**CSC540 Database Management Systems**

**Project Report 1**

**WolfWR, WolfCity wholesale store chain**

**Assumptions:**

1. A staff works for at most one store at a given point of time.
2. There is a single warehouse handling the inventory within multiple stores.
3. A person can sign up for as a club member only at a store.
4. A club member can return only a damaged/defective product. Thus, we do not add the returned product back to the inventory.
5. Only two membership levels exist, gold and platinum.
6. Both gold and platinum memberships last for a year after which the membership has to be renewed.
7. One checkout by the customer at a store is considered as one transaction.
8. Each transaction will have exactly one staff (cashier) associated with it.
9. The cashback amount for platinum memberships will be rewarded in the first transaction of the next year.

**Problem Statement**

We have designed a WolfWR wholesale-store Database System for WolfCity. The staff at WolfWR will be using this database system to maintain staff information, club-member information, supplier information, store information and merchandise information.

The four major tasks that will be performed include: information processing, maintaining inventory records, maintaining billing & transaction records and generating reports regarding total sales, customer growth, & merchandise stock. All the transactions made by club members will be recorded and maintained. Staff will be able to record and maintain information pertaining to new shipments received at the warehouse, register new club members, handle product transfers and returns between stores and customers.

In order to record and maintain the various operations taking place in the WolfWR wholesale-store chain, a database is required to collect and store the data generated in an efficient manner. There may be instances where multiple transactions take place at the same time. For instance, when multiple customers are signing up or purchasing commodities at the same time or when multiple stores are transferring products to one another at the same time. When these actions happen concurrently, there could arise some inefficiency if the data is stored in a system of files. A database would also help the store staff in taking data driven decisions regarding merchandise stock and also generate insightful reports on inventory and customer behaviour.

**Intended Users:**

* **Administrative Staff (Admins):** Admins have access to all information pertaining to the wholesale-store chain. These are the users responsible for adding, editing, or deleting staff, customers, suppliers and can view the personal information of the staff members. They have complete control of the WolfWR wholesale-store chain Management System.
* **Registration Staff:** The Registration Staff handles registering new customers and cancelling membership subscriptions of existing customers. They can view customer information such as their personal details, membership level and membership expiration date.
* **Warehouse Staff:** The Warehouse Staff handles checking in new shipments in the warehouse, product returns from stores to warehouse, and monitors the transfer of products between stores.
* **Billing Staff:** The Billing Staff generates bills for suppliers and handles cashback reward checks to platinum members. They can also view transaction related information and generate reports.
* **Cashier Staff:** The Cashier Staff is responsible for scanning transactions at stores. They can view merchandise and store information that can help in case a product is throwing errors while scanning or when a product is not showing expected discounts. They can also view customer information to check for specific membership discounts.
* **Manager Staff:** The Manager Staff has access to all the information of a particular store. The manager can view staff information of the particular store he/she manages, merchandise stock information, customer information and transaction information related to the store.

**Main Entities:**

1. **Staff Information**: Staff ID, Name, Date of Birth, Job Title, Home Address, Email Address, Employee Start Date, Employee End Date, Phone Number
2. **Club Member Information**: Customer ID, First Name, Last Name, Home Address, Email Address, Phone Number, Membership Expiration Date, Active Status
3. **Supplier Information**: Supplier ID, Supplier Name, Phone Number, Email Address, Location
4. **Merchandise Information**: Product ID, Product Name, Production Date, Market Price, Buy Price, Expiration Date, Total Quantity
5. **Store Information**: Store ID, Address, Phone Number

**Application Program Interfaces:**

**Information Processing:**

**enterStaffInfo**(StaffID, StoreID, Name, DateOfBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, PhoneNumber)

return confirmation

**updateStaffInfo**(StaffID, StoreID, Name, DateOfBirth, PhoneNumber, JobTitle, EmpEndDate, EmailAdd, HomeAddress)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**deleteStaffInfo**(StaffID)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**enterStoreInfo**(StoreID, Address, PhoneNumber)

return confirmation

**updateStoreInfo**(StoreID, PhoneNumber, Address)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**deleteStoreInfo**(StoreID)

return confirmation

* If NULL value for the field, then they will not be updated.

