**Football Boots Store**

1. **Introduction**

This document outlines the development of an e-commerce application for a Football Boots Store. The project encompasses the design of OLTP and OLAP database systems, data loading from CSV files, and the creation of visual reports. The aim is to create an efficient platform for managing product inventory, processing orders, and generating insightful sales data.

1. **Instructions**

Here are just the instructions how to use scripts and other data. For better understanding, their full description is provided below in steps 3-6.

1. *OLTP creation*

The tables, indexes, functions, roles and triggers for OLTP database are created by running script from file ‘1.2-OLTP-Solution.sql’. This script will create all the necessary components. At the end of the file, there are also some ‘test-cases’ to check functionality.

1. *OLTP data loading*

The tables should be populated with data using the script from the file ‘1.4-Data-Loading.sql’. To ensure correct output, all the file paths in the script should be updated to match your actual ones. Save folders ‘Dataset-1’ and ‘Dataset-2’ and provide file paths to them. Execute the script, and the tables will be populated.

1. *OLAP creation*

The tables, indexes, functions, and triggers for the OLAP database are created by running the script from the file ‘2.1-OLAP-Solution.sql’. This script will create all the necessary components.

1. *ETL to fill OLAP database*

To run the ETL script, open the solution from the ETL-Script folder. It contains C# code created to perform ETL on data. All you need to do is update the database connection strings in both contexts (Model -> FotballBootsStoreContext.cs and Model2 -> FootballBootsStoreOlapContext.cs) and execute the script.

1. **Overall Description of Schemas**
2. *OLTP Schema*

The OLTP (Online Transaction Processing) schema is designed to handle the day-to-day operations of our Football Boots Store, including user management, product catalog, order processing, and customer interactions. It consists of the following tables:

* Roles
* Purpose: Stores role information for different types of users.
* Primary Key: RoleID
* Columns: RoleID (PK), RoleName (Unique)
* Users
* Purpose: Stores user account information.
* Primary Key: UserID
* Foreign Key: RoleID
* Columns: UserID (PK), Username, Password, Email, Address, Phone, RoleId (FK)
* Categories
* Purpose: Stores product category information.
* Primary Key: CategoryID
* Columns: CategoryID (PK), CategoryName
* Brands
* Purpose: Stores brand information.
* Primary Key: BrandID
* Columns: BrandID (PK), BrandName
* Products
* Purpose: Stores product details.
* Primary Key: ProductID
* Foreign Keys: CategoryID, BrandID
* Columns: ProductID (PK), ProductName, Description, Price, CategoryID (FK), BrandID (FK)
* Sizes
* Purpose: Stores size information.
* Primary Key: SizeID
* Columns: SizeID (PK), USSize, UKSize, EURSize, CMSize, BRSize, CNSize
* ProductSizes
* Purpose: Associates products with sizes and stock quantities.
* Primary Key: ProductSizeID
* Foreign Keys: ProductID, SizeID
* Columns: ProductSizeID (PK), ProductID (FK), SizeID (FK), StockQuantity
* OrderStatuses
* Purpose: Stores possible statuses for orders.
* Primary Key: StatusID
* Columns: StatusID (PK), StatusName
* Orders
* Purpose: Stores order information.
* Primary Key: OrderID
* Foreign Keys: UserID, StatusID
* Columns: OrderID (PK), UserID (FK), OrderDate, TotalAmount, StatusID (FK)
* OrderDetails
* Purpose: Stores details of individual items in orders.
* Primary Key: OrderDetailID
* Foreign Keys: OrderID, ProductSizeID
* Columns: OrderDetailID (PK), OrderID (FK), ProductSizeID (FK), Quantity, Price
* Reviews
* Purpose: Stores customer reviews for products.
* Primary Key: ReviewID
* Foreign Keys: UserID, ProductID
* Columns: ReviewID (PK), UserID (FK), ProductID (FK), Rating, Comment, ReviewDate
* ShoppingCart
* Purpose: Stores shopping cart information.
* Primary Key: CartID
* Foreign Key: UserID
* Columns: CartID (PK), UserID (FK)
* ShoppingCartItems
* Purpose: Stores items in shopping carts.
* Primary Key: CartItemID
* Foreign Keys: CartID, ProductSizeID
* Columns: CartItemID (PK), CartID (FK), ProductSizeID (FK), Quantity
* PaymentMethods
* Purpose: Stores payment method information.
* Primary Key: PaymentMethodID
* Columns: PaymentMethodID (PK), MethodName
* Payments
* Purpose: Stores payment transactions.
* Primary Key: PaymentID
* Foreign Keys: OrderID, PaymentMethodID
* Columns: PaymentID (PK), OrderID (FK), PaymentDate, Amount, PaymentMethodID (FK)

