Table of Contents

Data Types

Category

Attribute	Data Type	Nullable
Name	String	Not Null

City

Attribute	Data Type	Nullable
City_name	String	Not Null
Population	Integer	Not Null

State

Attribute	Data Type	Nullable
State_name	String	Not Null

Product

Attribute	Data Type	Nullable
PID	Integer	Not Null
Retail_price	Float	Not Null
Name	String	Not Null

Manager

Attribute	Data Type	Nullable
First_name	String	Not Null
Last_name	String	Not Null
Email_address	String	Not Null

Active_Manager

Attribute	Data Type	Nullable
First_name	String	Not Null
Last_name	String	Not Null
Email_address	String	Not Null

Inactive_Manager

Attribute	Data Type	Nullable
First_name	String	Not Null
Last_name	String	Not Null
Email_address	String	Not Null

Sale

Attribute	Data Type	Nullable
Sale_price	Float	Not Null
Sale_date	Date	Not Null

Store

Attribute	Data Type	Nullable
Store_number	Integer	Not Null
Phone_number	String	Not Null
Street_number	String	Not Null
Street_name	String	Not Null

Manufacturer

Attribute	Data Type	Nullable
Name	String	Not Null
Maximum_discount	Float	Null

Purchase_Day

Attribute	Data Type	Nullable
Date	Date	Not Null

Normal_Day

Attribute	Data Type	Nullable
Date	Date	Not Null

Holiday

Attribute	Data Type	Nullable
Date	Date	Not Null
Name	String	Not Null

Business Logic Constraints

1. Retail price is in effect unless there is a sale.
2. Maximum discount of 0% means the product cannot be placed on sale.
3. No product can be discounted for more than 90% of the retail price.
4. If a product is on sale, it's on sale in all stores at the same discount.
5. No sale price shall be higher than the retail price.
6. Sale prices must adhere to the maximum discount that the manufacturer has established.
7. If sale spans multiple days, a record will be added to the database for each day.
8. A manager cannot be removed from the database until they have been removed from their stores.
9. The price of the sale must be derived based on the date purchased and the quantity.

Report Dashboard

Task Decomposition

Lock Types: N/A
Number of Locks: N/A
Enabling Conditions: User has been authenticated by the system.
Frequency: Multiple times a day.
Consistency (ACID): Not critical, order is not critical.
Subtask: Mother task is not needed. No decomposition required.

Abstract Code 1

- Display form with below links to sections:
  - "Report 1 – Manufacturer's Product Report"
  - "Report 2 – Category Report"
  - "Report 3 – Actual versus Predicted Revenue for GPS units"
  - "Report 4 –Store Revenue by Year by State"
  - "Report 5 – Air Conditioners on Groundhog Day?"
  - "Report 6 – State with Highest Volume for each Category"
  - "Report 7 – Revenue by Population"

Abstract Code 2

- Upon:
  - Click Report 1 – Manufacturer's Product Report– Jump to **Generate Manufacturer's Product** task.
  - Click Report 2 – Category Report– Jump to **Generate Category** task.
  - Click Report 3 – Actual versus Predicted Revenue for GPS units– Jump to **Generate Actual versus Predicted Revenue** for GPS Units task.
  - Click Report 4 –Store Revenue by Year by State– **Jump to Generate Store Revenue by Year and State** task.
  - Click Report 5 – Air Conditioners on Groundhog Day? – Jump to **Generate Air Conditioners on Groundhog Day** task.
  - Click Report 6 – State with Highest Volume for each Category– Jump to **Generate State with Highest Volume for each Category** task.
  - Click Report 7 – Revenue by Population– Jump to **Get Revenue by Population** task.

Data Warehouse Admin

Task Decomposition

Lock Types: N/A
Number of Locks: N/A
Enabling Conditions: User has been authenticated by the system.
Frequency: Weekly or ad-hoc when Managers, Holidays or City Population needs to be updated.
Consistency (ACID): Not critical, order is not critical.
Subtask: Mother task is not needed. No decomposition required.

Abstract Code 1

- Display form with below links to sections:
  - "Update City Population"
  - "Maintain Holiday(s)"
  - "Maintain Manage(s)"

Abstract Code 2

- Upon:
  - Click Update City Population –Jump to **Update City Population** task.
  - Click Maintain Holiday(s) – Jump to **Maintain Holiday** task.
  - Click Maintain Manage(s) – Jump to **Maintain Manager** task.

