Relational Database Design - Final Project

Introduction

For my final project, I imagine that I am constructing a database for an ice cream store chain. It will keep track of every sale of ice cream at every store and allow the company to perform data analysis. I envision the database keeping track of the following entities:

- Stores: The company has multiple stores, and the database will record each store's opening date, location, and size.

- Employees: Basic information about each employee will be recorded, including name, hire date, salary, date of birth, and store location.

- Customers: The business will gather data about customers for each sale. Customers can have a favorite ice cream flavor and can sign up for a membership at a store in order to receive discounts.

- Orders: Each order will be described in terms of a single ice cream flavor, type of treat (cone, cup, sundae, or shake), and size. The date, time, and price of each order will be recorded, along with membership and seasonal discounts applied. I assume that prices are the same at all stores.

- Flavors: Each ice cream flavor will be recorded along with its ingredients.

- Ingredients: The database will keep track of the ingredients used in the ice cream flavors, including their supplier and cost.

Entity - Relationship Model

I will assume the following relationships between the entities above:

- A store must have one or more employees; an employee must be employed at one and only one store.

- A store may have one or more orders; an order must be made at one and only one store.

- An employee must have one and only one supervisor; an employee may supervise one or more other employees.

- An order must be processed by one and only one employee; an employee may process one or more orders.

- A customer may subscribe to a membership with one and only one store; a store may have one or more members.

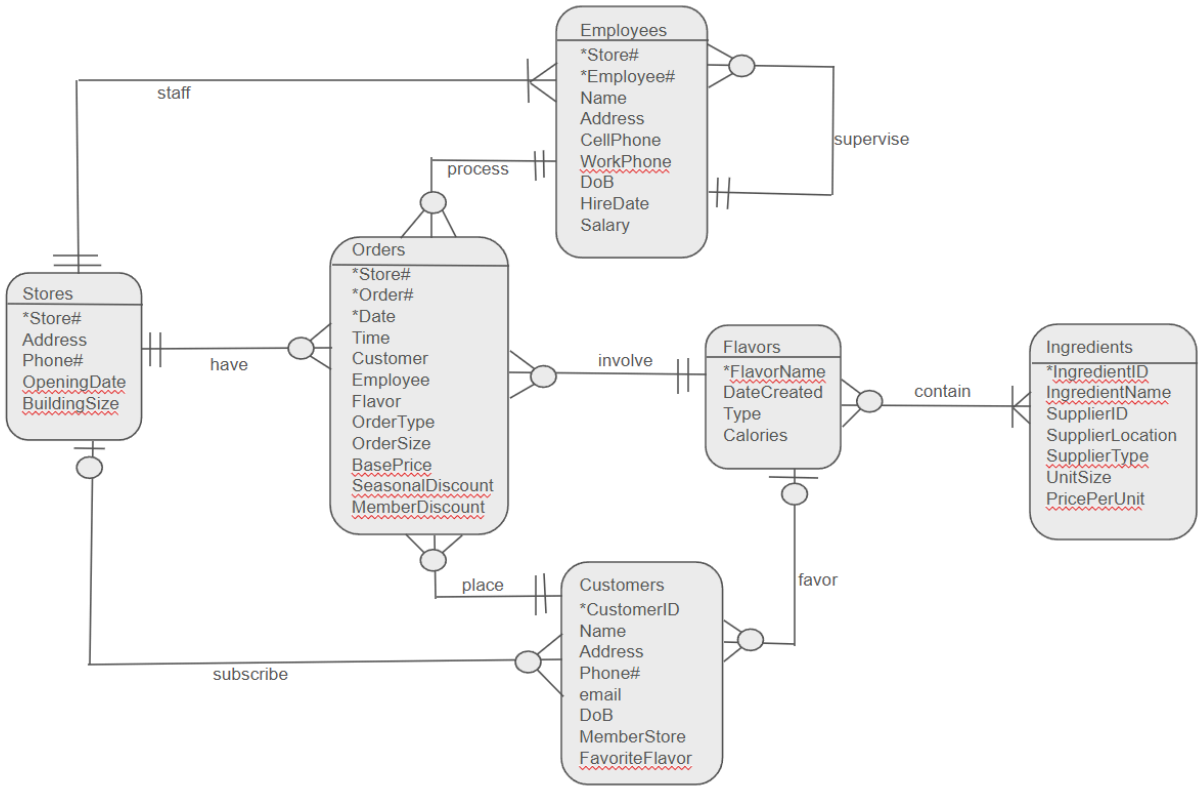
- A customer may place one or more orders; an order must be placed by one and only one customer.

- A customer may have one and only one favorite flavor; a flavor may be favored by one or more customers.

- An order must involve one and only one flavor; a flavor may be involved in one or more orders.

- A Flavor must contain one or more ingredients; an ingredient may be used in one or more flavors.

These relationship are represented in the entity-relationship diagram below:



Relational Model

Building the relational model requires identifying the primary and foreign keys for each relation.

- The primary key for Stores is Store#.
- The primary keys for Employees are Store# and Employee#. Store# also serves as a foreign key linking the employee to a store instance. A new foreign key called SupervisorID links to the employee# of the supervisor.
- The primary key for Flavors is FlavorName.
- The primary key for Customers is CustomerID. MemberStore is a foreign key matching the Store# of the store for which the customer subscribed to a membership, and FavoriteFlavor is a foreign key matching the FlavorName of the customer's favorite flavor.
- The primary keys for Orders are Store#, Order#, and Date. The foreign keys are Store#, CustomerID, Employee, and Flavor.
- The primary key for Ingredients is IngredientID.
- A new relation called Flavors_Ingredients is added to list the pairings of flavors and ingredients, using FlavorName and IngredientID as both primary and foreign keys.

