Adventure Works Bicycles, Inc.

Managing Relational and Non-Relational Data

Group TP2-G

Raysa Rocha (20232051), Baran Can Çelik (20232067),
Priyá Dessai (20232053) and Carlos Lourenço (20232020).

NOVA IMS Information Management School

May, 2024.

Table of Contents

1 Introduction

Adventure Works Bicycles is a renowned fictional company specialized in the manufacture of bicycles, components, accessories and clothing. With a diverse product line comprised of 97 bicycle brands across three categories – mountain, road and touring – Adventure Works serves a global customer base.

Despite its presence in online retail and its extensive network of more than 700 stores worldwide (United States, Canada, Australia, United Kingdom, France and Germany), the company faces challenges in effectively managing inventory liquidation, especially during the launch of new products. To solve this problem, Adventure Works is pioneering an innovative online auction system to expedite inventory clearance and promote product innovation.

Simultaneously, Adventure Works is preparing to expand its reach with physical stores, marking a strategic shift in customer engagement. However, opening physical stores requires careful consideration to avoid direct competition with existing retail partners. Adventure Works aims to identify ideal store locations, preserving valuable partnerships, ensuring sustainable expansion and market growth.

2 Business Problem 1 - Stock Clearance

The objective of this project is to develop a solution to solve the recurring problem of excess inventory that prevents the successful introduction of new bicycle models. Previous attempts, including aggressive discount campaigns, have fallen short of resolving the problem. In response, the leadership team has decided to implement a new approach— a comprehensive online auction covering products expected to be replaced by new models within the last two weeks of November. To support this initiative, the Adventure Works database schema must be extended to accommodate the new auction features seamlessly.

The online auction will include all products for which a new model is anticipated, with initial bid prices ranging from 50% to 75% of the listed price. The auction will take place during the last two weeks of November of 2014, coinciding with Black Friday, requiring measures to ensure website reliability under high workload.

3 Functional Specifications for Stock Clearance

3.1 Schema Overview

Adventure Works Bicycles' database schema was extended to accommodate the new auction features. The schema includes two new primary tables: Auctions.Products and Auctions.BidsOffers.

- Auctions.Products: This table stores information about products eligible for auction. It includes columns for ProductID, ExpireDate, InitialBidPrice, and AuctionStatus. The ProductID serves as the primary key, linking to the main product data in the Production.Product table.
- Auctions.BidsOffers: This table records bids placed by customers for auctioned products. It includes columns for ProductID, CustomerID, BidAmount, and BidTime. The ProductID and CustomerID columns are foreign keys referencing the main product and customer data, respectively.