Report 1 - Manufacturer's Product Report

Task Decomposition

Lock Types: Read-only: MANUFACTURER, PRODUCT
Number of Locks: 2
Enabling Conditions: User runs the Manufacturer's Product Report from the Report Dashboard
Frequency: Ad-hoc depending on business needs (seldom)
Consistency (ACID): Should not be run during update cycles.
Subtasks: Mother task not needed. Decomposition not required.

Abstract Code

- User runs the **Manufacturer's Product Report** task:
  - Query for all MANUFACTURER by Name
    - For each MANUFACTURER:
      - Display the Name
      - Find each PRODUCT for the MANUFACTURER
      - For each PRODUCT:
        - Find the Retail_price of PRODUCT
      - Count total number of PRODUCT for MANUFACTURER
      - Sum all Retail_price of PRODUCT for MANUFACTURER
      - Average sum of all Retail_price over total number of PRODUCT
      - Display total number of PRODUCT for MANUFACTURER
      - Find minimum Retail_price for MANUFACTURER
        - Display minimum Retail_price for MANUFACTURER
      - Find maximum Retail_price for MANUFACTURER
        - Display maximum Retail_price for MANUFACTURER
      - Run **SubReport 1-Drill Down Detail** task
        - Display **Detail** button for accessing report
    - Sort Average of all Retail_price for all MANUFACTURER descending
      - Display first 100 MANUFACTURER

SubReport 1 - Drill Down Detail

Task Decomposition

Lock Types: MANUFACTURER, PRODUCT, CATEGORY
Number of Locks: 3
Enabling Conditions: User clicks for details in the Manufacturer's Product Report
Frequency: Ad-hoc depending on business needs (seldom).
Consistency (ACID): Should not be run during update cycles.
Subtasks: Mother task not needed. Decomposition not required

Abstract Code

- User clicks for details in the **Manufacturer's Product Report** to display manufacturer details
- Display summary information from **Manufacturer's Product Report**
- Run the **Generate Drill Down Report 1** task:
    - Query for MANUFACTURER by Name
        - Display the Name
        - Find Maximum_discount
        - Display Maximum_discount
        - Find each PRODUCT for the MANUFACTURER
        - For each PRODUCT:
            - Find PID of PRODUCT
            - Display PID
            - Find Name of PRODUCT
            - Display Name
            - Find each CATEGORY for PRODUCT
                - For each CATEGORY:
                - Concatenate into one comma-delimited value
            - Display concatenated CATEGORY for PRODUCT
            - Find the Retail_price of PRODUCT
            - Display Retail_price
        - Display each PRODUCT, sorted descending by Price

Report 2 - Category Report

Task Decomposition

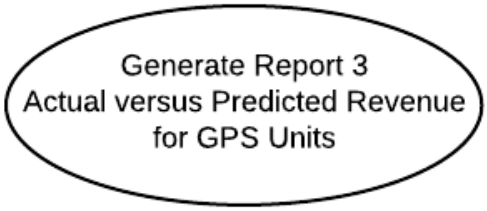
Lock Types: Read-only: MANUFACTURER, PRODUCT, CATEGORY
Number of Locks: 3
Enabling Conditions: User runs the Category Report from the Report Dashboard
Frequency: Ad-hoc depending on business needs (seldom)
Consistency (ACID): Should not be run during update cycles.
Subtasks: Mother task not needed. Decomposition not required.

Abstract Code

- User runs the **Category Report** task from the Report Dashboard:
  - Query for all categories by Name
    - For each CATEGORY:
      - Find each Product in the CATEGORY:
        - Find the MANUFACTURER of PRODUCT
        - Find the Retail_price of PRODUCT
      - Count total number of PRODUCT in CATEGORY
      - Sum all Retail_price of PRODUCT in CATEGORY
      - Average sum of all Retail_price over total number of PRODUCT
      - Count unique MANUFACTURER in CATEGORY
      - Display total number of PRODUCT in CATEGORY
      - Display total number of unique MANUFACTURER in CATEGORY
      - Display average of Retail_price in CATEGORY
    - Display each CATEGORY, sorted ascending by Name

Report 3 - Actual versus Predicted Revenue for GPS Units

Task Decomposition



Lock Types: Read-only: PRODUCT, CATEGORY, PURCHASE_DAY, SALE, STORE
Number of Locks: 5
Enabling Conditions: Triggered by running Report 3, "Actual versus Predicted Revenue for GPS", from UI
Frequency: Ad-hoc depending on business needs (seldom)
Consistency (ACID): Should not be run during update cycles.
Subtasks: Mother task not needed.

