run the **View Category Details** task:

- For each Category, find all Product associated with that Category and retrieve CategoryName, ProductID, and Retail Price.
  - Count total number of products in that category using ProductID.
  - Calculate average retail price for all products in that category.
  - For each Product, find all Manufacturer offered that Product and retrieve CategoryName, ProductID, and ManufacturerID.
    - Count total number of unique manufacturers offering products in that category using ManufacturerID.
  - For each Product, find all sales for the product and identify actual selling price:
    - If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.
    - Retrieve CategoryName, ProductID, Actual Selling Price, and QuantitySold.
    - Calculate total revenue for all products sold in that category using QuantitySold multiply by Actual Selling Price.
- Display CategoryName, total number of products in that category, total number of unique manufacturers offering products in that category, average retail price, and total revenue for all products sold in that category.
- Sort by CategoryName ascending.

- When ready, user can select next action from Navigation Bar.

Revenue Gap Analysis

Task Decomposition

Lock Types: Read-only lookups on Product, Category, OnSale, and Calendar
Number of Locks: 4
Enabling Conditions: None.
Frequency: Around 50 times per day.
Consistency (ACID): is not critical.
Subtasks: Mother task is not needed. No task decomposition.

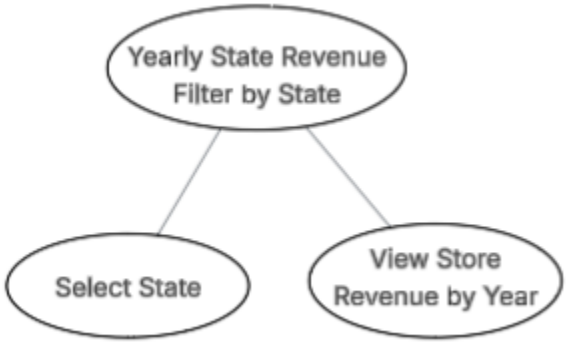
Abstract Code

- User click ***Actual vs. Predicted Revenue for Speaker Units*** button form **Main Menu** form.
- Run the **Revenue Gap Analysis** task:
  - Find all products in Speaker category using Category.CategoryName, and retrieve ProductID, ProductName, Retail Price.
  - For each Product in Speaker category:

    - Add all QuantitySold for this product as total number of units sold.
    - Find all sales for the product.
        - Calculate total number of units sold at discount by adding all QuantitySold if SaleDate (i.e. Calendar.Date) is in the OnSaleDate of OnSale.
        - Calculate total number of units sold at retail price by adding all QuantitySold if Sale Date (i.e. Calendar.Date) is not the OnSaleDate of OnSale.
        - Calculate actual revenue by adding QuantitySold * Actual Selling Price for all sales (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.)
        - Calculate the predicted revenue by adding Retail Price * Predicted QuantitySold for all sales (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the predicted quantity sold is 75% QuantitySold else the predicted quantity sold is QuantitySold.)
        - Compute the difference between actual and predicted revenue.
    - Display ProductID, ProductName, Retail Price, total number of units sold, total number of units sold at discount, total number of units sold at retail, actual revenue collected from all sales of the product, predicted revenue had the products never been put on sale and difference between actual revenue and predicted revenue.
    - Limit only revenue difference greater than $5000 (positive or negative).
    - Sort by revenue difference in descending order.
- When ready, user can select next action from Navigation Bar.

Yearly Store Revenue Filter by State

Task Decomposition



Lock Types: Read-only lookups for City, Store, Calendar, Product, and OnSale.
Number of Locks: 5
Enabling Conditions: None.
Frequency: Around 50 times per day, all subtasks have the same frequency.
Consistency (ACID): is not critical.

Table of Contents

Subtasks: Mother task is needed to coordinate subtasks. Order is critical because user must first select a state.

Abstract Code

- User click ***Store Revenue by Year by State*** button form **Main Menu** form.
- If user select a *state* ('$state') from a drop-down box, then run the **View Store Revenue by Year** task:
  - Query for information about the selected state where $state is stored in the web session.
  - Find all City in the selected state using City.State = '$state' and retrieve CityName.
  - Find all Store located in the City and retrieve Store StreetAddress, ZipCode.
  - For each Store and each sales year:
    - Calculate total revenue by adding QuantitySold * Actual Selling Price for all sales (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.)
    - Calculate revenue for products sold at retail price by adding QuantitySold * Retail Price for all sales if Sale Date (i.e. Calendar.Date) is not the OnSaleDate of OnSale.
    - Calculate revenue for products sold at discounted price by adding QuantitySold * OnSalePrice for all sales if Sale Date (i.e. Calendar.Date) is the OnSaleDate of OnSale.
  - Display StoreNumber, StreetAddress, ZipCode, CityName, sales year, total revenue, revenue for products sold at retail price, and revenue for products sold at discounted price.
  - Sort the report first by year in descending order and then by revenue in descending order.
- When ready, user can select next action from Navigation Bar.

View Air Conditioners Sales on Groundhog Day

Task Decomposition

Lock Types: Read-only lookups for Category, Product, and Calendar.
Number of Locks: 3
Enabling Conditions: None.
Frequency: Around 50 times per day.
Consistency (ACID): is not critical.
Subtasks: Mother task is not needed. No task decomposition.