**enterCustomerAndSignupInfo**(CustomerID, StaffID, StoreID, SignUpDate, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

return confirmation

**updateCustomerAndSignupInfo**(CustomerID, StaffID, StoreID, SignUpDate, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**deleteCustomerAndSignupInfo**(CustomerID, StaffID, StoreID)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**enterSupplierInfo**(SupplierID, SupplierName, PhoneNumber, EmailAdd, Location)

return confirmation

**updateSupplierInfo**(SupplierID, SupplierName, PhoneNumber, EmailAdd, Location)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**deleteSupplierInfo**(SupplierID, SupplierName, PhoneNumber, EmailAdd, Location)

return confirmation

* If NULL value for any of the fields, then they will not be updated.

**enterProductDiscount**(StoreID, ProductID, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice)

return confirmation

**Maintaining inventory records:**

**addNewShipments**(ProductID, StaffID (Warehouse Staff), ProductName, ProductionDate, ExpirationDate, MarketPrice, BuyPrice, TotalQuantity)

returns true if added, and false otherwise.

**updateStoreReturns**(StoreID, ProductID, returnQuantity)

returns true if updated, and false otherwise.

* If NULL value for any of the fields, then they will not be updated.

**transferProduct**(ProductID, StaffID (Warehouse Staff), ProductSource, ProductDest, transferQuantity)

return true if updated, and false otherwise.

* If NULL value for any of the fields, then they will not be updated.

**Maintaining billing and transaction records:**

**generateSupplierBills**(SupplierID, StaffID (Billing Staff), ProductID, BuyPrice)

return amount and update it in the Suppliers table.

* If NULL value for any of the fields, then they will not be updated.

**generateRewardChecks**(StaffID (Billing Staff), CustomerID, transactionID)

return cashback and update it in the Customer table.

* If NULL value for any of the fields, then they will not be updated.

**checkProductDiscount**(StoreID, Staff ID (Billing Staff), ProductID)

return discount information, sale start date and sale end date.

* If NULL for any of the fields, then function would return NULL.

**calculateTotalPriceofTransaction**(TransactionID, StaffID (Billing Staff), CustomerID, StoreID, ProductID)

return discounted total amount and update it in the Transaction table.

* If NULL value for any of the fields, then they will not be updated.

**Reports:**

**getTotalSales**(transactionID, startDay, endDay, startMonth, endMonth, startYear, endYear)

return list of transactions with total sales amount.

**getStoreSalesGrowth**(transactionID, storeID, startDay, endDay, startMonth, endMonth, startYear, endYear)

return list of transactions with total sales amount.

**getMerchandiseStockStore**(StoreID, productID, storeQuantity)

return list of products available in a store with their store quantity.

**getMerchandiseStockProduct**(ProductID, storeQuantity)

return list of products available throughout the stores with their store quantity.

**getCustomerGrowth**(CustomerID, startDay, endDay, startMonth, endMonth, startYear, endYear)

return count of all active customers from start date to end date.

**getCustomerActivity**(CustomerID, startDay, endDay, startMonth, endMonth, startYear, endYear)

return customer transaction amounts from start date to end date.

**Tasks and Operations – Realistic Situations:**

**Situation 1:** The admin wants to know the merchandise stock report generated for the current month in store A. She/He then realizes that there is an excess of Product B. A product transfer/return is requested from store A to the warehouse.

**Situation 2:** An existing club member named Henry, enters one of the wholesale stores. He realizes that his new phone number has not been updated in the system. The registration office staff updates his new phone number in the system.

**Situation 3:** A platinum member named Emma is billing her products in the billing counter. Once the bill is generated, she realizes that the discount was not applied for certain products and that she did not receive a 2% cashback reward for the previous year. The billing staff updates the sale information of each product in the store and sends a reward check based on her membership level.

