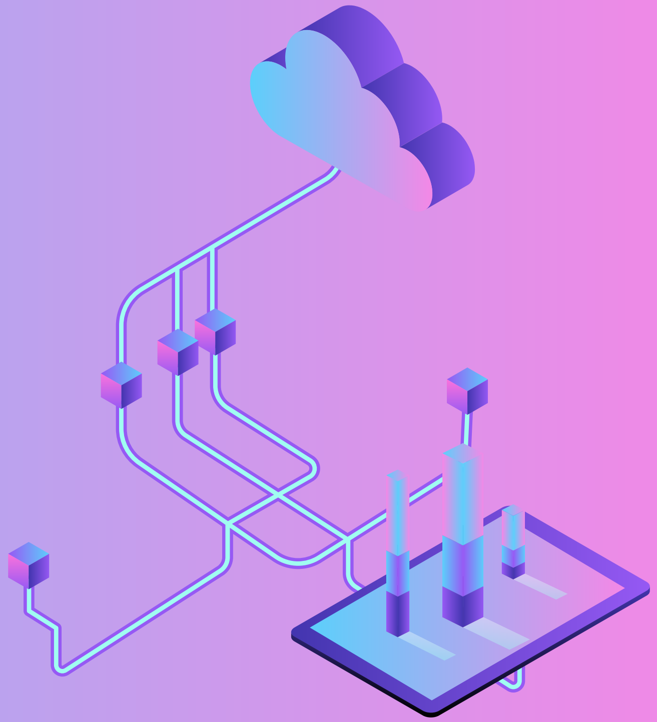
Relational Database Creation for a Bakery business

Ana K. Santiago Monroy
Information School, University of Washington
IMT 543: Relational Database Management Systems
Prof. Thakkar
May 31, 2024

AGENDA

- Business
- Data modeling
- DDL
- Takeaways



BUSINESS CONTEXT

Bakery franchising

Thriving bakery, complex administration across locations

Bakery Operations

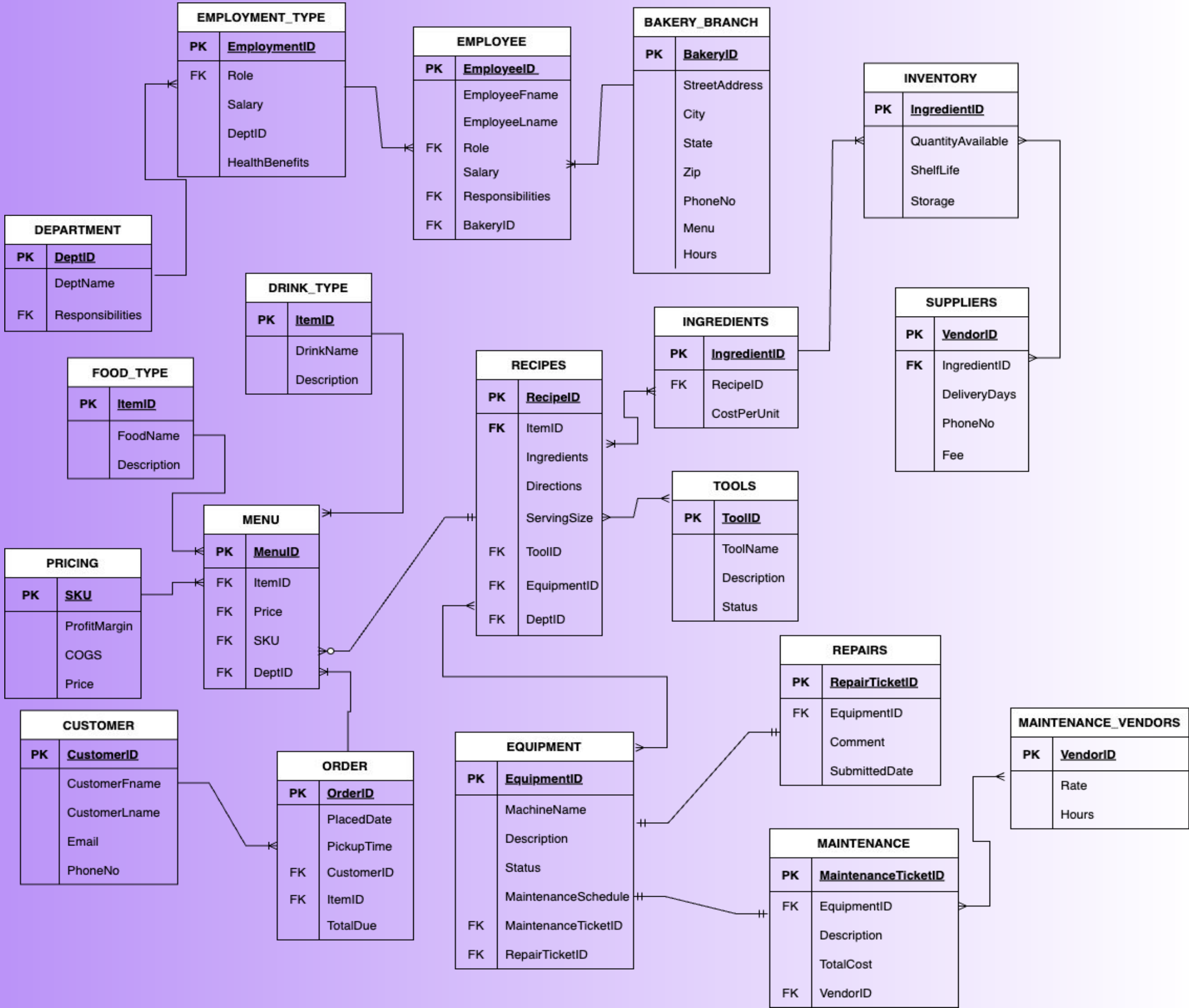
Coordinating locations, clients, suppliers, repair and maintenance vendors, on top of franchise staff

