**COMP1140**

**Assignment 1**

**Mega Pizza Project**

**Requirement Analysis and Conceptual Design**

**Due: during in class, Thursday, 23th July 2020**

**Table of Contents**

[Data Requirements 2](#_Toc49955267)

[Transaction Requirements 6](#_Toc49955268)

[Business rules 6](#_Toc49955269)

[Entity Types 7](#_Toc49955270)

[Relationship types 8](#_Toc49955271)

[Attributes 9](#_Toc49955272)

[Reflection for my assignment 1 14](#_Toc49955273)

[Relational Model Schema 14](#_Toc49955274)

[Normalizing the relational schema to Boyce-Codd Normal Form 17](#_Toc49955275)

[the relations that are in BCNF 32](#_Toc49955276)

# Data Requirements

**Order Processing**

Mega Pizza takes orders via phone as well as through walk-in customers. Mega Pizza provides both delivery and pickup services. When a customer orders, the customer’s phone number is entered to the system along with the id of the staff taking the order. If the customer has previously ordered, the name and address appear on the screen. The customer is then asked for his/her name and address, and then the order is taken. If the customer has not ordered before or if the name and address given do not correspond with what are recorded in the computer, a new customer record is created and the order is taken. For each phone order, the time the call was answered as well as the time the call was terminated is recorded. For each walk-in order, the time the customer walks in is recorded. For a phone order, after the order has been taken, a verification process occurs whereby the staff dials the number given and confirms the order with the customer. If it is not confirmed, the customer is recorded as a hoax and the order is kept on hold (if and until the customer calls back in which case the verification process takes place before hoax is removed and order goes through). Each order contains date of order, the items ordered, quantity of each item, price of each item, total amount due, payment method, order status, type (phone/walk-in), and Page 2 of 4 description. If the payment is via card, a payment approval number is recorded. For a phone order, if the order is a pickup order, the pickup time is recorded; if the order is a delivery order, the delivery time and address and the driver who delivered the order are recorded.

**Two entities included here:**

**1. Order**

**2. Customer**

The following are data requirement for **Order** entity:

* OrderNo(Unique order identifier)
* OrderDate&Time(date of order)
* Total amount due
* TypeOfOrder (phone or walk-in)
* payment Method
* order status
* Approval number(if payment is via card)
* Pick up time(for phone order)
* Delivery time(for delivery order)
* DriverInfo(for delivery order)

The following are data requirement for **Customer** entity:

* customer's Id(Unique customer identifier)
* FirstName
* LastName
* address
* phone
* Status

**Menu Items, Ingredients and Suppliers**

Each item in the menu has an item code (unique), name, size and a current selling price. An item in the menu is made up of a number of ingredients. The ingredients and quantities used for the item are recorded in the database. Each ingredient has a code (unique), name, type, description, stock level at current stocktake period, date last stocktake was taken, suggested current stock level, reorder level, and a list of suppliers who supply the ingredients. A supplier can supply many ingredients. Each ingredient can be supplied by many suppliers. A stocktake is taken each week, where the actual levels of ingredients in store are recorded. This is then compared with suggested levels (based on orders for the week). This report is used by the store manager to order ingredients for the following week. Information about ingredient orders needs to be maintained in the database, including order number, date of the order, date received order, total amount, order status, description, quantity and price of all ingredients, supplier number, and ingredient code.

**Four entities included here:**

1. **Menu Items**
2. **Ingredients**
3. **Ingredients Order**
4. **Suppliers**

The following are data requirement for **MenuItems** entity:

* ItemNO(Unique menu item identifier)
* Name
* Size
* CurrentSellingPrice
* Description.

The following are data requirement for **Ingredients** entity:

* code (Unique ingredient identifier)
* name
* type
* Description
* StockLevelAtCurrentPeriod
* StockLevelAtLastStocktake
* SuggestedStockLevel
* reorder level
* put in time
* shelf life

The following are data requirement for **Ingredients Order** entity:

* OrderID
* Date of the order
* Date received order
* TotalAmount
* Status
* Description.

