Table of Contents

Data Types

Manager

Attribute	**Data type**	**Nullable**
Name	String	Not Null
EmailAddress	String	Not Null
Active	Boolean	Not Null

Store

Attribute	**Data type**	**Nullable**
StoreNumber	String	Not Null
PhoneNumber	String	Not Null
StreetAddress	String	Not Null
City	String	Not Null
Manager	String	Null

City

Attribute	**Data type**	**Nullable**
Population	Float	Not Null
State	String	Not Null
CityName	String	Not Null

Product

Attribute	**Data type**	**Nullable**

Name	String	Not Null
PID	String	Not Null
RetailPrice	Float	Not Null

Categories

Attribute	**Data type**	**Nullable**
Name	String	Not Null
PID	String	Not Null

Manufacturer

Attribute	**Data type**	**Nullable**
Name	String	Not Null
CapDiscount	Float	Null
PID	String	Not Null

Transaction

Attribute	**Data type**	**Nullable**
Quantity	Float	Not Null
PID	String	Not Null
Date	Date	Not Null
StoreNumber	String	Not Null

Sale

Attribute	**Data type**	**Nullable**
SalesDate	Date	Not Null
PID	Float	Not Null
SalePrice	Float	Not Null

Holiday

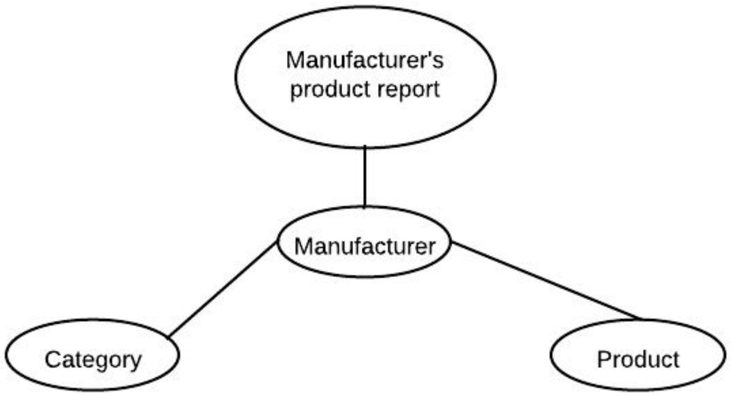
Attribute	**Data type**	**Nullable**
Date	Date	Not Null
HolidayName	String	Not Null

Business Logic Constraints

1. Sale price cannot be higher than retail price.
2. Every manufacturer could provide the maximum discount which should lower than 90% or not.

1 Manufacturer's Product Report

Task Decomposition



Lock Types: lookup three tables: manufacturer, product, category table. Read Only
Number of Locks: Several different schema constructs are needed
Enabling Conditions: None
Frequency: different frequencies
Consistency (ACID): not critical, order is not critical
Subtasks: All tasks must be done. Mother task is required to coordinate subtasks. Order is not necessary.

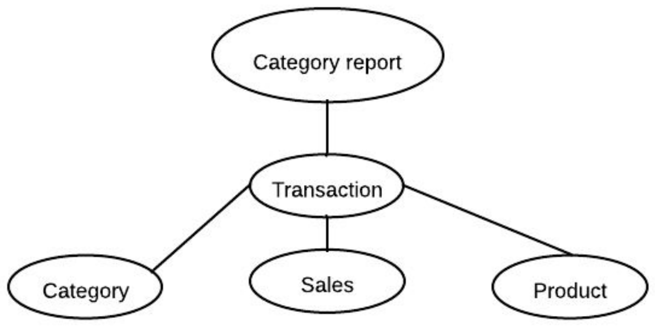
Abstract Code

- User clicked the ***Manufacturer Product Report*** button:

- Run the **Manufacturer Product Report** task**:**
  - Based on manufacturer table, look up product information such as retail price from product table by PID
  - Group by manufacturer, select manufacture, count total number of product, calculate the average, max and min price.
  - Order the result by calculated average price descending, only list top 100
- For each row of the master table, when mouse is over the manufacturer name, there is additional information such as maximum discount shown.
- For each row in the master table, if click manufacturer name, a new table pops up with product information of that manufacturer:
  - Based on manufacturer table, find the PID for the related manufacturer. Search product information for those PID in the product table, such as price, product name
  - Based on the selected PID, find the category from the category table. Concatenated if multiple categories are returned.
  - Order the product by retail price descending.
- it shows the report table on the screen. Each row represents a manufacturer, column represents the related information.