FIRST DRAFT ERD

Business Nouns

TABLES

Attributes



REVISED ERD

Added a SKU FK to the DRINK_TYPE and FOOD_TYPE tables identifier

Connected EMP-DEPT tables to remaining tables

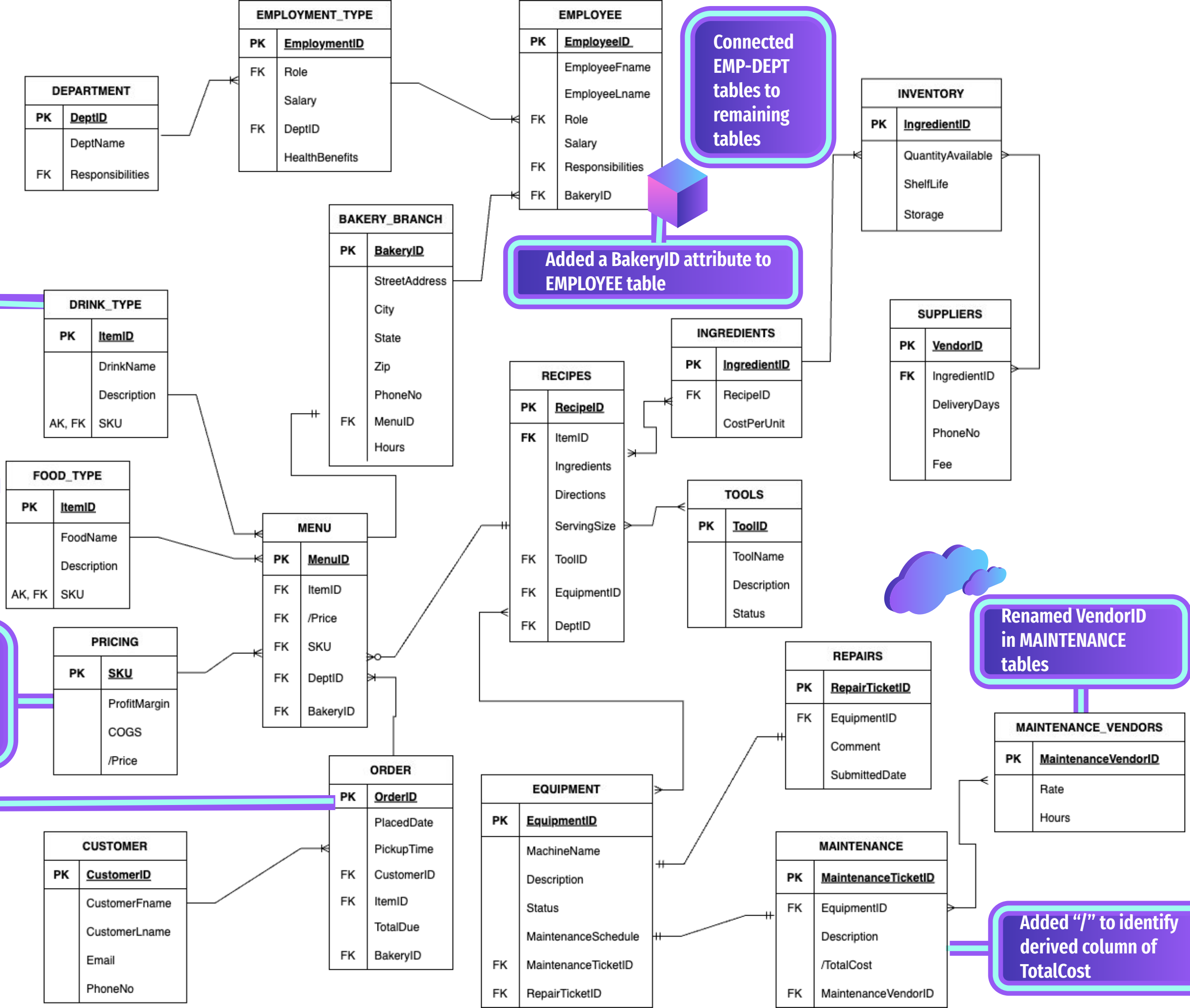
Added a BakeryID attribute to EMPLOYEE table