The following are data requirement for **Suppliers** entity:

* SupplierNo(Unique supplier identifier)
* Name
* Address
* Phone
* Email
* ContactPerson

**Employees**

Employees at the store can be divided into two types: those who work in the shop are paid hourly and those who carry out deliveries are paid by the number of deliveries. For each employee, there is an employee number, firstname, lastname, postal address, contact number, tax file number, bank details (bank code, bank name, account number), payment rate, status, and a description. Drivers also have a driver’s license number. Hours are not regular and a record is kept for each time an employee works – a shift (start date, start time, end date, end time). The orders a driver delivers during a shift is kept on the record. Payment rates for shop workers and drivers are maintained in the database. Employee payments are made for each shift to the employee’s bank account. Employee payment record needs to be maintained in the database. It includes gross payment, tax withheld, total amount paid, payment date, payment period starts date, payment period end date, and bank details of the employee

**Three entities included here:**

1. **Employee**
2. **Payment**
3. **Shift**

The following are data requirement for **Employee** entity:

* StaffNo(Unique staff identifier)
* FirstName
* LastName
* postal address
* contact number
* tax file number
* bank details (bank code, bank name, account number)
* payment rate
* hourly salary(for in store staff)
* date of birth
* status(work or rest)
* description
* driver’s license number(for driver)

The following are data requirement for **Payment** entity:

* Payment record Id(Unique payment identifier)
* gross payment
* tax withheld
* total amount paid
* payment period start date
* payment period end date
* bank details of the employee(bank code, bank name, account number)

The following are data requirement for **Shift** entity:

* shiftNo(Unique shift identifier)
* start date(The date the employee starts working)
* start time(The time the employee starts working)
* end date(The date the employee ends working)
* end time(The time the employee ends working)
* Shift type
* deliveryQuantity(for driver)
* number of working hours(for in store staff)

# Transaction Requirements

* **Data Manipulation:**

1. Insert, Update and Delete existing Order.
2. Insert, Update and Delete shift data.
3. Insert, Update and Delete a menu item in the Mega Pizza.
4. Insert, Update and Delete customer information
5. Insert, Update and Delete driver data.
6. Insert, Update and Delete employee information.
7. Insert, Update and Delete existing supplier.
8. Insert, Update and Delete existing ingredient order.
9. Insert, Update and Delete payment data.
10. Insert, Update and Delete existing ingredient.

* **Queries:**

1.Search a payment based on an employee number, on a particular pay date.

2. Search a delivery based on a customer order number.

3. List ingredients and quantity used for menu items.

4. Report of ingredient levels for current period and suggested stock levels.

5. The total amount of the payment based on the hourly salary, start time, and end time.

6. Search the customer information from customer No.

7. Search a driver shift based on an shift No.

8. List suppliers based on supplier No and name.

# Business rules

• The amount of each ingredient remaining must be updated every time some is used.

• The results of the weekly stock take must be input into the database.

• When an ingredients stock level decreases below its reorder level an order for the

ingredient must be placed.

• A new customer must be marked as un-verified until the verification process is successfully completed.

• Employees must record each shift they work in the database.

• An employee can only be either an in-store worker or a delivery driver.

• Employees cannot delete data from the database.

• An Employees’ status can only be either full time or part time.

• An order can only be phone order or walk in order.

• Payments can only be added by accounting staff

• An orders payment method can only have one of the following values:

• Credit card

• Debit card

• Cash

• An order’s type can only be either:

• Pick up

• Delivery

• If an order is paid for using a card, the approval number must be stored in the order’s payment Approval No