2 Category Report

Task Decomposition



Lock Types: lookup four tables: transaction, product, sales, category table. Read Only
Number of Locks: Several different schema constructs are needed
Enabling Conditions: None
Frequency: Different frequencies
Consistency (ACID): not critical, order is not critical
Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is shown as follows.

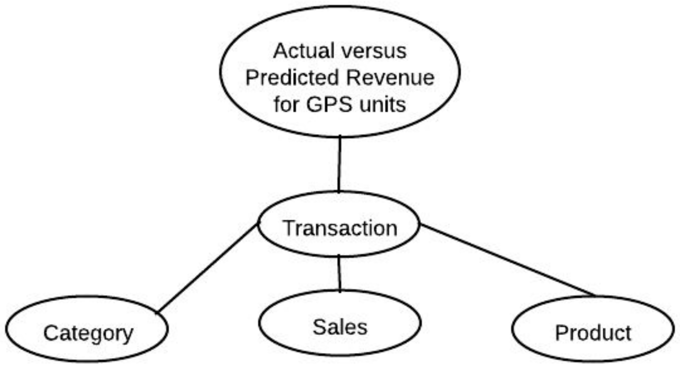
Abstract Code

- User clicked the ***Category Report*** button:
- Run the **Category Report** task:

- Based on category table, merge with product and manufacturer table by PID as key to get retail price and manufacturer information. Group by category, count unique PID, count unique manufacturer and calculate average retail price.

3 Actual versus Predicted Revenue for GPS units

Task Decomposition



Lock Types: lookup four tables: transaction, product, sales, category table. Read Only
Number of Locks: Several different schema constructs are needed
Enabling Conditions: None
Frequency: different frequencies
Consistency (ACID): not critical, order is not critical
Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is shown as follows.

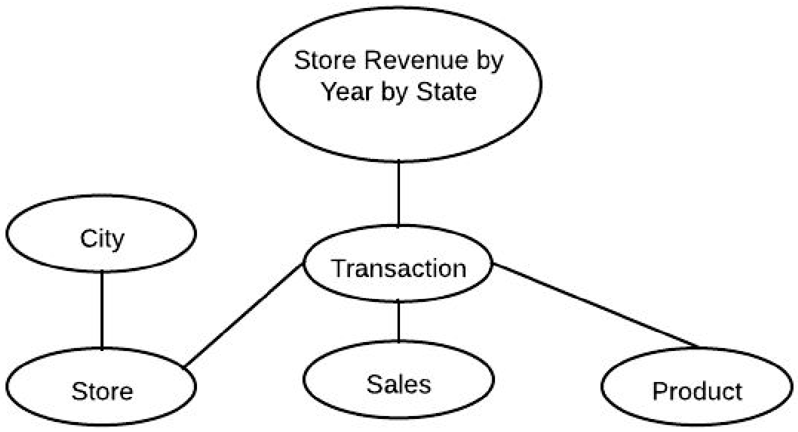
Abstract Code

- User clicked the ***Actual versus Predicted Revenue for GPS units*** button:
- Run the **Actual versus Predicted Revenue for GPS units** task:
    - Find Product ID in the GPS Category from Category table.
    - Find related transactions from *Transaction* table using selected Product ID.
    - Find regular price from Product table using selected Product ID.
    - Find sale price from Sales table using related transaction date and Product ID from Transa
    - Create Boolean attribute is_sale to indicate whether a sale or not. Make transaction price as sale_price if is_sale=1, otherwise transaction price is retail price.
- Summarize information by product and generate **report**
    - Sum up total quantity, sale quantity if is_sale=1, regular quantity if is_sale=0
    - Sum up the transaction price * total quantity to get actual revenue
      Sum up the retail price *(quantity if is_sale=0, quantity*0.75 if is_sale=1) as predict revenue
    - Get the revenue difference as predict revenue – actual revenue
    - Get the revenue difference as predict revenue – actual revenue

- For **report**, only keep PID with revenue difference>5000, sort by revenue difference in descending order.