1. *OLAP Schema*

The OLAP (Online Analytical Processing) schema is designed for data analysis and reporting. It consists of dimension and fact tables optimized for querying and analysis:

* Dimension Tables:
  + DimRole
* Purpose: Stores role information for users.
* Primary Key: RoleID
* Columns: RoleID (PK), RoleName
  + DimDate
* Purpose: Stores date-related information.
* Primary Key: DateID
* Columns: DateID (PK), Date, Year, Month, Day, Week
  + DimSize
* Purpose: Stores size information.
* Primary Key: SizeID
* Columns: SizeID (PK), USSize, UKSize, EURSize, CMSize, BRSize, CNSize
  + DimCustomer
* Purpose: Stores customer information.
* Primary Key: CustomerID
* Foreign Key: RoleID
* Columns: CustomerID (PK), Username, Email, Address, Phone, RoleID (FK)
  + DimCategory
* Purpose: Stores product category information.
* Primary Key: CategoryID
* Columns: CategoryID (PK), CategoryName
  + DimBrand
* Purpose: Stores brand information.
* Primary Key: BrandID
* Columns: BrandID (PK), BrandName
  + DimOrderStatus
* Purpose: Stores order status information.
* Primary Key: StatusID
* Columns: StatusID (PK), StatusName
  + DimProduct\_SCD
* Purpose: Stores product information with slowly changing dimensions.
* Primary Key: ProductSCDID
* Foreign Keys: CategoryID, BrandID
* Columns: ProductSCDID (PK), ProductID, ProductName, Description, Price, CategoryID (FK), BrandID (FK), EffectiveDate, ExpirationDate, IsCurrent
  + DimProduct
* Purpose: Stores product information.
* Primary Key: ProductID
* Foreign Keys: CategoryID, BrandID
* Columns: ProductID (PK), ProductName, Description, Price, CategoryID (FK), BrandID (FK)
* Fact Tables:
  + FactSales
* Purpose: Stores sales transaction information.
* Primary Key: SalesID
* Foreign Keys: DateID, ProductID, SizeID, CustomerID
* Columns: SalesID (PK), DateID (FK), ProductID (FK), SizeID (FK), CustomerID (FK), Quantity, TotalAmount
  + FactInventory
* Purpose: Stores inventory information.
* Primary Key: InventoryID
* Foreign Keys: DateID, ProductID, SizeID
* Columns: InventoryID (PK), DateID (FK), ProductID (FK), SizeID (FK), StockQuantity

1. *Indexes*

Indexes are created to improve the performance of queries by providing faster access to the data in the tables:

* OLTP Indexes:
  + Users Table:
* Index: idx\_users\_email on Email column to speed up search queries based on email addresses.
  + Products Table:
* Index: idx\_products\_categoryid on CategoryID column to enhance performance for queries filtering by category.
* Index: idx\_products\_brandid on BrandID column to enhance performance for queries filtering by brand.
  + Orders Table:
* Index: idx\_orders\_userid on UserID column to optimize queries related to user orders.
* Index: idx\_orders\_statusid on StatusID column to optimize queries related to order status.
  + OrderDetails Table:
* Index: idx\_orderdetails\_orderid on OrderID column to speed up access to order details.
  + ShoppingCart Table:
* Index: idx\_shoppingcart\_userid on UserID column to optimize queries for shopping cart retrieval.
* OLAP Indexes:
  + DimCustomer Table:
* Index: idx\_dimcustomer\_email on Email column to enhance performance for queries filtering by email.
  + DimProduct Table:
* Index: idx\_dimproduct\_categoryid on CategoryID column to enhance performance for category-based queries.
* Index: idx\_dimproduct\_brandid on BrandID column to enhance performance for brand-based queries.
  + DimProduct\_SCD Table:
* Index: idx\_dimproduct\_scd\_categoryid on CategoryID column to enhance performance for category-based queries.
* Index: idx\_dimproduct\_scd\_brandid on BrandID column to enhance performance for brand-based queries.
  + FactSales Table:
* Index: idx\_factsales\_dateid on DateID column to enhance performance for date-based queries.
* Index: idx\_factsales\_productid on ProductID column to enhance performance for product-based queries.
* Index: idx\_factsales\_sizeid on SizeID column to enhance performance for size-based queries.
* Index: idx\_factsales\_customerid on CustomerID column to enhance performance for customer-based queries.
  + FactInventory Table:
* Index: idx\_factinventory\_dateid on DateID column to enhance performance for date-based queries.
* Index: idx\_factinventory\_productid on ProductID column to enhance performance for product-based queries.
* Index: idx\_factinventory\_sizeid on SizeID column to enhance performance for size-based queries.

1. *Triggers and Helping Functions*

* Password Encryption:
  + Function:
* encrypt\_password to encrypt user passwords before insert or update.
  + Triggers:
* encrypt\_password\_insert for encrypting passwords on insert.
* encrypt\_password\_update for encrypting passwords on update.
* Slowly Changing Dimension Type 2:
  + Function:
* update\_dimproduct\_scd to manage historical data for products when changes occur.
* Trigger:
* trg\_update\_dimproduct\_scd to execute the function after updates on DimProduct.

1. *Roles*

Roles are created to manage user permissions and access control:

* guest: Can only search products (read-only access to Categories, Brands, and Products tables).
* client: Can add to cart, place orders, and leave reviews (read and write access to relevant tables).
* admin: Full access to all tables, functions, and sequences in the schema.

1. **Main Functions and Procedures**

This section describes the key functions and procedures used in the database, focusing on those for administrative actions, user actions, and ETL processes.

1. *Stored Functions for Admin Actions*

Admin functions handle CRUD (Create, Read, Update, Delete) operations for various entities in the database, ensuring that only users with 'admin' roles can perform these actions. Each function raises a 'Permission Denied' exception if the current user is not 'admin'.

1. *Stored Procedures for User Actions*

User functions facilitate actions that users with 'client' roles can perform, such as managing shopping carts and placing orders. These functions also ensure appropriate permission checks.

* AddToCart
* Adds an item to a user's shopping cart.
* Parameters: uid, productSizeID, quantity.
* Checks if the current user is 'admin' or 'client'.
* UpdateCartItemQuantity
* Updates the quantity of an item in the shopping cart.
* Parameters: ciID, quantity.
* Checks if the current user is 'admin' or 'client'.
* RemoveFromCart
* Removes an item from the shopping cart.
* Parameters: ciID.
* Checks if the current user is 'admin' or 'client'.
* PlaceOrder
* Places an order for the items in the user's shopping cart.
* Parameters: uID, pmID.
* Checks if the current user is 'admin' or 'client'.
* Validates stock availability, updates stock quantities, moves items to order details, and clears the shopping cart.
* SearchProducts
* Searches for products based on a search term.
* Parameters: searchTerm.
* Returns product details matching the search criteria.
* RegisterUser
* Registers a new user with a 'client' role.
* Parameters: username, password, email, address, phone.
* Resets role and sets the new user to 'guest' after registration.
* Login
* Logs in a user and sets their role.
* Parameters: username, password.
* Returns the user's role if authentication is successful, otherwise returns 'guest'.
* Logout
* Logs out a user and sets their role to 'guest'.