**Data Dictionary**

# Entity Types

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Name** | **Description** | **Aliases** | **Occurrence** |
| Order | Describing orders customers have made | Pizza Order | When an order is made by a customer |
| WalkInOrder | The customer makes an order in store |  | When a customer went to the store to make an order |
| PhoneOrder | The customer orders by phone |  | When a customer makes an order by phone |
| Delivery | pick-up or delivery. |  | When the customer wants to delivery |
| PickUp | pick-up or delivery. |  | When the customer orders by phone and picks up |
| MenuItem | Dishes in Mega pizza | Menu | When the customer orders dishes. |
| Customer | People who go to the Mega Pizza to order |  | When a person orders a meal from Mega Pizza |
| Payment | Payment for shop workers and drivers |  | Boss wants to pay employees' salaries |
| DriverPayment | Payment rates for drivers |  | Boss wants to pay drivers' salaries |
| InStoretaffPayment | Payment rates for shop workers | Shop worker | Boss wants to pay shop workers' salaries |
| Shift | A shift for shop workers or drivers |  | When an employee finishes work and shifts |
| DriverShift | A shift for drivers |  | When drivers finish work and shifts |
| InStoreStaffShift | A shift for shop workers |  | When a shop worker shifts |
| Employee | Two Types: instore, and delivery. | Staff | Mega Pizza has two types of employees |
| In Store Staff | Employees working in the Mega Pizza store |  | Employee works in the store |
| Driver | An employee responsible for delivering phone orders |  | Driver delivers a phone order |
| Ingredient | ingredients of a menu item |  | Ingredients for item |
| Supplier | Company supply ingredients to the Mega Pizza store | Provider company | When Mega pizza need ingredients |
| IngredientOrder | supply order provided by Mega Pizza to the supplier | Ingredient order | Made by manager |

# Relationship types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Multiplicity** | **Relationship** | **Multiplicity** | **Entity Name** |
| Order | 0..\* | Consists of | 1..\* | MenuItem |
| (Man, Or) | Generalisation | (Man, Or) | WalkInOrder |
| (Man, Or) | Generalisation | (Man, Or) | PhoneOrder |
| PhoneOrder | (Man, Or) | Generalisation | (Man, Or) | Delivery |
| (Man, Or) | Generalisation | (Man, Or) | PickUp |
| MenuItem | 1..\* | Included in | 0..\* | Order |
| 0..\* | Consists of | 1..\* | Ingredient |
| Customer | 1..1 | Makes | 1..\* | WalkInOrder |
| 1..1 | Makes | 1..\* | PhoneOrder |
| Employee | (Man, Or) | Generalisation | (Man, Or) | InStoreStaff |
| (Man, Or) | Generalisation | (Man, Or) | Driver |
| InStoreStaff | 1..1 | Takes | 0..\* | Order |
| Driver | 1..1 | Delivers | 0..\* | Delivery |
| Ingredient | 1..\* | Included in | 0..\* | MenuItem |
| 1..\* | Reports | 0..\* | IngredientOrder |
| Supplier | 1..1 | Checks | 0..\* | IngredientOrder |
| 1..\* | Supply | 0..\* | Ingredient |
| Payment | (Man, Or) | Generalisation | (Man, Or) | DriverPayment |
| (Man, Or) | Generalisation | (Man, Or) | InStoreStaffPayment |
| Shift | (Man, Or) | Generalisation | (Man, Or) | DriverShift |
| (Man, Or) | Generalisation | (Man, Or) | InStoreStaffShift |
| DriverPayment | 0..\* | Pay | 1..1 | Driver |
| InStoretaffPayment | 0..\* | Pay | 1..1 | InStoreStaff |
| DriverShift | 0..\* | Shift | 1..1 | Driver |
| InStoreStaffShift | 0..\* | Shift | 1..1 | InStoreStaff |