4 Store Revenue by Year by State

Task Decomposition



Lock Types: lookup five tables: store, city, transaction, product and sales table. Read Only
Number of Locks: Several different schema constructs are needed
Enabling Conditions: None
Frequency: different frequencies
Consistency (ACID): not critical, order is not critical
Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is shown as follows. The first two steps can be down in parallel then do summary.

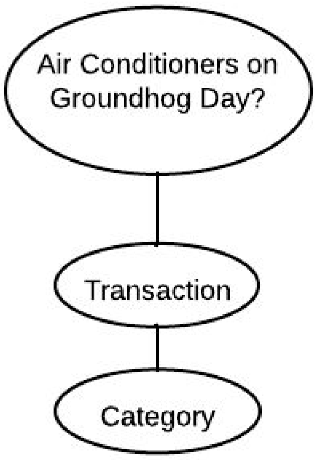
Abstract Code

- User clicked the ***Store Revenue by Year by State*** button. Run the **Store Revenue by Year by State** task:
- Enrich the transaction with the city and state
    - Merge city and store table by key: city to get the state
    - Merge the transaction and the new store table by key: StoreNumber. So for each transaction, it has the sold store,city and the state
- Enrich the transaction with price
    - Merge the new transaction table with product table by key: PID, to get retail price
    - Merge the transaction table with sales table by key: PID and Date, to get sales price
    - Create transaction price as sales price if there is a sale, otherwise regular price
- Summarize the result

  - Based on the enriched transaction table, group by store and year, sum up the revenue calculated by product of transaction price times quantity.
  - Also keep the store ID, address, city name and state
  - Sort the result by year ascending and revenue descending
- Generate **report** with a **drop down box** to select state
  - For each state, list all the stores with store ID, address, city name, sales year and total revenue. Order has been sorted in the previous step

5 Air Conditioners on Groundhog Day?

Task Decomposition



Lock Types: lookup two tables: transaction and category. Read Only
Number of Locks: Several different schema constructs are needed
Enabling Conditions: None
Frequency: different frequencies
Consistency (ACID): not critical, order is not critical
Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is shown as follows. The first two steps can be down in parallel then do summary.