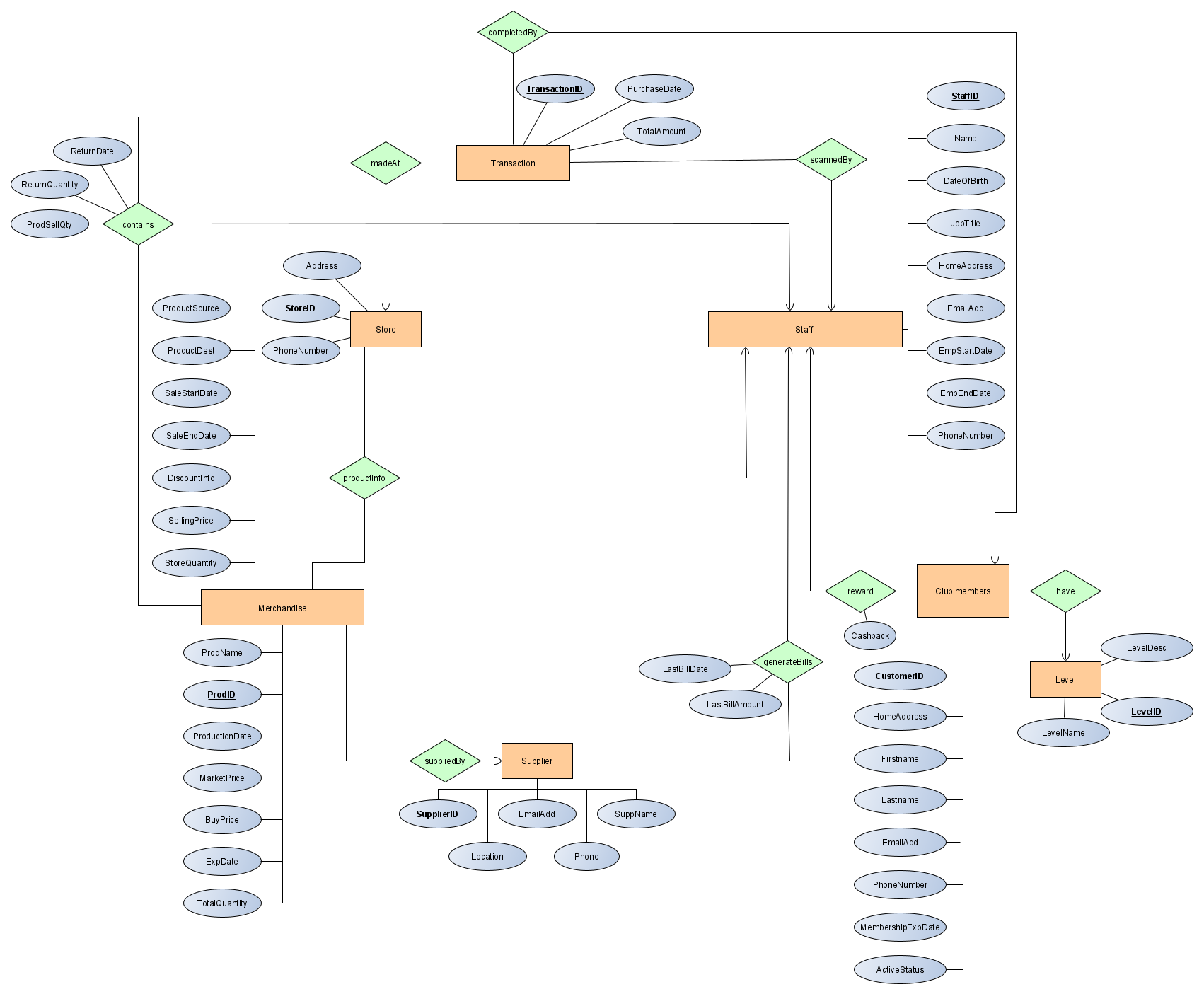
**Description of Views:**

* Billing staff: Billing staff can generate bills that need to be paid to suppliers. They can also handle yearly rewards for platinum members and send out reward checks. All transaction and billing related reports can be generated by the billing staff.
* Registration staff: Operators in the registration office can sign up new customers and cancel memberships. Need not to be aware of supplier and warehouse information.
* Warehouse staff: Operators in the warehouse office can add in new shipments received, transfer products between stores, and handle returns. This staff need not be aware of customers info and store level information.
* Administrators: Admins will have access to everything in the database. She/he can view all Staff information, look at all the Transactions taking place at each store, Signup customers, and update Supplier information.
* Manager staff: Managers have admin rights for the store they are managing. They can view staff working in that store, transaction summary of the store, product stock of the store and signup customers in that store.
* Cashier Staff: Cashiers also have the same rights as manager for the particular store they are working in except signing up new customers. They can view merchandise information for the store to check on product discounts, transaction summary of the store and can see other cashier staff information.