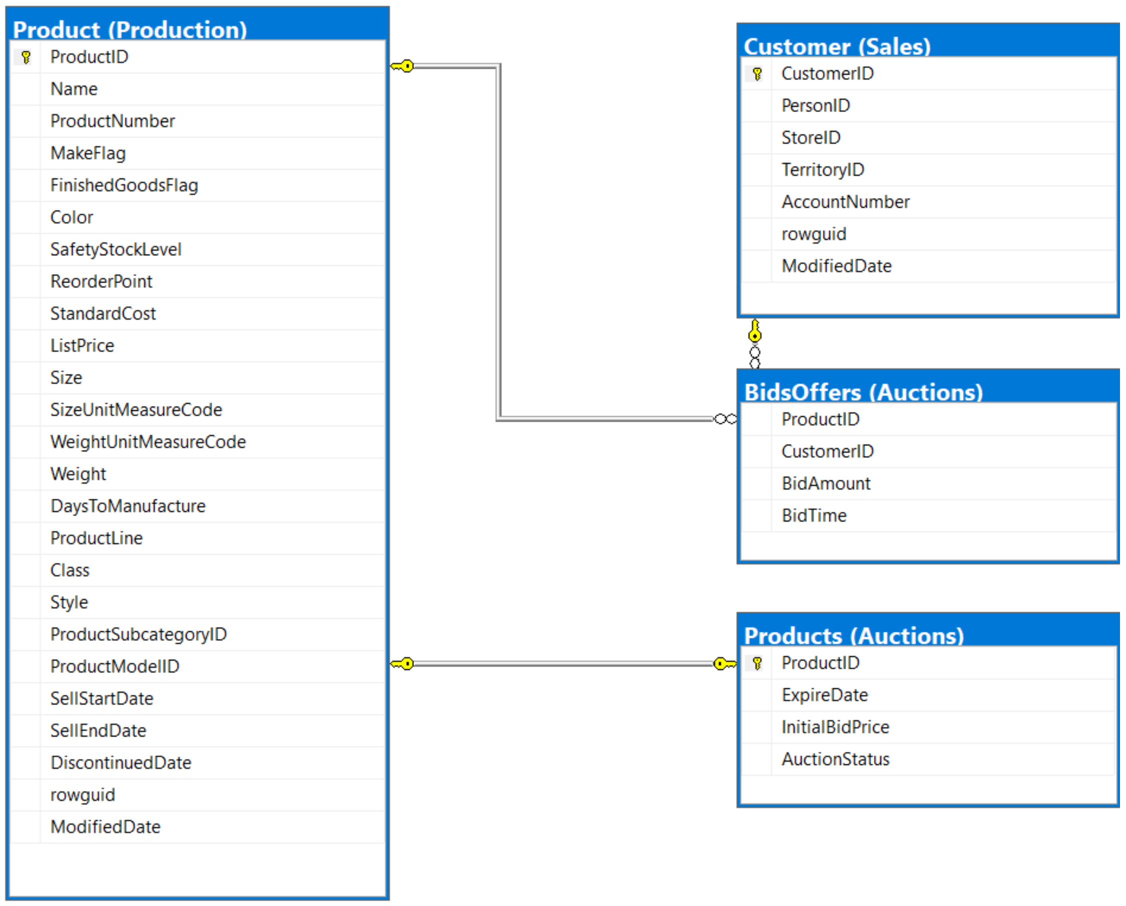

Figure 1 – Auctions Schema

3.2 Product Eligibility

- Only products currently commercialized (with no SellEndDate or DiscontinuedDate) are eligible.
- Initial bid prices for products not manufactured in-house (MakeFlag = 0) should be at least 75% of the listed price.
- For all other products, initial bid prices should be at least 50% of the listed price.

3.3 Bid Increment and Limit

- By default, users can increase bids by a minimum of 5 cents.
- The maximum bid limit is equal to the listed price.
- These thresholds are configurable within a table to accommodate changes without altering the database schema.

3.4 Stored Procedures

3.4.1 Auctions.uspAddProductToAuction

Parameters:

- @ProductID [int]: The ID of the product to be added to the auction.
- @ExpireDate [datetime]: (Optional) The auction expiration date for the Product. If not provided, the auction will end in one week.
- @InitialBidPrice [money]: (Optional) The initial bid price for the product. If not provided, it will be calculated based on predefined rules.

Functionality:

- Calculate default values for @ExpireDate and @InitialBidPrice if not provided.
- Check the data in AdventureWorks Production.Product table.
- Check the validity of the parameters and its conditions.
- Insert the product into the new table: Auctions.Products.
- Rollback transaction in case of any error.

Assumptions:

- To facilitate planning and customer engagement, we assume that products can be added to the auction two days prior to bidding (November 14th, 2014) to allow customers to preview items and strategize their bids effectively with the intention of auctioning off all products.
- Since the Adventure Works data used is from 2014, we will treat the current year as 2014 for our calculations.
- Given the challenge of simulating dates in SQL and to ensure that products can be added to the auction, we chose to set the current date as November 14th, 2014. In order to validate the code with different dates, it would be necessary to manipulate the current date.
- Since the current date was fixed, we decided to include the current actual time to distinguish the different times when products were added to the auction.

Validations:

- Products Eligibility - ensure products eligible for auction are currently commercialized:
    - ProductID cannot be null and must exist in Production.Product table.
    - SellEndDate and DiscontinuedDate must be null.
    - ListPrice cannot be null and must be higher than zero.
- Expire Date:
    - If @ExpireDate is not provided, calculate it as the current date (November 14, 2014) plus one week (November 21, 2014).
    - If @ExpireDate is provided, ensure it falls within a valid range (November 14 to November 30, 2014).
- Initial Bid Price:
    - If @InitialBidPrice is not provided, calculate it based on predefined conditions: 50% of ListPrice for in-house manufactured products, 75% for others.
    - If @InitialBidPrice is provided ensure it is:

        - Equal or higher than the predefined conditions (50% / 75% of ListPrice); and
        - Lower than ListPrice.

