University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: 13/11/2024

Group Number: 46

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Andrew Chen	42072777	x6b9i	Iphoneotto3@gmail.com
Helen Ma	44776268	n0j9u	helenjym@gmail.com
Stephanie Cao	38000618	y2m7b	stephmimi27@gmail.com

Project Description

This project is designed to create a database system for managing **warehouse operations** and **supply chain logistics**. The system focuses on efficiently tracking inventory levels, maintaining supplier relationships, and managing the process of restocking products when stock levels fall below a certain threshold. The goal is, therefore, to ensure that products are available without overstocking, enabling warehouse managers or business owners to make informed decisions about when and how much to restock.

Database Specifications

The database will allow users (like business owners or warehouse managers) to:

- Track inventory levels and determine when to restock.
- Manage relationships with suppliers, including placing and tracking orders.
- View product details and restock history.

The database will also store details about shipments, payments, and inventory availability in multiple warehouses, providing a comprehensive solution for supply chain management.

Accomplishments

The warehouse management system achieved the following:

- **Supplier Management**: enabled users to create, update, and manage supplier information.
- **Product Management**: the system allows users to retrieve, update and create products.
- **Advanced Data Queries**: users can make complex data retrieval (e.g., joins, aggregations, and nested queries) to analyze product, supplier.

Application Platform

We will use the department-provided **Oracle** for our database and **HTML/CSS & JavaScript** to implement the application in our technology stack.

Schema Change

The final schema followed the initial design, with minor adjustments to attributes' name and domain, and cardinality constraints.

CHANGE

- Adjusted column types for PostalCode in:

Supplier	VARCHAR(10) → CHAR(6)
SupplierProvince	VARCHAR(10) → CHAR(6)
SupplierCity	VARCHAR(10) → CHAR(6)
Warehouse	VARCHAR(10) → CHAR(6)
WarehouseProvince	VARCHAR(10) → CHAR(6)
WarehouseCity	VARCHAR(10) → CHAR(6)

- Changed column name for MinimumOrderQuantity in:

Product	MinimumOrderQuantity → MinimumOrder

Final Schema

Warehouse(<u>WID</u>: char(10), Capacity: int, Address: varchar(255), **PostalCode**: char(6))
PK: (WID)
FK: (PostalCode references WarehouseProvince PostalCode)

WarehouseProvince(<u>**PostalCode**</u>: char(6), Province: varchar(255))
PK: (PostalCode)
FK: (PostalCode references WarehouseCity PostalCode)

WarehouseCity(<u>PostalCode</u>: char(6), City: varchar(255))
PK: (PostalCode)

Employee(<u>EID</u>: char(10), Name: varchar(255), Salary: decimal(10, 2), **WID**: char(10))
PK: (EID)
FK: (WID references Warehouse ID)
NOT NULL: (Name, WID)

WareHouseManager(<u>**EID**</u>: char(10), TeamSize: int)
PK: (EID)
FK: (EID references Employee ID)

InventoryManager(<u>**EID**</u>: char(10), DeliveryVehicle: varchar(255))
PK: (EID)
FK: (EID references Employee ID)

DeliveryStaff(<u>**EID**</u>: char(10), Specialization: varchar(255))
PK: (EID)
FK: (EID references Employee ID)

Inventory (<u>IID</u>: char(10), QuantityAvailable: int, ReorderLevel: int, **WID**: char(10), **PID**: char(10))
PK: (IID)
FK: (WID references Warehouse ID)
(PID references Product ID)
CK: (PID)
NOT NULL: (QuantityAvailable, ReorderLevel, WID, PID)
UNIQUE: (PID)

Product(<u>PID</u>: char(10), **Description**: varchar(255), Name: varchar(255), UnitPrice: decimal(10, 2), MinimumOrder: int, **SID**: char(10))
PK: (PID)
FK: (SID references Supplier ID)

Supplier(<u>SID</u>: char(10), Address: varchar(255), **PostalCode**: char(6))
PK: (SID)
FK: (PostalCode references SupplierProvince PostalCode)

SupplierProvince (<u>**PostalCode**</u>: char(6), Province: varchar(255))
PK: (PostalCode)
FK: (PostalCode references SupplierCity PostalCode)

SupplierCity(<u>PostalCode</u>: char(6), City: varchar(255))
PK: (PostalCode)

RestockOrder(<u>ROID</u>: char(10), Status: ENUM, OrderDate: date, TotalCost: decimal(10, 2), **SID**: char(10), **WID**: char(10))
PK: (ROID)
FK: (WID references Warehouse ID)
(SID references Supplier ID)
NOT NULL: (Status, SID, WID)
ENUM('Pending', 'Confirmed', 'Processing', 'Completed', 'Returned')
ASSERTION & TRIGGER: (Every RestockOrder → OrderLine)