# Attributes

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity** | **Attributes** | **Description** | **Data Type&Length** | **Nulls** | **Multi-valued** | **Derived** | **Default** |
| **Order**  **Order** | OrderNo | Unique order identifier | char | **N** | **N** | **N** |  |
| OrderDate&Time | The date&time the order is made | datetime | **Y** | **Y** | **N** |  |
| TotalAmountDue | Total money for this order | char | **Y** | **N** | **Y** |  |
| OrderType | Walk in or phone order | varchar | **Y** | **N** | **N** |  |
| Payment Method | How to pay this order | varchar | **N** | **N** | **N** |  |
| OrderStatus | Current order completion progress | varchar | **N** | **N** | **N** |  |
| ApprovalNo | a payment approval number | char | **N** | **N** | **N** |  |
| **MenuItem** | ItemNo | Unique dish identifier | char | **N** | **N** | **N** |  |
| Name | Dish name | varchar | **N** | **N** | **N** |  |
| Size | Describing the size of the dish | varchar | **N** | **Y** | **N** |  |
| CurrentSellingPrice | Price of the dish | char | **N** | **N** | **N** |  |
| Description | Info such as taste, ingredients | varchar | **Y** | **N** | **N** |  |
| **Customer** | CustomerID | Unique customer  identifier | char | **N** | **N** | **N** |  |
| FirstName | Customer's firstname | varchar | **Y** | **N** | **N** |  |
| LastName | Customer's lastname | varchar | **Y** | **N** | **N** |  |
| Address | Customer's address | varchar | **N** | **N** | **N** |  |
| Phone | Customer's phone  number | char | **N** | **Y** | **N** |  |
| Status | Customer's status | varchar | **N** | **N** | **N** |  |
| **Employee** | StaffNo | Unique staff  identifier | char | **N** | **N** | **N** |  |
| FirstName | Staff’s firstname | varchar | **Y** | **N** | **N** |  |
| LastName | Staff’s lastname | varchar | **Y** | **N** | **N** |  |
| Postal  Address | Staff’s address | varchar | **N** | **N** | **N** |  |
| Contact Number | Staff’s phone  number | char | **N** | **Y** | **N** |  |
| Tax File Number | Staff’s tax file number | char | **N** | **N** | **N** |  |
| Bank Details | Staff’s bank details | varchar | **N** | **N** | **N** |  |
| Payment Rate | Staff’s payment rate | varchar | **N** | **N** | **N** |  |
| DateOfBirth | Staff’s birthday | datetime | **Y** | **N** | **Y** |  |
| Status | Staff’s status | varchar | **N** | **N** | **N** |  |
| **Driver** | Driver’s license number | Driver’s license number | char | **N** | **N** | **N** |  |
| Description | Description about drivers | varchar | **Y** | **N** | **N** |  |
| **In Store Staff** | Description | Description about InStoreStaff | varchar | **Y** | **N** | **N** |  |
| HourlySalary | Hourly salary for InStoreStaff | char | **N** | **N** | **N** |  |
| **Ingredient** | Code | Unique ingredient  identifier | char | **N** | **N** | **N** |  |
| Name | Ingredient’s name | varchar | **N** | **N** | **N** |  |
| Type | Ingredient’s type | varchar | **N** | **N** | **N** |  |
| Description | Ingredient’s description | varchar | **Y** | **N** | **N** |  |
| StockLevelAtCurrentPeriod | Ingredient’s StockLevelAtCurrentPeriod | varchar | **N** | **N** | **N** |  |
| StockLevelAtLastPeriod | Ingredient’s StockLevelAtLastPeriod | varchar | **N** | **N** | **N** |  |
| SuggestedStockLevel | Ingredient’s SuggestedStockLevel | varchar | **N** | **N** | **N** |  |
| Reorder  Level | Ingredient’s Reorder  Level | varchar | **N** | **N** | **N** |  |
| Put in Time | Put in time of ingredients | datetime | **N** | **N** | **N** |  |
| Shelf Life | Ingredient’s Shelf Life | datetime | **N** | **N** | **Y** |  |
| **IngredientOrder** | OrderID | Unique order  identifier | char | **N** | **N** | **N** |  |
| Date of the order | Date of the order | datetime | **N** | **N** | **N** |  |
| Date received order | Date received order | datetime | **N** | **N** | **N** |  |
| TotalAmount | Total Amount of the order | char | **N** | **N** | **Y** |  |
| Status | Status of the order | varchar | **N** | **N** | **N** |  |
| Description | Description of the order | varchar | **Y** | **N** | **N** |  |
| **Supplier** | SupplierNo | Unique supplier  identifier | char | **N** | **N** | **N** |  |
| Name | Name of the supplier | varchar | **N** | **N** | **N** |  |
| Address | Supplier’s address | varchar | **N** | **N** | **N** |  |
| Phone | Supplier’s phone | char | **N** | **Y** | **N** |  |
| Email | Supplier’s email | varchar | **N** | **Y** | **N** |  |
| ContactPerson | Supplier’s contact person | varchar | **N** | **N** | **N** |  |
| **WalkInOrder** | pickUpTime | Time for customers to take meals | char | **N** | **N** | **N** |  |
| **Phone**  **Order** | description | Description for the order | varchar | **Y** | **N** | **N** |  |
| **Delivery** | deliveryTime | Driver delivery’s time | char | **N** | **N** | **N** |  |
| driverInfo | Information about the driver who delivered the order | varchar | **N** | **N** | **N** |  |
| **PickUp** | pickUpTime | Time for customers to take meals | char | **N** | **N** | **N** |  |
| **Payment** | paymentRecordID | Unique payment  identifier | char | **N** | **N** | **N** |  |
| grossPayment | Gross payment of the employee | varchar | **N** | **N** | **Y** |  |
| taxWithheld | Tax withheld of the employee | varchar | **N** | **N** | **Y** |  |
| TotalAmountPaid | Total  Amount pay for the staff | char | **N** | **N** | **Y** |  |
| paymentPeriodStartDate | Payment Period Start Date of the employee | datetime | **N** | **N** | **N** |  |
| paymentPeriodEndDate | Payment Period End Date of the employee | datetime | **N** | **N** | **N** |  |
| bankDetails | bank details of the employee | varchar | **N** | **N** | **N** |  |
| **DriverPayment** | deliveryQuantity | Numbers of delivery | char | Y | N | N |  |
| BonousForEachDelivery | Money for each delivery | char | Y | N | N |  |
| **InStoreStaffPayment** | numberOfWorkingHours | Number of working hours | char | Y | N | N |  |
| hourlyRate | Hourly rate of the In Store Staff | char | N | N | N |  |
| **Shift** | ShiftNo | Unique shiftNo  identifier | char | N | N | N |  |
| startDate | Shift’s start date | datetime | N | N | N |  |
| startTime | Shift’s start date | varchar | N | N | N |  |
| endDate | Shift’s end date | datetime | N | N | N |  |
| endTime | Shift’s end time | varchar | N | N | N |  |
| shiftType | Shift’s shift type | varchar | N | N | N |  |
| **DriverShift** | deliveryQuantity | Numbers of delivery | char | Y | N | N |  |
| deliveryRate | deliveryRate of the driver | char | N | N | N |  |
| **InStoreStaffShift** | hourlyRate | Hourly rate of the In Store Staff | char | N | N | N |  |
| numberOfWorkingHours | numberOfWorkingHours | char | N | N | N |  |

