**CS 6400 Database Project**

**BuzzBuy Data Warehouse**

**Team 34**

**Table of Contents:**

BuzzBuy Data Warehouse Data Types

* [Data Types](#datatypes)

BuzzBuy Data Warehouse Constraints

* [Business Logic Constraints](#businesslogic)

Task Decomposition with Abstract Code:

* [Login](#login)

* [View Statistics](#stats)
* [View Holidays](#holidays)
* [Edit Holidays](#edithol)
* [View Reports](#viewreport)
* [View Audit Logs](#auditlog)

The first phase of the project deals with the Analysis and Specifications.

**Data Types:**

|  |  |  |
| --- | --- | --- |
| Entity.Attribute | Type | Format |
| User.EmployeeID | String | Unique |
| User.FirstName | String |  |
| User.Password | String |  |
| District.DistrictNumber | Integer | Unique |
| AuditLog.ReportName | String |  |
| AuditLog.TimeStamp | DateTime |  |
| Holidays.HolidayName | String |  |
| Holidays.Date | Date |  |
| Store.Number | integer | Unique |
| Store.PhoneNumber | string or integer | 10 digit phone number, (\_ \_ \_)-\_ \_ \_-\_ \_ \_ \_ |
| Store.District | String |  |
| Store.City | String | City name and City state |
| City.CityName | String |  |
| City.State | String | 2 letter abbreviation, ex. TX |
| Product.PID | String | To use much wider range of unique identifier, we use a string instead of using integer.  Unique |
| Product.RetailPrice | Float |  |
| Product.Name | String |  |
| Product.ManufacturerName | String |  |
| Manufacturer.Name | String | Unique |
| Category.Name | String | Unique |
| Categories.PID | String | Categories – a relationship mapping table |
| Categories.Name | String | Category Name |
| Discount.DiscountDate | Date |  |
| Discount.DiscountPrice | Float |  |
| ProductDiscount.PID | String | Product-Discount relationship mapping table |
| ProductDiscount.DiscountDate | Date |  |
| ProductDiscount.DiscountPrice | Float |  |
| SellsTransactions.QuantitySold | Integer |  |
| SellsTransactions.DateSold | Date |  |
| SellsTransactions.PID | String |  |
|  |  |  |

**Business Logic Constraints**

* An audit log entry must be created each time a report is viewed
* If a product in discounted, it has the same price in all stores
* The retail price is in effect unless there is a temporary, promotional discount
* If a product is discounted for multiple days in a row, then a record is stored in the data warehouse for each day

**Task Decomposition & Abstract Code**

Login

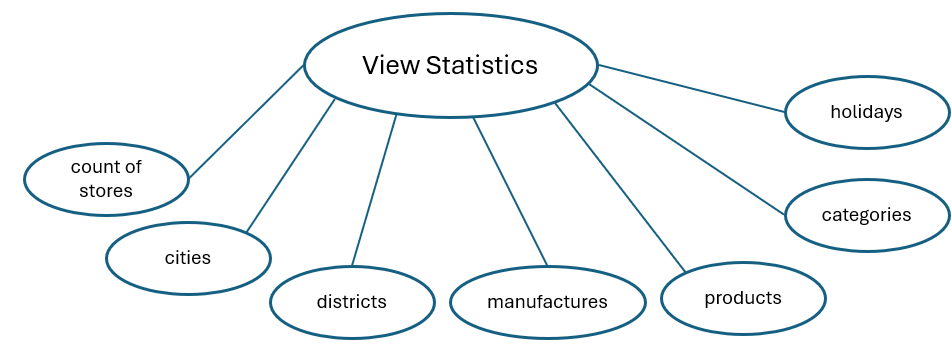
Task Decomp:

* **Lock Types**: Read-Only **User** Table
* **Number of Locks**: Single
* **Enabling Conditions:** None
* **Frequency**: Frequent
* **Consistency:** Not Critical, order is not critical
* **Subtasks:** Subtask is not needed, no further decomposition needed

Abstract Code:

* User enters employee id and password (last 4 ssn + “-” + LastName) input fields
* If data validation is success for both employee id and password input fields, then:
  + When Login button is clicked:
    - If User record is found but user password != (last 4 ssn + “-” + LastName)
      * Go back to **Login** form with error message
    - Else:
      * Store login information as session variable (EmployeeID)
      * Go to **Main Menu** Form

View Statistics

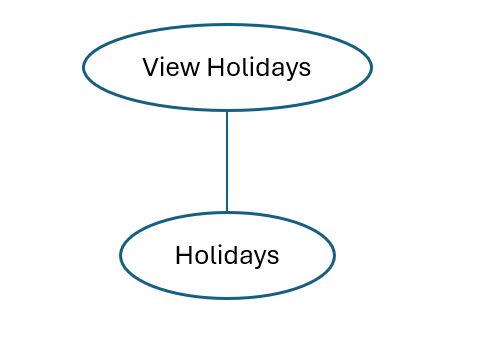


Task Decomp:

* **Lock Types:** Read-Only Lookups
* **Number of Locks:** Several schema constructs are needed
* **Enabling Conditions:** Successful login
* **Frequency:** Occassional (Upon successful login)
* **Consistency:** Not Critical
* **Subtasks:**
  + List available reports with navigation options.

Abstract Code:

* Display Data: Count of Stores, Cities, districts, manufacturers, products, categories, and holidays
* Show “***Manufacturer’s Product Report***”, “***Category Report****”*, “***Actual versus Predicted Revenue For GPS units***”, *“****Air Conditioners on Groundhog Day?****”*, *“****Store Revenue by Year by State***”, “***District with Highest Volume for each Category****”,* “***Revenue by Population***”, *“****Edit Holiday****”,* and *“****View Audit Log****”* Links
* Upon:
  + Click ***Manufacturer’s Product Report*** button- Jump to **View Reports** Task
  + Click ***Category Report*** button- Jump to **View Reports** Task
  + Click ***Actual versus Predicted Revenue For GPS units*** button- Jump to **View Reports** Task
  + Click ***Air Conditioners on Groundhog Day?*** button- Jump to **View Reports** Task
  + Click ***Store Revenue by Year by State*** button- Jump to **View Reports** Task
  + Click ***District with Highest Volume for each Category*** button- Jump to **View Reports** Task
  + Click ***Revenue by Population*** button- Jump to **View Reports** Task
  + Click ***Edit Holiday*** button- Jump to **Edit Holiday** Task
  + Click ***View Audit Log*** button- Jump to View **Audit Log** Task

View Holidays

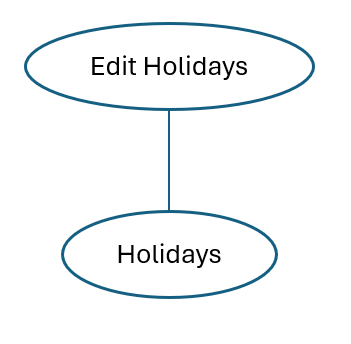
Task Decomp:

* **Lock Types**: 2 Read-Only Lookups of HolidayName and HolidayDate from **Holiday** Table
* **Number of Locks:** Several different schema constructs are needed
* **Enabling Conditions:** Both enabled to user’s login screen
* **Frequency:** Occasionally
* **Consistency:** Not Critical
* **Subtasks:**
  + User clicked ***View Holidays*** button from **Main Menu**
  + Run **View Holiday** task: query information about Holidays

Abstract Code

* Check if user has been granted to access to all districts
* If all districts are accessible, then:
  + View all holidays