Shipment(<u>ShipID</u>: char(10), Status: ENUM, ShipmentDate: date, ExpectedDeliveryDate: date, **ROID**: char(10))
PK: (ShipID)
FK: (ROID references RestockOrder ID)
CK: (RIOD)
NOT NULL: (Status, ROID)
UNIQUE: (ROID)
ENUM('Shipped', 'In Transit', 'Delayed', 'Delivered')

Payment(<u>PayID</u>: char(10), Status: ENUM, Date: date, PaymentMethod: varchar(50), AmountPaid: decimal(10, 2), **ROID**: char(10))
PK: (PayID)
FK: (ROID references RestockOrder ID)
CK: (RIOD)
NOT NULL: (Status, ROID)

UNIQUE: (ROID)
ENUM('Pending`, 'Processing', 'Completed', 'Failed', 'Cancelled', 'Refunded')

OrderLine(<u>OID</u>: char(10), QuantityOrder: int, **PID**: char(10), **<u>ROID</u>**: char(10))
PK: (ROID, OID)
FK: (ROID references RestockOrder ID)
(PID references Product ID)
NOT NULL: (QuantityOrder, PID)

SQL Queries

Insert

Location: src/services/productService.js: lines 4-5

SQL: INSERT INTO Product (PID, Name, Description, UnitPrice, MinimumOrder, SID)
VALUES (:PID, :Name, :Description, :UnitPrice, :MinimumOrder, :SID);

Update

Location: src/services/productService.js: lines 14-20

```
SQL: UPDATE Product SET Name = :Name,
                        Description = :Description,
                         UnitPrice = :UnitPrice,
                         MinimumOrder = :MinimumOrder,
                         SID = :SID
                         WHERE PID = :PID;
```

Delete

Location: src/services/supplierService.js: line 28

SQL: DELETE FROM Supplier WHERE SID = :SID;

Selection

Location: src/services/productService.js: line 64

SQL:

```
(Generic)           SELECT * FROM Product WHERE ${query};
(Case dependent)    SELECT * FROM Product WHERE PID = 1111111111 OR
                    PID = 0000000005 AND
                    MinimumOrder = 1;
```

Projection

Location: src/services/productService.js: line 54

SQL:

(Generic) SELECT ${attributes} FROM Product;

(Case dependent) SELECT PID, MinimumOrder, Description, SID FROM Product;

Join

Location: src/ services/analyticsService.js: lines 7-10

SQL: SELECT p.PID, p.Name, p.Description, p.UnitPrice, p.MinimumOrder, p.SID, c.city
    FROM Product p, Supplier s, SupplierCity c
    WHERE c.city = :city AND s.PostalCode = c.PostalCode AND s.SID = p.SID`;

Aggregation (GROUP BY)

Description: Count number of products supplied by each supplier.

Location: src/services/analyticsService.js: lines 27-29

SQL: SELECT SID AS SupplierID, COUNT(*) AS NumberOfProducts
    FROM Product
    GROUP BY SID`;

Aggregation (HAVING)

Description: Find supplier whose product count matches the given amount.

Location: src/services/analyticsService.js: line 47-52

SQL: SELECT s.SID AS SupplierID, COUNT(*) AS NumberOfProducts
    FROM Product p,
        Supplier s
    WHERE s.SID = p.SID
    GROUP BY s.SID
    HAVING COUNT(*) = :productCount`;

Nested Aggregation

Description: Find the supplier(s) with highest average product price.

Location: src/services/analyticsService.js: lines 59-67

SQL:
```
SELECT s.SID AS SupplierID, AVG(p.UnitPrice) AS AvgUnitPrice
    FROM Supplier s, Product p
    WHERE s.SID = p.SID AND p.UnitPrice IS NOT NULL
    GROUP BY s.SID
    HAVING AVG(p.UnitPrice) >= ALL (SELECT AVG(p2.UnitPrice)
                                    FROM Product p2
                                    WHERE p2.UnitPrice IS NOT NULL
                                    GROUP BY p2.SID);
```

Division

Description: Find the warehouse(s) that monitors all inventories.

Location: src/services/analyticsService.js: lines 74-81

SQL:
```
SELECT w.WID AS WarehouseID
FROM Warehouse w
WHERE NOT EXISTS (SELECT i.IID
                  FROM Inventory i
                       WHERE NOT EXISTS (SELECT ip.PID
                                         FROM Inventory ip
                                         WHERE ip.WID = w.WID
                                              AND ip.IID = i.IID)));
```