3.4.2 Auctions.uspTryBidProduct

Parameters:

- @ProductID [int]: The ID of the product being bid on.
- @CustomerID [int]: The ID of the customer placing the bid.
- @BidAmount [money]: (Optional) The amount of the bid. If not provided, it will be calculated based on predefined rules.

Functionality:

- Check the validity of the parameters and its conditions.
- Check the data in AdventureWorks Sales.Customer table (CustomerID column) and in Auction.Products table.
- Insert the bid into the new table: Auctions.BidsOffers.
- Rollback transaction in case of any error.

Assumptions:

- Given the challenge of simulating dates in SQL and to ensure that products can be bid on, we chose to set the current date as November 16th, 2014 (as the campaign takes place during the last two weeks of November including Black Friday). In order to validate the code with different dates, it would be necessary to manipulate the current date.
- Since the current date was fixed, we decided to include the current actual time to distinguish the different times when products were bid.

Validations:

- Product and Customer Existence - check if the product and customer exist in their respective tables:
    - ProductID and CustomerID cannot be null.
    - ProductID must exist in Auctions.Products table.
    - CustomerID must exist in Sales.Customer table.
- Auction Status - ensure the product is currently active in the auction and not cancelled:
    - If @AuctionStatus is “Cancelled”, no more bids are accepted.
- Bid Timing:
    - Allow bidding only between November 16th and November 30th, 2014.
    - Prevent bidding after the @ExpireDate (auction's expiration date).
- Bid Amount:
    - If @BidAmount is not provided, it calculates the bid amount as the last bid amount (@LastBidAmount) plus 5 cents.
    - If @BidAmount is provided, it checks:
        - whether it meets the minimum bid requirement. The minimum bid should be at least 5 cents higher than @LastBidAmount or the @InitialBidPrice.

        - if the @BidAmount has reached the @MaxBid (maximum bid limit).

3.4.3 *Auctions.uspRemoveProductFromAuction*

Parameters:

- @ProductID [int]: The ID of the product to be removed from the auction.

Functionality:

- Validate the parameters and existence of the product in the auction.
- Set the auction status of the product to 'Cancelled' in Auctions.Products table.
- Rollback transaction in case of any error.

Assumptions:

- None.

Validations:

- Product Existence - ensure the product exists in the Auctions.Products table:
    - ProductID must exist in Auctions.Products table.
- Auction Status - verify that the product is not already cancelled from the auction:
    - ProductID must have “Active” status in Auctions.Products table to be cancelled.

3.4.4 *Auctions.uspListBidsOffersHistory*

Parameters:

- @CustomerID [int]: The ID of the customer whose bid/offers history is being retrieved.
- @StartTime [datetime]: The start time for filtering bids history.
- @EndTime [datetime]: The end time for filtering bids history.
- @Active [bit]: (Optional) Flag to specify whether to include only active bids. Defaults to true.

Functionality:

- Returns bid history for the specified customer within the given time range (using and inner join between Auctions.Products and Auctions.BidsOffers tables and returning the columns: ProductID, CustomerID, BidAmount, BidTime and AuctionStatus).
- Optionally filter by active bids.
- Rollback transaction in case of any error.

Assumptions:

- For @Active parameter: 1 is True and 0 is False.

Validations:

- Customer Existence - ensure the CustomerID exists in Auctions.BidsOffers table.
- Time Range - validate that @StartTime is earlier than @EndTime.
- Active Bids:
    - if @Active is set to true (1), it checks whether the bids being retrieved are active.

  - if @Active is set to false (0), the procedure retrieves all bids regardless of the auction status (active or cancelled).

3.4.5 Auctions.uspUpdateProductAuctionStatus

Parameters:

- None

Functionality:

- Sets the current date and time.
- Updates the auction status of products where the auction has expired but not reached the maximum bid.
- Returns details of the last bid for each product (using and inner join between Auctions.Products and Auctions.BidsOffers tables and returning the columns: ProductID, ExpireDate, InitialBidPrice, CustomerID, BidAmount, BidTime and AuctionStatus).

Assumptions:

- For the procedure to work properly based on the previously established assumptions, it was necessary to set the current date to November 16, 2014, in order to guarantee consistency in the calculations.
- To validate the code with different dates and transition the product status from “Active” to “Sold” (if the expiration date is reached), it is essential to manipulate the current date.