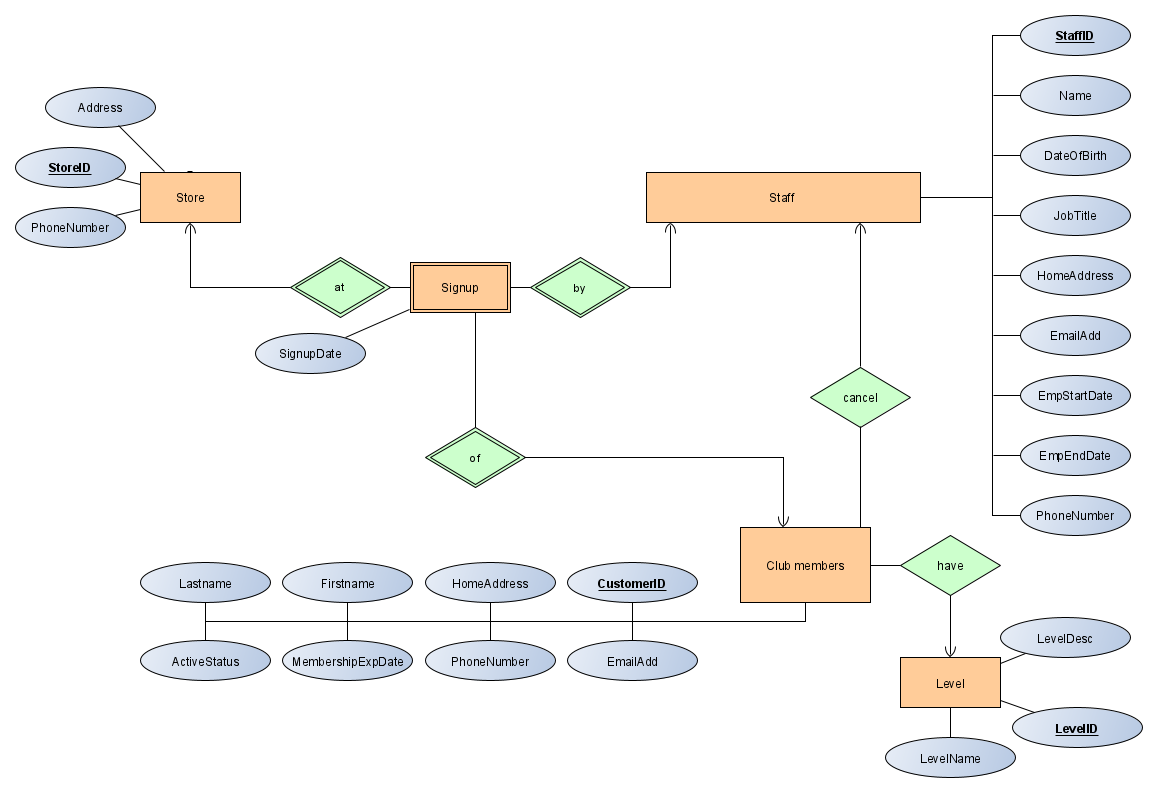
**Local E/R Diagrams:**

**Admin’s View:**

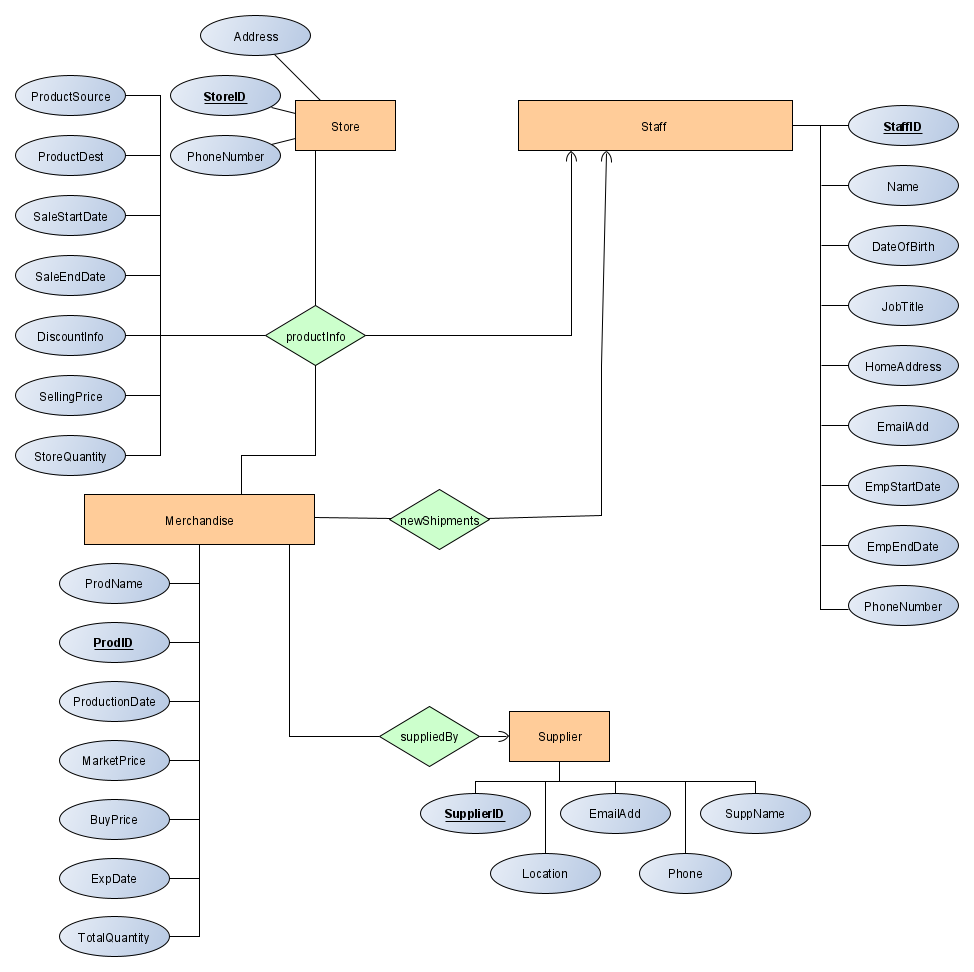
**Billing Staff view:**



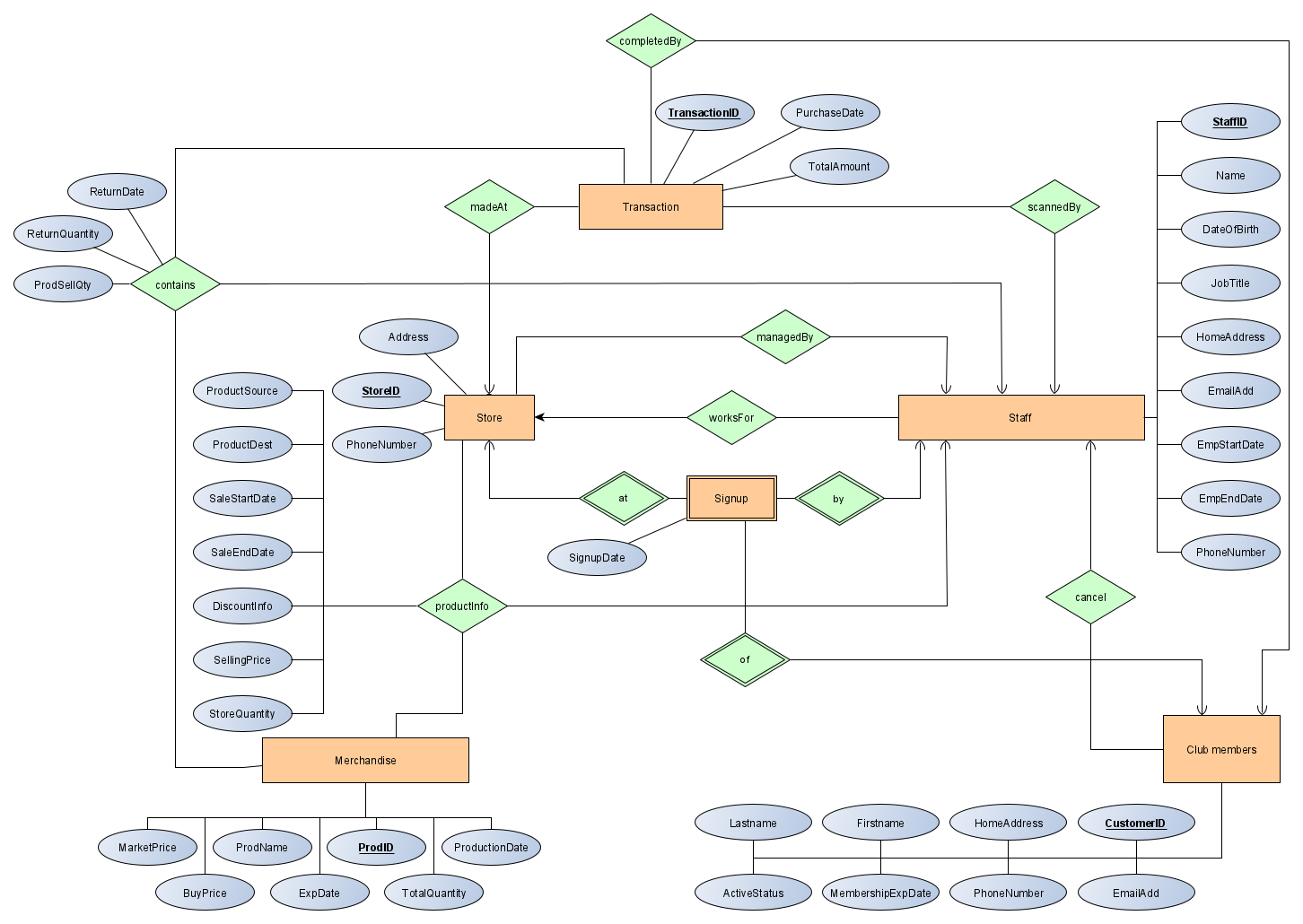
**Registration Staff View:**



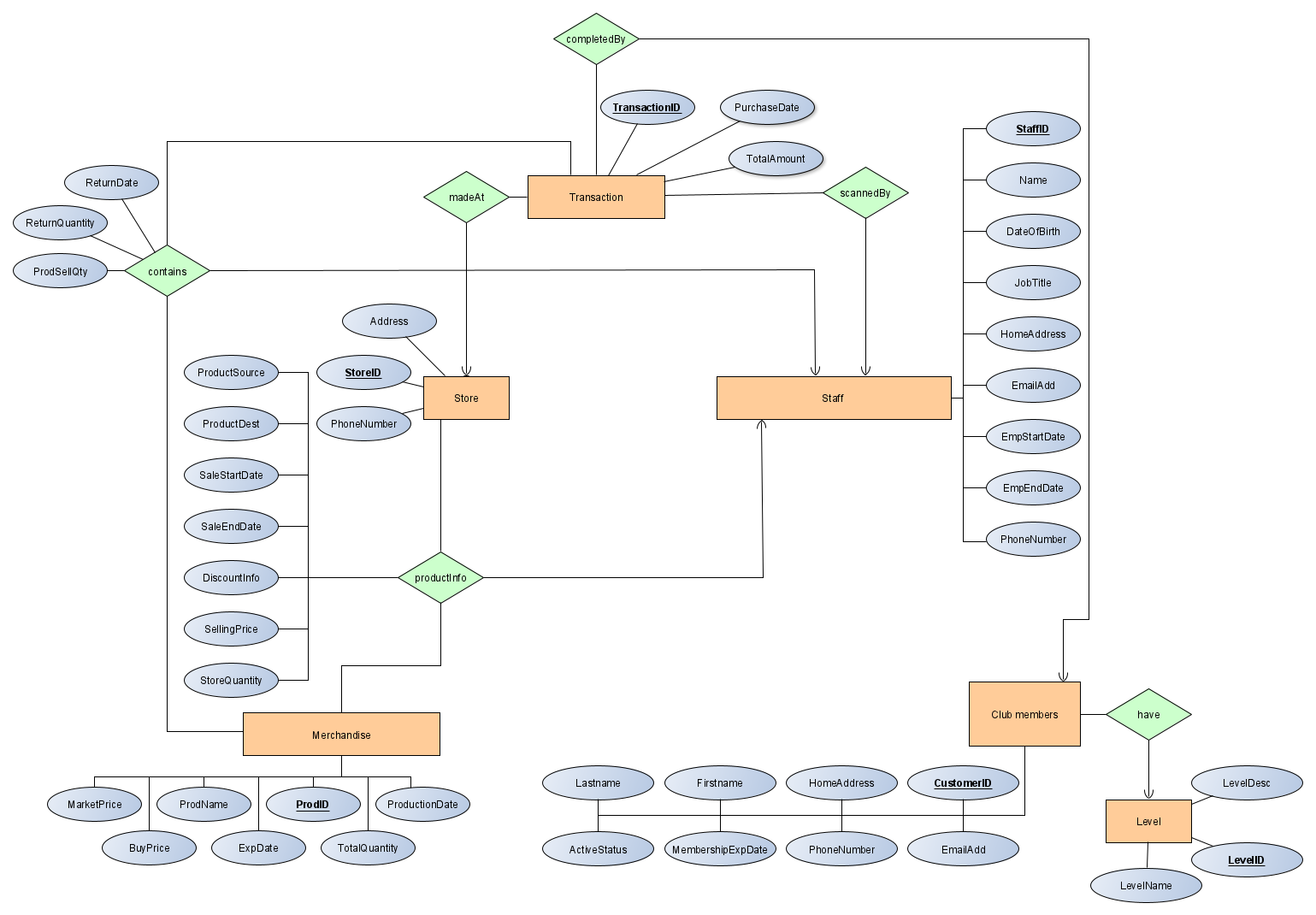
**Warehouse Staff View:**



**Manager View:**



**Cashier’s View:**



**Description of Local E/R diagrams:**

* The staff working at the wholesale store can be managers, billing staff, registration staff, warehouse staff, cashier etc.
* Merchandise uses ProdID as a key, since no two distinct products can have the same product number.
* Staff uses Staff ID as a key because there are many staff working in various departments. A unique id is used to distinguish between them.
* Store uses StoreID as a key, to differentiate between different stores located around the region/country.
* Club members are given a unique ID because two customers cannot have the same number.
* The ‘Levels’ entity use LevelID as a key, to distinguish between the different types of levels, such as gold or platinum.
* Suppliers are given a unique ID in order to differentiate between each supplier.