Abstract Code

- User clicks the "***Actual vs. Predicted Revenue for GPS Units Report***" button
- For each PRODUCT.Name.BELONGS_TO.CATEGORY.Name = "GPS"
    - For each PURCHASE_DAY.Date
        - Sum *Store Quantity* for all STORE.Store_number on that date
        - If there is a matching PRODUCT.HAS_A.SALE.Sale_date /*sale price*/
            - Add (75% * *Store Quantity)* to *Predicted Sales Quantity*
            - Add *Store Quantity* to *DIscounted Sales Quantity*
            - Add (*Store Quantity* * PRODUCT.HAS_A.SALE.Sale_price) to *Actual Revenue*
        - Otherwise /*no sale*/
            - Add *Store Quantity* to Predicted Sales Quantity
            - Add *Store Quantity* to *Retail Sales Quantity*
            - Add (*Store Quantity* * PRODUCT.Retail_price) to *Actual Revenue*
    - Calculate *Predicted Revenue* = (*Predicted Sales Quantity* * PRODUCT.Retail_Price)
    - Calculate *Delta* = (*Actual Revenue* – *Predicted Revenue)*
    - Store PRODUCT.Name, PRODUCT.PID, PRODUCT.Retail_price, *Discounted Sales Quantity*, *Retail Sales Quantity*, *Actual Revenue*, *Predicted Revenue*, and *Delta*
- Sort *Delta*s from high to low
- Display report headers
- For each *Delta* where *Delta* > 5000 or *Delta* < -5000
    - Display PRODUCT.PID, PRODUCT.Name, PRODUCT.Retail_price, *Retail Sales Quantity*, *Actual Revenue*, *Predicted Revenue*, *Delta*
- Display report footers

Report 4 - Store Revenue By Year by State

Task Decomposition

Lock Types: Read-only: STORE, PRODUCT, CITY, PRODUCT, SALE, PURCHASE_DAY
Number of Locks: 6
Enabling Conditions: Triggered by running Report 4, "Store Revenue by State", from UI
Frequency: Ad-hoc depending on business needs (seldom)
Consistency (ACID): Should not be run during update cycles.
Subtasks: Mother task not needed. Decomposition not required

Abstract Code

- User clicks the "***Store Revenue by Year by State***" button
- Create *list of states* from CITY.STATE.{}
- Present drop-down menu containing *list of states*
- User selects *State* from menu
- For each STORE LOCATED_IN *State*
    - For each PURCHASE_DAY.Date
        - Calculate *Year* from PURCHASE_DAY.Date
        - For each PRODUCT
            - If PRODUCT.HAS_A.SALE.Date = PURCHASE_DAY.Date, add (PRODUCT.PURCHASE_DAY.Quantity * PRODUCT.HAS_A.SALE.Sale_Price) to *Store Revenue.Year*
            - else add (PRODUCT.PURCHASE_DAY.Quantity * PRODUCT.Retail_price) to *Store Revenue.Year*
    - Store STORE.Store_number, STORE.Street_address, STORE.LOCATED_IN.City_name, *Store Revenue.{}*
- For each *Year* from oldest to newest
    - Sort by *Store Revenue.Year*
    - Display report headers
    - For each STORE.Store_number
        - Display STORE.Store_number, STORE.Street_address, STORE.LOCATED_IN.City_name, *Year*, *Store Revenue.Year*
    - Display report footers

Report 5 - Air Conditioners on Groundhog Day

Task Decomposition

Lock Types: Read-only: PRODUCT, CATEGORY, PURCHASE_DAY
Number of Locks: 3
Enabling Conditions: Triggered by running Report 5, "Air Conditioners on Groundhog", from UI
Frequency: Ad-hoc depending on business needs (seldom)
Consistency (ACID): Should not be run during update cycles.
Subtasks: Mother task not needed. Decomposition not required

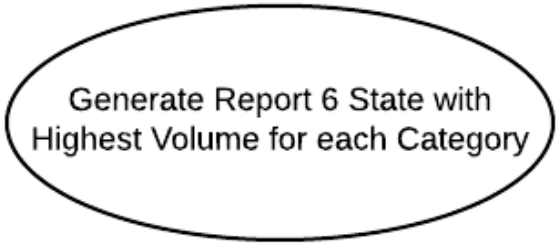
Abstract Code

- User clicks the "***Air Conditioners on Groundhog Day Report***" button
- For each PRODUCT.Name.BELONGS_TO .CATEGORY.Name = "Air Conditioner"
    - For each PURCHASE_DAY.Date
        - Calculate *Year*
        - Add Product.PURCHASE_DAY.Quantity to *Total Units.Year*
    - For each PURCHASE_DAY.Date that is a February 2
        - Calculate *Year*
        - add PRODUCT.PURCHASE_DAY.Quantity to *Groundhog Units.Year*
- For each *Year* (from lowest to highest)
    - Display *Year*, *Total Units.Year*, =(*Total Units.Year* / 365), *Groundhog Units.Year*