Added "/" to identify the derived attribute of Price

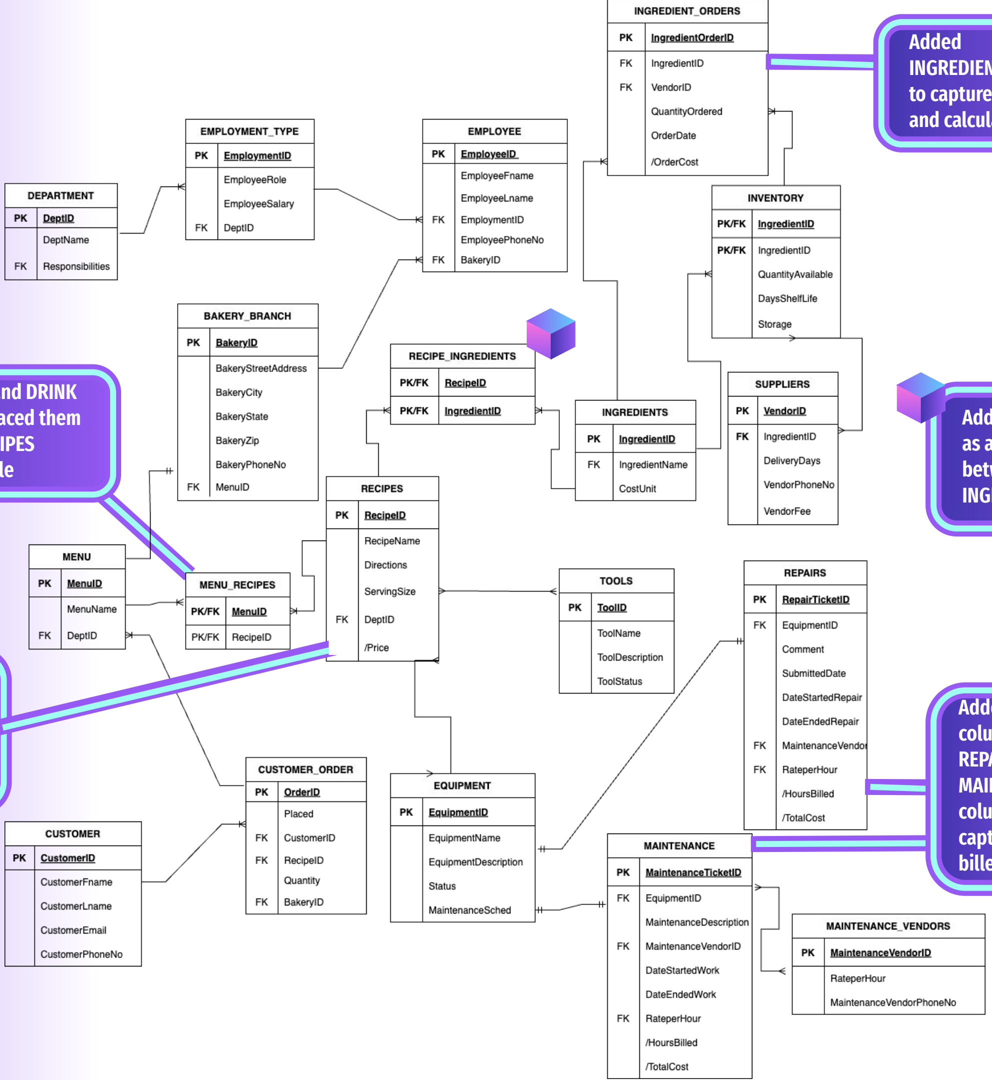
Renamed VendorID in MAINTENANCE tables

Added BakeryID as FK to the order table

Added "/" to identify derived column of TotalCost



FINAL 3NF ERD



Added INGREDIENT_ORDERS tbl to capture restock orders and calculate costs

Deleted FOOD and DRINK tables and replaced them with MENU_RECIPES association table

Added RECIPE_INGREDIENTS as an association table between RECIPES & INGREDIENTS

Removed PRICING table and added computed column to RECIPES to capture those values

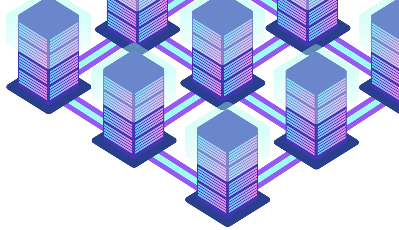
Added computed columns to REPAIRS, MAINTENANCE columns to capture hours billed and costs

Data Definition Language design

Data	From	To	Method
Customer info	CUSTOMER_ORDER	CUSTOMER	Stored Procedure
Order Insertion & new customer creation	CUSTOMER_ORDER	CUSTOMER_ORDER & CUSTOMER	Trigger
Inventory Stock check	INVENTORY	-	Stored Procedure
Fire up restock order	INVENTORY	INGREDIENTS_ORDER	Trigger

Trigger

DDL



```
CREATE TRIGGER trg_CustomerID_AfterOrderPlaced
ON dbo.CUSTOMER_ORDER
AFTER INSERT
AS
BEGIN
       DECLARE @OrderID INT
       DECLARE @CustomerFName VARCHAR(75)
       DECLARE @CustomerLName VARCHAR(75)
       DECLARE @CustomerEmail VARCHAR(100)
       DECLARE @CustomerPhoneNo VARCHAR(15)

       DECLARE cur CURSOR FOR
       SELECT i.OrderID, c.CustomerFName, c.CustomerLName, c.CustomerEmail, c.CustomerPhoneNo
       FROM INSERTED i
       JOIN dbo.CUSTOMER_ORDER co ON i.OrderID = co.OrderID
       JOIN dbo.CUSTOMER c ON co.CustomerID = c.CustomerID

       OPEN cur
       FETCH NEXT FROM cur INTO @OrderID, @CustomerFName, @CustomerLName, @CustomerEmail,
       @CustomerPhoneNo

          WHILE @@FETCH_STATUS = 0
              BEGIN