# Reflection for my assignment 1

There is a problem with my nulls in assessment1. The name options for customers and employees can be empty, and there is room for improvement in my table format. There is also a problem with the relationship between instore employees and instore payment. In this submission, I corrected the form and corrected the relationship of the EER model.

# Relational Model Schema

|  |  |
| --- | --- |
| **Order**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo)  **Primary Key** orderNo | **WalkInOrder**(orderNo, pickUpTime, customerID)  **Primary Key** orderNo  **Foreign Key** customerID **references Customer**(customerID)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** orderNo **references Order**(orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** |
| **PhoneOrder**(orderNo, description, customerID)  **Primary Key** orderNo  **Foreign Key** customerID **references Customer**(customerID)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** orderNo **references Order**(orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** | **DeliveryOrder**(orderNo, description, deliveryTime, driverInfo, staffNo)  **Primary Key** orderNo  **Foreign Key** staffNo **references Driver**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** orderNo **references PhoneOrder** (orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** |
| **PickupOrder**(orderNo, description, pickUpTime）  **Primary Key** orderNo  **Foreign Key** orderNo **references PhoneOrder** (orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** | **MenuItem**(itemNo, name, size, CurrentSellingPrice, description)  **Primary Key** itemNo  **Alternate Key** name |
| **QOrderMenuItem**(orderNo, itemNo, Quantity)  **Primary Key**(orderNo, itemNo)  **Foreign Key** orderNo **references Order**(orderNo)  **ON UPDATE CASCADE, ON DELETE CASCADE**  **Foreign Key** itemNo **references MenuItem**(itemNo)  **ON UPDATE CASCADE, ON DELETE CASCADE** | **Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time, shelf life)  **Primary Key** code |
| **QMenuItemIngredient**(itemNo, code，Quantity)  **Primary Key**(itemNo, code)  **Foreign Key** itemNo **references MenuItem**(itemNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** code **references Ingredient**(code)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **IngredientOrder**(orderID, dateOfTheOrder, recevedOrder, totalAmount, orderstatus, description, supplierNo)  **Primary Key** orderID  **Foreign Key** supplierNo **references** **Supplier**(supplierNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** |