Validations:

- Products with maximum bids:
  - Select the maximum bid amount for each Product (with values lower than ListPrice) from Auctions.BidsOffers table.
- Auction Status:
  - Consider products (from Auctions.Products table) with “Active” status, to ensure that auctions are ongoing.
  - Exclude products with “Cancelled” status from the update process.
- Expired Auctions:
  - Assumes a product's auction has expired if its expiration date is less than or equal to the current date and time.
- Update Auction Status:
  - If the conditions are met, sets the AuctionStatus of the products identified as “Sold”.

4 Business Problem 2 - Brick and Mortar Stores

Adventure Works is about to enter the world of traditional retail, planning to establish its first physical stores in the United States. However, this expansion presents a delicate balance: while aiming to improve direct customer interactions and brand presence, the company must avoid direct competition with existing retail partners. Strategically addressing this challenge involves identifying ideal store locations that maximize market reach while safeguarding relationships with key partners. Through data-driven insights and market analysis, Adventure Works seeks to pave the way for sustainable growth and successful expansion in physical retail.

5 Functional Specifications for Brick-and-Mortar Stores

5.1 Schema Overview

The schema below provides essential data for determining ideal locations for brick-and-mortar store expansion.

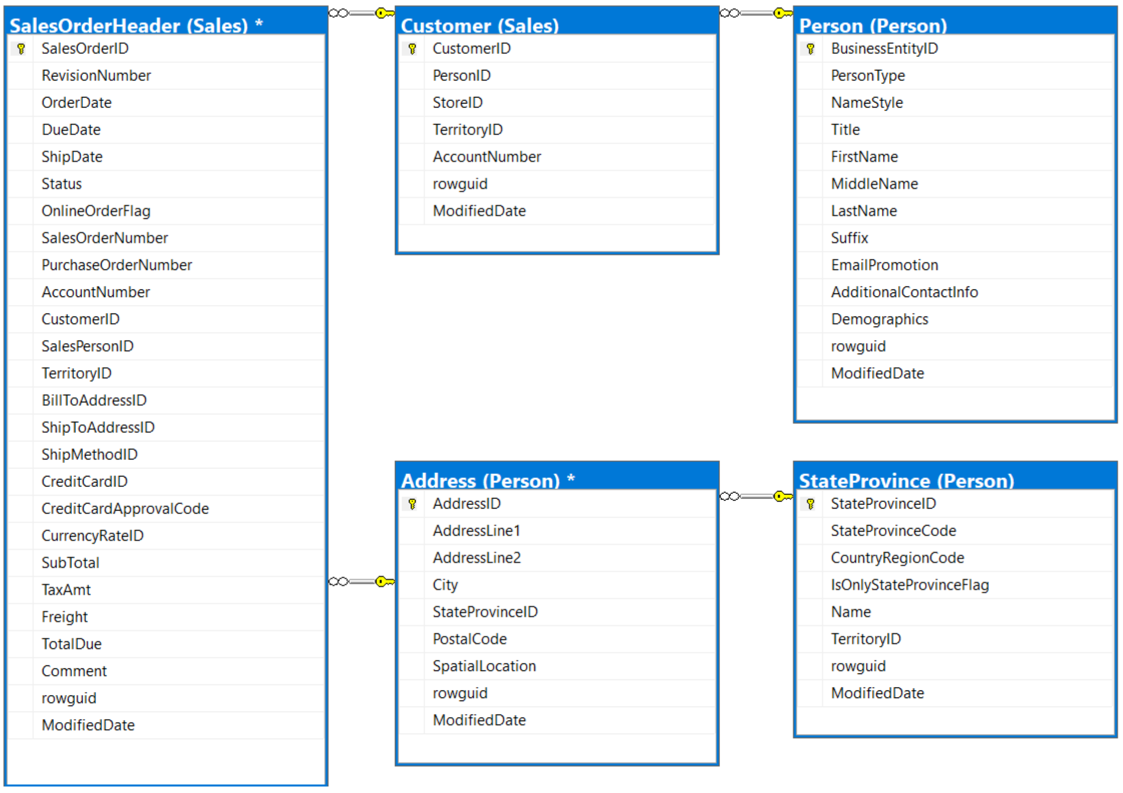

Figure 2 – Brick-and-Mortar Stores Schema.