    /* this assumes every year has air conditioner sales, and that every year has air conditioner sales on Groundhog Day */

Report 6 - State with Highest Volume for each Category

Task Decomposition



Lock Types: There are five read-only lookups: CATEGORY, CITY, STORE, PURCHASE_DAY, PRODUCT
Number of Locks: 5
Enabling Conditions: Triggered by running the Report 6 command from the Report Dashboard
Frequency: This report will be viewed monthly
Consistency (ACID): Consistency is not critical, report is ran for previous month's data. Order is critical, a list of categories must be created first
Subtasks: Mother task is not needed. Decomposition not required.

Abstract Code

- Show ***year and month drop down menus*** populated with available dates from database
- Show ***"generate report button"***
- Upon choosing *a year and month* and clicking the ***generate report button***:
    - Query the database to return the State and the highest sales in each Category
    - Display a row with the Category name, the State with highest sales in that Category, the number of sales in that Category, ***a button*** that generates SubReport 6.
        - *If* a Category has more than one State as equal in highest sales, Display a row for each state.

SubReport 6 - Drill Down Detail:

Task Decomposition

Lock Types: There are three read-only lookups: STORES, MANAGERS, CITY
Number of Locks: 3
Enabling Conditions: Triggered by clicking ***a button*** from a row in Report 6.
Frequency: This report may be viewed monthly.
Consistency (ACID): Consistency is not critical, report ran for previous months' data. Order is critical, Store Data must be gathered first
Subtasks: Mother task is not needed. Decomposition not required.

Abstract Code

- Upon a ***button*** click in report 6:
  - Show ***a header*** containing original criteria from parent report (*category, year/month, state*).
  - Show column headings of: Store address, Store id, city, Manager name, Manager email
  - Query database for Stores that are located in the passed-in state from parent report
    - get the Stores' address and store_id
  - Query database for City of the store
  - Query database for all Managers located in a store
    - get the Managers name and email_address
  - *for each* store in the arranged list of stores Display a row with Store name, Store id, City, Manager name, Manager email
    - *if* a Store has multiple Managers, display a new row *for each* manager in that store

Report 7 - Revenue by Population

Task Decomposition

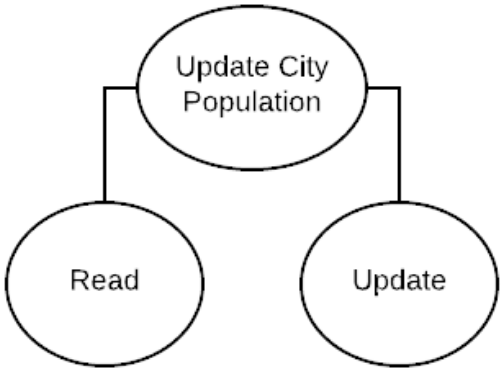
Lock Types: There are five read-only lookups: CITY, STORE, PURCHASE_DAY, PRODUCT, SALE
Number of Locks: 5
Enabling Conditions: Triggered by running the Report 7 command from the Report Dashboard
Frequency: Ad-hoc depending on business needs
Consistency (ACID): Consistency is not critical, sales data that is current within the past day will suffice. Order is not critical.
Subtasks: Mother task is not needed. Decomposition not required.

Abstract Code

- **Query** the database for the number_of_years in the database
  - Arrange years in ascending order (oldest to newest)
- *For each* year in the database:
  - Query the database to get the population of each City.
    - Arrange cities into predefined categories in ascending order (smallest to largest)
  - Query database to get the revenue for each store in a City.
  - Calculate average_revenue for population category
  - Display the data for the current year.
    - Initially the display will have city categories as the rows and years as the columns
- Show ***pivot button*** for switching "years"/"city category" as either columns or rows.
  - Upon pressing the ***button*** switch which category is rows.
    - If rows are represented by years, make years the columns and the city categories the rows
    - Else make rows the years and city categories the columns.