|  |  |
| --- | --- |
| **QIngredientOrderIngredient**(orderID, code, Quantity)  **Primary Key**(orderID, code)  **Foreign Key** orderID **references** IngredientOrder(orderID)  **ON UPDATE CASCADE, ON DELETE CASCADE**  **Foreign Key** code **references** Ingredient(code)  **ON UPDATE CASCADE, ON DELETE CASCADE** | **Customer**(customerID, firstname, lastName, address, phoneNumber, status)  **Primary Key** customerID |
| **Supplier**(supplierNo, name, address, phone, email, contactPerson)  **Primary Key** supplierNo | **QSupplierIngredient**(supplierNo, code, Quantity)  **Primary Key** supplierNo, code  **Foreign Key** supplierNo **references Supplier**(supplierNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** code **references Ingredient**(code)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **Employee**(staffNo, firstname, lastName, postalAddress, contactNumber, taxFileNumber, bankDetails, paymentRate, dateOfBirth, status)  **Primary Key** staffNo | **Driver**(staffNo, Description, driver’s License Number)  **Primary Key** staffNo  **Foreign Key** staffNo **references Employee**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **In Store Staff**(staffNo, Description, hurlySalary)  **Primary Key** staffNo  **Foreign Key** staffNo **references Employee**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **Shift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType)  **Primary Key** ShiftNo |
| **DriverShift**(ShiftNo, deliveryQuantity, deliveryRate, staffNo)  **Primary Key** ShiftNo  **Foreign Key** staffNo **references Driver**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** ShiftNo **references Shift**(shiftNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **InStoreStaffShift**(ShiftNo，hourly rate , numberOfdelivers, staffNo)  **Primary Key** ShiftNo  **Foreign Key** staffNo **references InStoreStaff**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** ShiftNo **references Shift**(shiftNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **Payment**(paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails)  **Primary Key** paymentRecordID | **DriverPayment**(paymentRecordID,  deliveryQuantity, bonousForEachDelivery, staffNo)  **Primary Key** paymentRecordID  **Foreign Key** staffNo **references Driver**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** paymentRecordID **references Payment**(paymentRecordID)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **InStoreStaffPayment**(paymentRecordID,  numberOfWorkingHours, hourlyRate, staffNo)  **Primary Key** paymentRecordID  **Foreign Key** staffNo **references InStoreStaff**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** paymentRecordID **references Payment**(paymentRecordID)  **ON UPDATE CASCADE,ON DELETE CASCAD** |  |

# Normalizing the relational schema to Boyce-Codd Normal Form

**Order Relation**

**Order**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo)

**Primary Key** orderNo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| orderNo | orderDate&Time | totalAmountDue | orderType | paymentMethod | orderStatus | approvalNo |
| 001 | 11:30 20/15 | $45 | PhoneOrder | Cash | Receive | 2020 |
| 002 | 12:40 20/15 | $48 | WalkInOrder | Card | New | 2023 |

**orderNo** 🡪 orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo” completely rely on “orderNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**WalkInOrder Relation**

**WalkInOrder**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, pickUpTime, customerID)

**Primary Key** orderNo

**orderNo** 🡪 orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, pickUpTime, customerID)

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, pickUpTime, customerID” completely rely on “orderNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**PhoneOrder Relation**

**PhoneOrder**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, customerID)

**Primary Key** orderNo

**orderNo** 🡪 orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, customerID)

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, customerID” completely rely on “orderNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**Delivery Relation**

**Delivery**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, deliveryTime, driverInfo, staffNo)

**Primary Key** orderNo

**orderNo** 🡪 orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, deliveryTime, driverInfo, staffNo

All the attributes are fully functionally dependent on the PK

StaffNo 🡪 driverInfo partial dependency

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Order  No | Order  Date&Time | Total  Amount  Due | Order  Type | Payment  Method | Order  Status | Approval  No | description | Delivery  Time | Driver  Info | Staff  No |
| 001 | 16:30 20/15 | $49 | Phone | Cash | New | 2020 | ……. | 17:00 | Jack | 01 |
| 002 | 17:30 20/15 | $65 | Phone | Card | New | 2023 | ……. | 18:10 | Jack | 01 |

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, deliveryTime, driverInfo, staffNo” completely rely on “orderNo”.