The relational schemas are listed below:

- Stores (Store#, Address, Phone#, OpeningDate, BuildingSize)
- Employees (Store# (fk), Employee#, SupervisorID (fk), Name, Address, CellPhone, WorkPhone, DoB, HireDate, Salary)
- Customers (CustomerID, Name, Address, Phone, email, DoB, MemberStore (fk), FavoriteFlavor (fk))
- Orders (Store# (fk), Order#, Date, Time, CustomerID (fk), Employee# (fk), FlavorName (fk), OrderType, OrderSize, BasePrice, SeasonalDiscount, MemberDiscount)

- Flavors (FlavorName, DateCreated, Type, Calories)

- Ingredients (IngredientID, IngredientName, SupplierID, SupplierLocation, SupplierType, UnitSize, PricePerUnit)

- Flavors_Ingredients (FlavorName (fk), IngredientID (fk), quantity)

Normalization to 3NF (third normal form)

The functional dependencies of the relations are as follows:

- Stores
    - FD1: Store# → Address, Phone#, OpeningDate, BuildingSize

- Employees
    - FD1: Store#, Employee# → SupervisorID, Name, Address, CellPhone, WorkPhone, DoB, HireDate, Salary
    - FD2: Store# → WorkPhone

- Customers
    - FD1: CustomerID → Name, Address, Phone, email, DoB, MemberStore, FavoriteFlavor

- Orders
    - FD1: Store#, Order#, Date → Time, CustomerID, Employee#, FlavorName, OrderType, OrderSize, BasePrice, SeasonalDiscount, MemberDiscount
    - FD2: Date → SeasonalDiscount
    - FD3: Store#, CustomerID → MemberDiscount
    - FD4: OrderType, OrderSize → BasePrice

- Flavors
    - FD1: FlavorName → DateCreated, Type, Calories

- Ingredients
    - FD1: IngredientID → IngredientName, SupplierID, SupplierLocation, SupplierType, UnitSize, PricePerUnit
    - FD2: SupplierID → SupplierLocation, SupplierType
    - FD3: IngredientName, SupplierID, UnitSize → PricePerUnit

- Flavors_Ingredients
    - FD1: FlavorName, IngredientID → quantity

The following steps are needed to normalize the relations:

- Stores, Customers, Flavors, and Flavor_Ingredients are already in 3NF because they have no partial or transitive functional dependencies.

- Employees is in 1NF but not 2NF because FD2 is a partial functional dependency in which the employee's work phone number depends only on the store. Because the Stores relation already has a phone number, this attribute can simply be removed from the Employees relation:

    - Employees (Store# (fk), Employee#, SupervisorID (fk), Name, Address, CellPhone, DoB, HireDate, Salary)

- Orders is in 1NF but not 2NF because FD2 is a partial functional dependency. To fix this, I will create a new relation for seasonal discounts using Date as the primary key:

    - Orders (Store# (fk), Order#, Date (fk), Time, CustomerID (fk), Employee# (fk), FlavorName (fk), OrderType, OrderSize, BasePrice, MemberDiscount)
    - SeasonalDiscounts (Date, discount)

- Orders is now in 2NF but not 3NF because FD3 and FD4 are transitive functional dependencies. Fixing these requires adding a new relation for member discounts using Store# and CustomerID as the primary keys and adding a new relation for BasePrice using OrderType and OrderSize as the primary keys:

    - Orders (Store# (fk), Order#, Date (fk), Time, CustomerID (fk), Employee# (fk), FlavorName (fk), OrderType (fk), OrderSize (fk))
    - SeasonalDiscounts (Date, discount)
    - MemberDiscounts (Store#, CustomerID, discount)
    - BasePrices (OrderType, OrderSize, Price)

- Ingredients is in 2NF but not 3NF because FD2 and FD3 are transitive functional dependencies. To fix these I will create a Supplier relation using SupplierID as the primary key and an IngredientPrice relation using IngredientName, SupplierID, and UnitSize as the primary keys:

    - Ingredients (IngredientID, IngredientName (fk), SupplierID (fk), UnitSize (fk))
    - Suppliers (SupplierID, SupplierLocation, SupplierType)
    - IngredientPrices (IngredientName, SupplierID, UnitSize, PricePerUnit)

Final Relational Model

After normalizing the relations, the final relational model can be written out as follows:

- Stores (Store#, Address, Phone#, OpeningDate, BuildingSize)

- Employees (Store# (fk), Employee#, SupervisorID (fk), Name, Address, CellPhone, DoB, HireDate, Salary)

- Customers (CustomerID, Name, Address, Phone, email, DoB, MemberStore (fk), FavoriteFlavor (fk))

- Orders (Store# (fk), Order#, Date (fk), Time, CustomerID (fk), Employee# (fk), FlavorName (fk), OrderType (fk), OrderSize (fk))

- SeasonalDiscounts (Date, discount)

- MemberDiscounts (Store#, CustomerID, discount)

- BasePrices (OrderType, OrderSize, Price)

- Flavors (FlavorName, DateCreated, Type, Calories)

- Ingredients (IngredientID, IngredientName (fk), SupplierID (fk), UnitSize (fk))

- Suppliers (SupplierID, SupplierLocation, SupplierType)

- IngredientPrices (IngredientName, SupplierID, UnitSize, PricePerUnit)

- Flavors_Ingredients (FlavorName (fk), IngredientID (fk), quantity)