* Transactions have a unique id called Transaction\_id. This is given so that the staff can handle the transactions taking place in different stores, and can be used to handle returns.
* SignUp is a weak entity connected between Stores and Staff. It does not exist without a Store or Staff. One staff can only sign up one customer at a time.
* Each product is supplied by exactly one supplier, and each supplier can supply many products.
* Each staff works for at most one store and each store is managed by exactly one staff (manager). See assumption #1
* Each transaction is completed by exactly one club member and club members can have exactly one level.
* Each staff can sign up exactly one customer at exactly one store. Check assumption #3.
* Many new shipments of products can be handled by exactly one staff and many bills can be generated for suppliers by exactly one staff.
* Multiple transactions can be made at one store and scanned by one staff.
* The membership of many club members can be cancelled by exactly one staff member. This is done so that there would not be any conflict when cancelling the memberships.
* Many rewards checks are given out by exactly one billing staff member based on their membership level.

**Local Relational Schemas:**

**Admin View:**

Staff(StaffID, Name, DateofBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, EmpEndDate, PhoneNumber)

Transaction(TransactionID, PurchaseDate, TotalAmount)

Store(StoreID, Address, PhoneNumber)

Merchandise(ProductID, ProductName, ProductionDate, MarketPrice, BuyPrice, ExpirationDate, TotalQuantity)

Supplier(SupplierID, Location, EmailAdd, PhoneNumber, SupplierName)

ClubMembers(CustomerID, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

Level(LevelID, LevelDesc, LevelName)

SignUp(SignUpDate, CustomerID, StaffID, StoreID)

reward(Cashback, StaffID, CustomerID, LevelID)

productInfo(ProductSource, ProductDest, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice, StoreQuantity, StoreID, ProductID)

contains(ReturnDate, ReturnQuantity, ProdSellQty, ProductID, TransactionID, StaffID)

generateBills(LastBillDate, LastBillAmount, StaffID, SupplierID)

newShipments(StaffID, ProductID)

suppliedBy(SupplierID, ProductID)

managedBy(StaffID, StoreID)

worksFor(StaffID, StoreID)

**Billing Staff View:**

Staff(StaffID, Name, DateofBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, EmpEndDate, PhoneNumber)

Transaction(TransactionID, PurchaseDate, TotalAmount)

Store(StoreID, Address, PhoneNumber)

Merchandise(ProductID, ProductName, ProductionDate, MarketPrice, BuyPrice, ExpirationDate, TotalQuantity)

Supplier(SupplierID, Location, EmailAdd, PhoneNumber, SupplierName)

ClubMembers(CustomerID, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

Level(LevelID, LevelDesc, LevelName)

reward(Cashback, StaffID, CustomerID, LevelID)

productInfo(ProductSource, ProductDest, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice, StoreQuantity, StoreID, ProductID)

contains(ReturnDate, ReturnQuantity, ProdSellQty, ProductID, TransactionID, StaffID)

generateBills(LastBillDate, LastBillAmount, StaffID, SupplierID)

**Registration Staff View:**

Staff(StaffID, Name, DateofBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, EmpEndDate, PhoneNumber)

ClubMembers(CustomerID, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

Level(LevelID, LevelDesc, LevelName)

Store(StoreID, Address, PhoneNumber)

SignUp(SignUpDate, CustomerID, StaffID, StoreID)