**It’s not in 3rd Normal Form** because non-candidate-key attribute is transitively dependent on a candidate key, orderNo🡪staffNo, staffNo🡪driverInfo, from the example of the table, it can be seen that once the staffNo is the same, the driverInfo and the order type are the same, which generates a lot of data redundancy.

--------------------------------

**MenuItem Relation**

**MenuItem**(itemNo, name, size, CurrentSellingPrice, description)

itemNo 🡪 name, size, CurrentSellingPrice, description

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| itemNo | name | size | CurrentSellingPrice | description |
| 001 | King Pizza | Small | $11 | Pork and Beef inside |
| Medium | $16 |
| Large | $21 |

**It is not in 1th normal form** because of each MenuItem can have multiple sizes and different prices.

--------------------------------

**PickUp Relation**

**PickUp**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, pickUpTime）

**Primary Key** orderNo

**orderNo** 🡪 orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, pickUpTime

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, pickUpTime” completely rely on “orderNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**QOrderMenuItem Relation**

**QOrderMenuItem**(orderNo, itemNo, Quantity)

**Primary Key**(orderNo, itemNo)

orderNo, itemNo🡪 Quantity

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “Quantity” completely relies on “orderNo, itemNo”. There are no partial dependencies. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**Ingredient Relation**

**Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time, shelf life)

code🡪 name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time, shelf life

All the attributes are fully functionally dependent on the PK

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| code | name | type | description | stockLevel  At  Current  Period | StockLevel  At  Last  Period | Suggested  Stock  Level | reorder Level | put in time | shelf life |
| 003 | Chicken meat | Chicken | For pizza | 2Kg | 5Kg | 16Kg | 2Kg | 15/20 | 4Day |
| 004 | Chicken Paste | Chicken | For noodles | 3Kg | 5Kg | 11Kg | 3Kg | 15/20 | 4Day |

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th** **normal form** because “name”, “type”, “description”, “stockLevelAtCurrentPeriod”, “StockLevelAtLastPeriod”, “SuggestedStockLevel”, “reorder Level”, “put in time”, “shelf life” completely rely on “code”. There are no partial dependencies.

**Not conform the 3th normal form** because “shelf life” not only rely on “code”, but also rely on “type”. code🡪type, type🡪shelf life , code🡪shelf life. Non-candidate-key attribute is transitively dependent on a candidate key

--------------------------------

**QMenuItemIngredient Relation**

**QMenuItemIngredient**(itemNo, code，Quantity)

**Primary Key**(itemNo, code)

itemNo, code 🡪 Quantity

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “Quantity” completely relies on “itemNo, code”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**IngredientOrder Relation**

**IngredientOrder**(orderID, dateOfTheOrder, recevedOrder, totalAmount, status, description, supplierNo)

**Primary Key** orderID

orderID 🡪 dateOfTheOrder, recevedOrder, totalAmount, status, description, supplierNo

All the attributes are fully functionally dependent on the PK

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| orderID | dateOfTheOrder | recevedOrder | totalAmount | status | description | supplierNo |
| 001 | 16/08 | 18/08 | $400 | Receive | Pork | S01 |
| 002 | 17/08 | 18/08 | $260 | New | Fish | S02 |

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “dateOfTheOrder, recevedOrder, totalAmount, status, description, supplierNo” completely relies on “orderID”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**QIngredientOrderIngredient Relation**

**QIngredientOrderIngredient**(orderID, code, Quantity)

**Primary Key**(orderID, code)

orderID, code 🡪 Quantity

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “Quantity” completely relies on “orderID, code”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**Customer Relation**

**Customer**(customerID, firstname, lastName, address, phoneNumber, status)

**Primary Key** customerID

customerID 🡪 firstname, lastName, address, phoneNumber, status

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| customerID | firstname | lastName | address | phoneNumber | status |
| 101 | zed | Jack | 11 H street | 0501564980 | Waiting |
| 0311023369 |

**It is not in 1th normal form** because of each customer can have more than one phone number.

--------------------------------

**Supplier Relation**

**Supplier**(supplierNo, name, address, phone, email, contactPerson)

**Primary Key** supplierNo

supplierNo 🡪 name, address, phone, email, contactPerson

customerID 🡪 firstname, lastName, address, phoneNumber, status

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**It is not in 1th normal form** because of each supplier can have more than one phone number and email.