              EXEC dbo.CheckandCreateCustomerID
                  @OrderID = @OrderID,
                  @CustomerFName = @CustomerFName,
                  @CustomerLName = @CustomerLName,
                  @CustomerEmail = @CustomerEmail,
                  @CustomerPhoneNo = @CustomerPhoneNo
              FETCH NEXT FROM cur INTO @OrderID, @CustomerFName, @CustomerLName, @CustomerEmail,
       @CustomerPhoneNo
          END
          CLOSE cur
          DEALLOCATE cur
       END;
```

```
CREATE PROCEDURE  dbo.CheckandCreateCustomerID
        @OrderID  INT,
        @CustomerFName  VARCHAR(75),
        @CustomerLName  VARCHAR(75),
        @CustomerEmail  VARCHAR(100),
        @CustomerPhoneNo  VARCHAR(15)
AS
BEGIN
        IF (@OrderID IS NULL OR @CustomerFName  IS NULL OR @CustomerLName  IS NULL OR @CustomerEmail  IS NULL OR @CustomerPhoneNo  IS NULL)
        BEGIN
            THROW 51000, 'Customer information cannot be empty. Fill in all values to proceed.'   , 1
            RETURN
           END
           DECLARE @CustomerID  INT

           SELECT  @CustomerID  = CustomerID
           FROM CUSTOMER
           WHERE CustomerEmail =  @CustomerEmail

           IF @CustomerID  IS NULL
           BEGIN
               INSERT INTO dbo.CUSTOMER (CustomerFName, CustomerLName, CustomerEmail, CustomerPhoneNo)
               VALUES ( @CustomerFName, @CustomerLName, @CustomerEmail, @CustomerPhoneNo  )

               SET  @CustomerID  = SCOPE_IDENTITY()

               UPDATE dbo.CUSTOMER_ORDER
               SET CustomerID =  @CustomerID
               WHERE OrderID =  @OrderID
           END
           ELSE
           BEGIN
               RAISERROR( 'CustomerID already exists.' , 11, 1)
               RETURN
           END
        END;
```