Update City Population

Task Decomposition



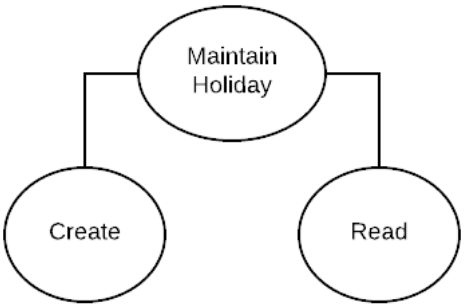
Lock Types: Read only lookup for the City that is updated.
Number of Locks: One schema construct needs to be considered. (CITY)
Enabling Conditions: Edits enabled when user views cities.
Frequency: Annually or when new population data is made available.
Consistency (ACID): Given the frequency for changes, lock the resource for reads while we update.
Subtask: The Update City Population task has been broken down into two tasks per the specification: Read and Update.

Abstract Code

- User clicks on the Update City link from the **Data Warehouse Admin** form.
- Show **Cities** form
- For each City in CITY.
  - Display CITY attributes: CITY.City_name, CITY.State and CITY.Population
  - User updates population ('$population') input field to desired value.
  - When the **Update** button is clicked
    - Validate the *$population* is an integer
      - If valid: Update the populate for CITY.Population to the value stored in *$population.*
      - Else: Prompt user to use a valid integer, validate and Update.

Maintain Holiday

Task Decomposition



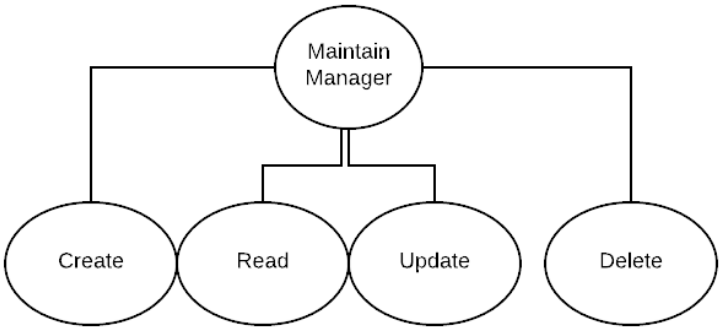
Lock Types: Read and write locks depending on the task.
Number of Locks: One (HOLIDAY)
Enabling Conditions: All three are enabled via the Manager form. This form will support create, read, update and delete for rows in the manager table. (CRUD)
Frequency: Yearly or monthly at most.
Consistency (ACID): not critical, order is not critical.
Subtask: The Maintain Holiday task has been broken down into two tasks per the specification: Create, Read.

Abstract Code

- User clicks on the *Maintain Holiday* link from the **Data Warehouse Admin** form.
    - Query database for all holidays and sort in ascending order by date.
        - The **Maintain/View Holiday** form will show with a table of holidays sorted in ascending order by date.
        -
    - Click **Add** button – User will be prompted for Holiday name $holiday_name and $date. The **CREATE** task will be run with $holiday_name and $date.
        - User selects either a *$store* or a manager name $manager_name
            - If holiday exists: query databases for manager where Holiday.name = $holiday_name and Date=$date
                - Inform the user that the holiday already exists in the database.
            - Else: Add the new Holiday into the database.

Maintain Manager

Task Decomposition



Lock Types: Read and write locks depending on the task.
Number of Locks: One (MANAGER)
Enabling Conditions: All three are enabled via the Manager form. This form will support create, read, update and delete for rows in the manager table. (CRUD)
Frequency: Infrequently; whenever a manager leaves/joins a store location.
Consistency (ACID): A manager cannot be removed before the association to all stores has been removed.
Subtask: The Maintain Manager task has been broken down into four distinct tasks: Create, Read, Update and Delete. These tasks are carried out depending on what the user selects in the Manager Form.

Abstract Code

- User clicks on the Maintain Manager link from the **Data Warehouse Admin** form.
  - The **Manager** form will show a table of managers along with a drop down for Stores and Manager Names for selection.
    - Query database for unique STORE.Store_number
    - Query database for unique MANAGER.Manager_name
  - Populate Drop downs with results from queries
  - Click **Add** button – User will be prompted for MANAGER.name *$manager* and STORE.Store_number *$store*. The **CREATE** task will be run with these arguments.
    - User selects either a *$store* or a manager name $manager_name
    - If store: query databases for manager where STORE.Store_number = $store
    - If manager: query databases for manager where MANAGER.Manager_name = $name

- Update the **Manager** form with information retrieved from the database. If the manager manages multiple stores, each row will be displayed on the table with the STORE.Store_number.
- Click **Delete Manager** button – Present the user with confirmation that the manager and his associations will be removed from database, if yes proceed, else cancel
- Click **Unassign** button – This will remove the association between the MANAGER and STORE.
- Click **Assign** button – The user will be prompted to select the STORE.Store_number from a drop down. Once selected the database will create the association with MANAGER and STORE.