Table of Contents

Abstract Code

- User click ***Air Conditioners on Groundhog Day*** button from **Main Menu** form.
- Run **View Air Conditioners Sales on Groundhog Day** task:
    - Find all sales and retrieve Calendar.Date, Product.ProductID, and QuantitySold.
    - For each sales year:
        - Adding all QuantitySold for Product whose Category.CategoryName contains "Air Conditioner" as total number of items sold that year in air conditioner category.
        - Calculate the average number of units sold per day as total number of items sold that year in air conditioner category divided by 365 days.
        - Adding all QuantitySold for Product sold on Groundhog Day (February 2nd) of that year as total number of units sold on Groundhog Day.
    - Display year, total number of items sold that year in the air conditioner category, the average number of units sold per day, and the total number of units sold on Groundhog Day of that year.
    - Sort by year in ascending order.
- When ready, user can select next action from Navigation Bar.

Top State for each Category Filter by Year-Month

Task Decomposition

Lock Types: Read-only lookups on Calendar, Product, Category, Store, and City.
Number of Locks: 5
Enabling Conditions: None.
Frequency: Around 50 times per day.
Consistency (ACID): is not critical.
Subtasks: Mother task is needed to coordinate subtasks. Order is critical as user must first select a year and month.

Abstract Code

Table of Contents

- User click ***State with Highest Volume for each Category*** button from **Main Menu** form.
- If user select *Year* ('$year') and *Month* ('$month') from drop-down box, then run **Find top state in each Category** task:
    - Find all sales in selected year and month using Calendar.Date.
        - Find all Category for each Product sold.
        - Find all City for each sold Store.
        - Retrieve Calendar.Date, StoreNumber, ProductID, CategoryName, and QuantitySold.
    - For each Category:
        - Calculate total number of units sold for all stores in each State by adding QuantitySold for all stores in each state.
        - Find the State that sold the highest number of units in that category.
    - Display CategoryName, the state that sold the highest number of units in that category, and the total number of units that were sold in all stores in thta state.
    - Sort by CategoryName in ascending order.
- When ready, user can select next action from Navigation Bar.

View Revenue by Population

Task Decomposition

Lock Types: Read-only lookups on Calendar, Product, Store, City, and OnSale.
Number of Locks: 5
Enabling Conditions: None.
Frequency: Around 50 times per day.
Consistency (ACID): is not critical.
Subtasks: Mother task is not needed. No task decomposition.

Abstract Code

- User click ***Revenue per Population*** button from **Main Menu** form.
- Run **View Revenue per Population** task:
    - Find all sales for every Store.
        - Find all Product and Store related to that sale.
        - Find all City for that Store located in.
            - Derived CitySizeCategory based on City.Population.
        - Find all OnSale for the Product sold in that sale.

        - Retrieve Calendar.Date, ProductID, StoreNumber, CityName, State, CitySizeCategory, Retail Price, OnSalePrice, and QuantitySold.
    - For each year:
        - Calculate the revenue for small cities by adding QuantitySold * Actual Selling Price for small cities (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.).
        - Calculate the revenue for medium cities by adding QuantitySold * Actual Selling Price for medium cities (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.).
        - Calculate the revenue for large cities by adding QuantitySold * Actual Selling Price for large cities (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.).
        - Calculate the revenue for extra large cities by adding QuantitySold * Actual Selling Price for extra large cities (i.e., If the SaleDate (i.e. Calendar.Date) is the OnSaleDate in OnSale; the actual selling price is OnSalePrice else the actual selling price is Retail Price.).
    - Sort by year in ascending order.
- When ready, user can select next action from Navigation Bar.

Membership Trend and Drill-down

Task Decomposition

Lock Types: Read-only lookups on Membership, Store, and City.
Number of Locks: 3
Enabling Conditions: None.
Frequency: Around 50 visits per day on main report and around 100 times per day on drill-down reports. All subtasks have different frequency and usually drill-down is more than main report.
Consistency (ACID): is not critical.

Table of Contents

Subtasks: Mother task is needed to coordinate subtasks. Order is critical, we probably more often have general query comes first and then drill down.

Abstract Code

- User click on ***Membership Trend*** button from **Main Menu** form.
- Run **View Memberships Sold per Year** task:
    - Find all Membership and retrieve MemberID and SingupDate.
    - Count total number of memberships sold for each year.
    - Display year and total number of memberships sold for that year.
    - Sort by year in ascending order.
- When ready, user can click drill-down or select next action from Navigation Bar.
-
- If user click ***drill-down*** button on a year, then run **Top and Bottom 25 Cities Drill-down** task:
    - Query for information about selected year where $year is stored in the web session.
    - Find all Membership sold in the selected year using Membership.SignupDate.
        - Find the City from which each Membership is purchased.
        - Retrieve MemberID, CityName, and State.
        - Count total number of memberships sold for each City in the selected year using MemberID.
    - The first section display the top 25 cities that sold the most memberships with selected year in the report header, CityName, State, and total number of memberships sold for that year.
        - Sort first by number of memberships in descending order then by CityName in ascending alphabetical order.
    - The second section display the bottom 25 cities that sold the least memberships with selected year in the report header, CityName, State, and total number of memberships sold for that year.
        - Sort first by number of memberships in ascending order then by CityName in ascending alphabetical order.
    - When ready, user can click drill-down or select next action from Navigation Bar.

    - If user click ***drill-down*** button on a city, then run **City to Store level Drill-down** task:
        - Query for information about selected city (‘$city’) and state (‘$state’) where is stored in the web session.
        - Find all Membership sold in the selected city and state during that year using City.CtiyName, State, and SignupDate.
            - Find all Store in the selected city and state.
            - Retrieve MemberID, CityName, State, and SignupDate.

Table of Contents

        - Count total number of memberships sold for each Store in the selected city and state during that year using MemberID.
    - Display selected year in the report header, StoreNumber, StreetAddress, ZipCode, CityName, State, and total number of memberships sold during that year in this store.
    - When ready, user can select next action from Navigation Bar.