1. **Data Loading and Transformation**

In this section, we detail the scripts and processes involved in loading data from CSV files and transforming it using an ETL (Extract, Transform, Load) pipeline implemented in C#.

1. *Data Loading from CSV Files*

To load data from CSV files into temporary tables in a PostgreSQL database, the script from file ‘1.4-Data-Loading.sql’ is used. This script is executed twice, once for each dataset (Dataset-1 and Dataset-2).

* Drop Temporary Tables: If temporary tables from previous runs exist, they are dropped.
* Create Temporary Tables: Temporary tables are created with the same structure as the main tables.
* Load Data from CSV: Data is copied from CSV files into the temporary tables using the COPY command.
* Insert Data into Main Tables: Data is transferred from temporary tables to the main tables, ensuring no duplicate entries are created.
* Drop Temporary Tables: Temporary tables are dropped after the data transfer is complete.

1. *ETL Process*

The ETL process in C# is responsible for extracting data from the OLTP (Online Transaction Processing) system, transforming it as necessary, and loading it into the OLAP (Online Analytical Processing) system. The ETL process leverages Entity Framework for database interactions and ensures that the data is consistently and accurately moved from the OLTP database to the OLAP database.

* Context Initialization: Two database contexts are initialized, one for OLTP and one for OLAP. The contexts weren’t created manually, instead, Scaffold-DbContext command was used.
* ETL Process Execution: The ETL process is executed by calling various methods to extract and load data for different entities such as roles, customers, categories, brands, products, sizes, inventory, and sales.
* Extract and Load Methods: Each method extracts data from the OLTP database, checks for existing records in the OLAP database to prevent duplicates, transforms the data as necessary, and then loads it into the OLAP database.

1. **PowerBI Report**

The Power BI report (file ‘2.3-Visual-Report’), titled “Sales Trends of FG and SG Football Boots Priced Above 120”, provides a comprehensive analysis of sales trends for FG and SG football boots that are priced above 120. The report is divided into several sections, each providing specific insights.

* On the left side of the report, there’s a pie chart titled “Sales Volume by ProductName”. This chart shows the distribution of sales volume across various football boot products. Each product is represented by a different color and labeled with the product name and the corresponding percentage of total sales.
* In the center top of the report, there’s a bar graph titled “Sales Volume by EurSize”. This graph shows the revenue amounts for different sizes of boots, with sizes on the horizontal axis and revenue on the vertical axis. This allows for a clear view of how revenue has fluctuated over sizes.
* Below the bar graph, there’s a line chart titled “Sales Volume by Month” (Total Cost per Month). This chart depicts cost trends over several months, again with months on the horizontal axis and cost values on the vertical axis. This visualization helps in understanding how costs have varied over time.
* On the right side of the report, there are slicers or filters for price unit (over 120), and category name (‘FG’ and ‘SG’). These filters allow users to refine the data displayed in the report based on specific criteria.

Overall, this Power BI report provides valuable insights into sales performance over time for specific products and specific sizes at certain price points. It enables users to analyze trends in volume sold, best sizes to sell, and costs incurred across different time periods. The visualizations enable quick identification of patterns or anomalies in sales data that could inform business decisions.

1. **Conclusion**

The Football Boots Store e-commerce application has been meticulously developed, covering database design, data loading, ETL processes, and report generation. With clear instructions and role-based access control, the system ensures user-friendliness and security. Leveraging C# and Entity Framework for ETL, it seamlessly transfers data between OLTP and OLAP systems. The Power BI report provides actionable insights into sales trends and product performance. Overall, this project demonstrates a comprehensive approach to database management, facilitating efficient operations and informed decision-making.