**Warehouse Staff View:**

Staff(StaffID, Name, DateofBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, EmpEndDate, PhoneNumber)

Store(StoreID, Address, PhoneNumber)

Merchandise(ProductID, ProductName, ProductionDate, MarketPrice, BuyPrice, ExpirationDate, TotalQuantity)

Supplier(SupplierID, Location, EmailAdd, PhoneNumber, SupplierName)

productInfo(ProductSource, ProductDest, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice, StoreQuantity, StoreID, ProductID)

newShipments(StaffID, ProductID)

suppliedBy(SupplierID, ProductID)

**Cashier View:**

Staff(StaffID, Name, DateofBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, EmpEndDate, PhoneNumber)

Transaction(TransactionID, PurchaseDate, TotalAmount)

Store(StoreID, Address, PhoneNumber)

Merchandise(ProductID, ProductName, ProductionDate, MarketPrice, BuyPrice, ExpirationDate, TotalQuantity)

ClubMembers(CustomerID, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

Level(LevelID, LevelDesc, LevelName)

productInfo(ProductSource, ProductDest, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice, StoreQuantity, StoreID, ProdID)

contains(ReturnDate, ReturnQuantity, ProdSellQty, ProductID, TransactionID, StaffID)

madeAt(StoreID, TransactionID, StaffID)

**Manager View:**

Staff(StaffID, Name, DateofBirth, JobTitle, HomeAddress, EmailAdd, EmpStartDate, EmpEndDate, PhoneNumber)

Transaction(TransactionID, PurchaseDate, TotalAmount)

Store(StoreID, Address, PhoneNumber)

Merchandise(ProductID, ProductName, ProductionDate, MarketPrice, BuyPrice, ExpirationDate, TotalQuantity)

ClubMembers(CustomerID, HomeAddress, FirstName, LastName, EmailAdd, PhoneNumber, MembershipExpDate, ActiveStatus)

Level(LevelID, LevelDesc, LevelName)

managedBy(StaffID, StoreID)

productInfo(ProductSource, ProductDest, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice, StoreQuantity, StoreID, ProductID)

contains(ReturnDate, ReturnQuantity, ProdSellQty, ProductID, TransactionID, StaffID)

SignUp(SignUpDate, CustomerID, StaffID, StoreID)

worksFor(StaffID, StoreID)

1. **Local Schema Documentation:**

Entity Sets to Relations:

● The entity sets in our diagram were made into relations, with attributes of the entities (Staff, Supplier, Customer, Store, Merchandise)

Combining Many-One Relationships:

In the following places, an attribute is made by combining a many-one relationship. As this reduces redundancy and decreases the overhead that many tables cause, it makes queries quicker.

* Weak entity set SignUp was made into a relation with all its attributes plus the StoreID, Staff ID and Customer ID which are the foreign keys from the Store, Staff and ClubMembership entity sets respectively. Including the StoreID represents the “at” relationship in the diagram between SignUp and Store, including StaffID represents the “by” relationship in the diagram between SignUp and Staff, including CustomerID represents the “of” relationship in the diagram between SignUp and ClubMember; therefore, we do not have a separate relationship in the schema for “at”,”by” or “of”. SignUp was made into a weak entity set because it is a many-one relationship that we need to know the StoreID, StaffID, and CustomerId in combination with SignUpDate to look it up.
* Entity set Levels was made into a relation using all its attributes plus a CustomerID that comes from combining its many-one relationship to ClubMember. Thus we do not have a separate relationship for the “have” relation from the E/R diagram.
* Entity set Transactions was made into the relation using its attributes and the StaffID from its many-one relationship (“scannedBy”) with Staff. It also includes the StoreID from its many-one relationship with Store through the “madeAt” relationship.
* Entity set Supplier was made into a relation using all its attributes plus a ProductID from its many-one relationship (“suppliedBy”) with Merchandise. It also includes the StaffID from its many-one relationship with Staff through the “generateBills” relationship.

Relationships to Relations:

* Relationships newShipments, generateBills, contains, have, productInfo, and reward from the E/R diagrams have each been turned into relations in our schema. Their attributes are the keys of the entities they represent.
* generateBills has additional attributes LastBillDate and LastBillAmount. The relationship also gets the SupplierID attribute from the Supplier entity. productInfo has additional attributes ProductSource, ProductDest, SaleStartDate, SaleEndDate, DiscountInfo, SellingPrice, StoreQuantity. This relationship also take the StoreID attribute from the Store entity.