5.2 Analyzing the SQL Query

5.2.1 SQL Query

```sql
SELECT TOP(5)
    SUM(SOH.TotalDue) AS order_value,
    a.City
FROM
    sales.SalesOrderHeader AS SOH
LEFT JOIN
    person.Address AS a ON SOH.ShipToAddressID = a.AddressID
LEFT JOIN
    person.StateProvince AS s ON a.StateProvinceID = s.StateProvinceID
LEFT JOIN
    sales.Customer AS c ON c.CustomerID = soh.CustomerID
LEFT JOIN
    person.Person AS p ON c.PersonID = p.BusinessEntityID
WHERE
    p.PersonType = 'IN'
    AND s.CountryRegionCode = 'US'
    AND NOT a.City IN (
        SELECT TOP(30)
            a.City
        FROM
            sales.SalesOrderHeader AS SOH
        LEFT JOIN
            person.Address AS a ON SOH.ShipToAddressID = a.AddressID
        LEFT JOIN
            person.StateProvince AS s ON a.StateProvinceID = s.StateProvinceID
        LEFT JOIN
            sales.Customer AS c ON c.CustomerID = soh.CustomerID
        LEFT JOIN
            person.Person AS p ON c.PersonID = p.BusinessEntityID
        WHERE
            p.PersonType = 'SC'
            AND s.CountryRegionCode = 'US'
        GROUP BY
            soh.CustomerID, a.City
        ORDER BY
            SUM(TotalDue) DESC
    )
GROUP BY
    a.City
ORDER BY
    order_value DESC
```

5.2.2 Interpreting Query Results and Business Implications

1. **Calculation of Order Value:** Calculating the total order value (order_value) for each city by using the SUM function in the TotalDue column of Sales.SalesOrderHeader table, it will aggregate order values, allowing insight into customer spending patterns across different locations.
2. **Table Joins:** LEFT JOIN operations were performed to aggregate relevant data from different tables:
   - Person.Address (aliased as a): Contains address information.
   - Person.StateProvince (aliased as s): Holds state and province details.
   - Sales.Customer (aliased as c): Relates to customer data.
   - Person.Person (aliased as p): Contains personal information.

   By merging these tables, the query builds a unified dataset encompassing essential details from each source.
3. **Filtering Results:** The WHERE clause applies specific conditions to filter the results:
   - persontype ='in': Ensures inclusion of only records pertaining to individual customers ('in').
   - s.CountryRegionCode = 'US': Restricts results to records from the United States.
   - NOT a.City IN (...): Excludes cities identified in a subquery, aligning with the project's directive to avoid competition with top retail partners.
4. **Subquery Exclusion:** A subquery is used to identify and exclude the top 30 cities by highest total order value. Grouping results by soh.CustomerID, a.City and ordering by the sum of TotalDue in descending order, the main query omits these cities from consideration.
5. **Grouping and Ordering:** The outer query groups results by city (a.City). Finally, the results are ordered by order_value (the calculated sum of TotalDue) in descending order, facilitating easy identification of cities with the highest customer spending.

In our quest to identify ideal locations for Adventure Works' first physical stores, our analysis focused on three critical factors: location, customer spending and shopper type. Using SQL queries, we identified **Bellflower and Burbank** as ideal locations for Adventure Works physical stores.

6 Conclusion

Adventure Works Bicycles, Inc. has taken proactive steps to overcome two significant challenges impacting its business operations. Firstly, by implementing a comprehensive online auction system, the company effectively solved the persistent problem of excess inventory during the launch of new bicycle models. This strategic approach not only optimizes stock management processes but also facilitates the liquidation of existing stocks, thus creating space for the introduction of new product lines.

Furthermore, Adventure Works' expansion into brick-and-mortar retail represents a significant diversification of its sales channels and a strategic move to enhance its market presence. Analyzing data-driven insights, the company identified **Bellflower and Burbank** as prime locations for its inaugural stores, strategically positioning itself to capitalize on significant order values and customer demographics while mitigating competitive concerns.

In conclusion, Adventure Works Bicycles' adoption of an online auction system and establishment of brick-and-mortar stores in Bellflower and Burbank increases strategic investments to improve inventory management and expand market presence.