Stored Procedure

DDL

TAKEAWAYS

NORMALIZATION

Continued to refine data modeling in coding

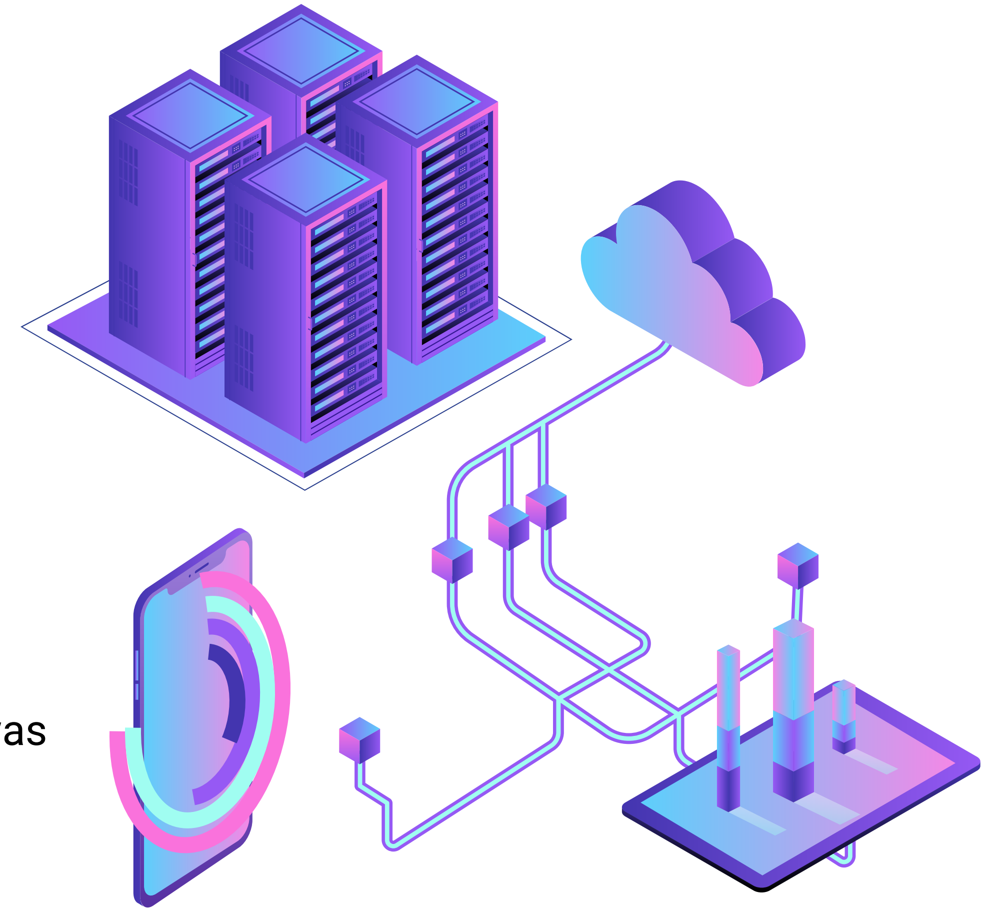
DDL & DML

Leveraging parameters, CTE, combining triggers & stored procedures

BUSINESS

Nouns, requisites, specifications

Thank you!



CREDITS: this presentation template was created by **Slidesgo**