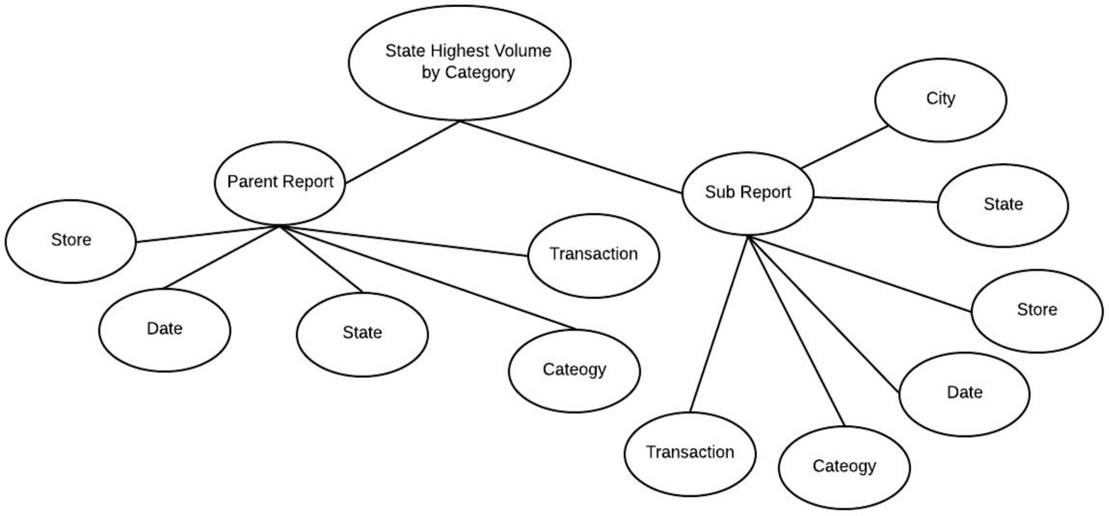
Abstract Code

- User clicked the ***Air Conditioners on Groundhog Day*** button. Run the **Air Conditioners on Groundhog Day** task:
- Select transactions only on category “air conditioning”
  - Merge with transaction table and category table by key: PID, only select category as “air conditioning”
  - Get the transaction year, and also create attribute is_groundhog=1 if it is on Feb 2nd.
- Summarize the result and generate **report**

  - Group by year, sum up the total sales quantity divided by 365 to get average sales per day, sum up quantity*is_groundhog to get sales quantity at groundhog day
  - Sort by year ascending

6 State with Highest Volume for each Category

Task Decomposition



Lock Types: lookup five tables: store, city, category, transaction, manager table. Read Only

Number of Locks: Several different schema constructs are needed

Enabling Conditions: None

Frequency: different frequencies

Consistency (ACID): not critical, order is not critical

Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

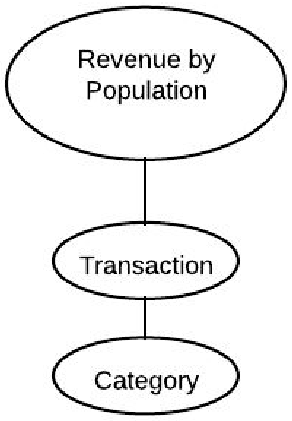
User clicked the ***State with Highest Volume for each Category*** button:

- Run the **State with Highest Volume for each Category**:
- Choose Date to get *year and month.*
- Create a master table which includes all the necessary attributes for the transaction
  - First merge store and city table by key: city to get each store's state, then merge with the transaction table by: key store number. So each transaction record has sold store and state.
  - Get the year and month for each transaction
- Get transaction information for category table
  - Merge the category table with new transaction table by PID. So each category has the transaction information for all the products it sold.
- Summarize all the information and generate **report**

  - Based on the enriched category table, group by year and month, as well as category and state, calculate the summation of sold quantity.
  - Sort the output by sold quantity descending and only select the highest state.
  - Create summary view, which has a drop down window to select year and month, and the report will show all the categories' names, the highest sold state and quantity.Sort by **category name** ascending
- Drill-down details with store and manager information
  - Each row of the report has a hyperlink to include store **IDs, addresses, and cities, managers names and email addresses**
  - Sort **store IDs** ascending, parent header include **category, year/month, state**

7 Revenue by Population

Task Decomposition



Lock Types: lookup five tables: store, city, transaction, product and sales table. Read Only
Number of Locks: Several different schema constructs are needed
Enabling Conditions: None
Frequency: different frequencies
Consistency (ACID): not critical, order is not critical
Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

- User clicked the ***Revenue by Population*** button. Run the **Revenue by Population** task:
- Transaction with the city and population
  - Based on the criterion, create the attribute CitySize of small, medium, large and extra large based on population. Merge city and store table by key: city to enrich store table.
  - Merge the transaction and the new store table by key: StoreNumber. So for each transaction, it has the sold city and the citySize,
- Enrich the transaction with price
  - Merge the new transaction table with product table by key: PID, to get retail price

    - Merge the transaction table with sales table by key: PID and Date, to get sales price
    - Create transaction price as sales price if there is a sale, otherwise regular price
- Summarize the result and create **pivot table**
    - Based on the enriched transaction table, group by year and citySize, sum up the product of transaction price and quantity to get the revenue.
    - Make a pivot table with year as row, citysize as column. Each cell represents the total revenue.