Edit Holidays

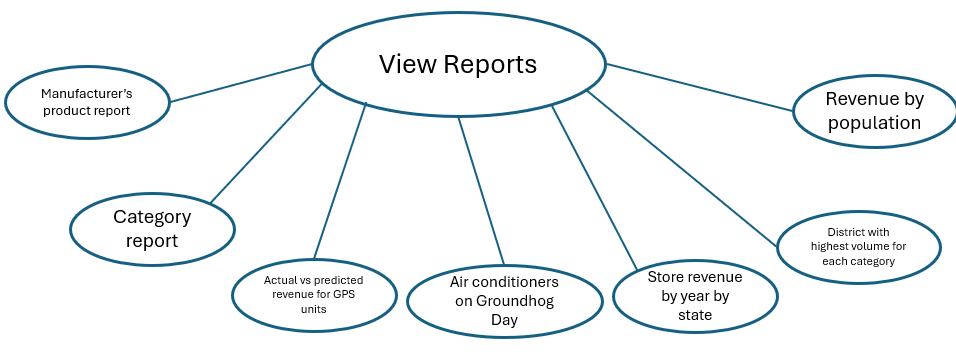
Task Decomp:

* **Lock Types**: Write Locks on **Holiday** Table
* **Number of Locks**: Several schema constructs are needed
* **Enabling Conditions**: User has access to all districts
* **Frequency**: Infrequent (Only certain users can add holidays)
* **Consistency**: Critical (To prevent duplicate holidays)
* **Subtasks**:
  + User clicks ***Edit Holidays*** button from the **Main Menu**.
  + Validate user's permission to add holidays.
  + If permission is valid:
    - Check if the holiday already exists for the date.
    - If not, allow user to add a new holiday.
    - Record which user added the holiday.

Abstract Code:

* Check if user has been granted to access to all districts
* If all districts are accessible, then:
  + View all holidays
  + Select a holiday to edit
  + “Delete” is also supported from the same view ***“All Holidays***”

View Reports



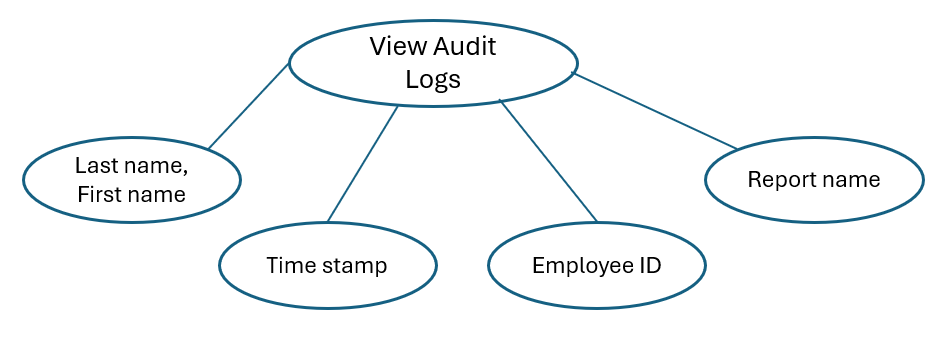
Task Decomp:

* **Lock Types:** Read-Only Lookups
* **Number of Locks**: Several schema constructs are needed
* **Enabling Conditions**: Successful login
* **Frequency**: Occasional (Reports viewed based on user needs)
* **Consistency**: Not Critical
* **Subtasks**:
  + User selects a report from the list.
  + Query the data warehouse for the selected report.
  + Display the report data to the user.

Abstract Code:

* Check if user has been granted to access to Reports
* If a user is granted to access, then:
  + View all available reports
  + Choose a report to generate a report
  + Upon completion of generating a report
    - User is able to export the output to different formats of files (PDF or CSV)

View Audit Logs



Task Decomp:

* **Lock Types:** Read-Only Lookups on **Audit Log** Table
* **Number of Locks:** Several schema constructs are needed
* **Enabling Conditions**: User has audit log permissions
* **Frequency**: Infrequent (Only users with permissions can view)
* **Consistency**: Not Critical
* **Subtasks**:
  + User clicks ***View audit log***button from the **Main Menu**.
  + Query the most recent audit log records.
  + Display the audit log records in a table format, highlighting entries where the user has access to all districts.

Abstract Code:

* Check if user has been granted to access to Audit Logs
* If a user is granted to access, then:
  + View all available audit logs
    - Search by Date, User, ReportName
    - Sort by Date
  + Choose an audit log to view
  + A selected log or multiple logs can be printed or exported to other file formats