--------------------------------

**QSupplierIngredient Relation**

**QSupplierIngredient**(supplierNo, code, Quantity)

**Primary Key** supplierNo, code

supplierNo, code 🡪 Quantity

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “Quantity” completely relies on “supplierNo, code”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**Shift Relation**

**Shift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType)

**Primary Key** ShiftNo

ShiftNo🡪 StartDate, StartTime, endDate, endTime, shiftType

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “StartDate, StartTime, endDate, endTime, shiftType” completely relies on “ShiftNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**DriverShift Relation**

**DriverShift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType, deliveryQuantity, deliveryRate, ordersADriverDeliversDuringAShift, staffNo)

**Primary Key** ShiftNo

ShiftNo🡪 StartDate, StartTime, endDate, endTime, shiftType, deliveryQuantity, deliveryRate, ordersADriverDeliversDuringAShift, staffNo

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “StartDate, StartTime, endDate, endTime, shiftType, deliveryQuantity, deliveryRate, ordersADriverDeliversDuringAShift, staffNo” completely relies on “ShiftNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**InStoreStaffShift Relation**

**InStoreStaffShift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType, hourlyRate, numberOfWorkingHours, staffNo)

**Primary Key** ShiftNo

ShiftNo🡪 StartDate, StartTime, endDate, endTime, shiftType, hourlyRate, numberOfWorkingHours, staffNo

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “StartDate, StartTime, endDate, endTime, shiftType, hourlyRate, numberOfWorkingHours, staffNo” completely relies on “ShiftNo”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**Payment Relation**

**Payment**(paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails)

**Primary Key** paymentRecordID

paymentRecordID 🡪 grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails” completely relies on “paymentRecordID”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**DriverPayment Relation**

**DriverPayment**(paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails,

deliveryQuantity, moneyForEachDelivery, staffNo)

**Primary Key** paymentRecordID

paymentRecordID 🡪 grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails, deliveryQuantity, moneyForEachDelivery, staffNo

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails, deliveryQuantity, moneyForEachDelivery, staffNo” completely relies on “paymentRecordID”. There are no partial dependencies.

attribute is transitively dependent on a candidate key

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**InStoreStaffPayment Relation**

**InStoreStaffPayment**(paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails,

numberOfWorkingHours, hourlyRate, staffNo)

**Primary Key** paymentRecordID

paymentRecordID 🡪 paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails,

numberOfWorkingHours, hourlyRate, staffNo

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

**Conform the 1th** **normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th normal form** because “grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails, deliveryQuantity, moneyForEachDelivery, staffNo” completely relies on “paymentRecordID”. There are no partial dependencies.

**Conform the 3th normal form** because attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

**Conform the BCNF** because if and only if every functional dependency, X -> Y in R, X is a candidate key.

--------------------------------

**Customer Relation**

**Customer**(customerID, firstname, lastName, address, phoneNumber, status)

**Primary Key** customerID

--------------------------------

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| customerID | firstname | lastName | address | phoneNumber | status |
| 101 | zed | Jack | 11 H street | 0501564980 | Waiting |
| 0311023369 |

**It is not in 1th normal form** because of each customer can have more than one phone number.

Original Customer(with the phone as a multivalued attributes):

**Customer**(customerID, firstname, lastName, address, phoneNumber, status)

**Primary Key** customerID

--------------------------------

**Functional Dependencies:**

**X 🡪 Y**

**FD1:**customerID🡪 firstname, lastName, address, phoneNumber, status

**FD2:** customerID🡪 phoneNumber (phoneNumber is repeating group/not a single atomic value)

--------------------------------

**Normalization to 1NF:**

The phone fields is repeating groups that need to be separated so that different phone numbers in their individual fields.

**Customer**(customerID, firstname, lastName, address, status)

\* Remove phoneNumber from the original table

A new table will be made to store the phone and email values in:

**PhoneCustomer**(customerID, phoneNumber)

\* customerID , phoneNumber become a composite primary key.

\*customerID is also a foreign key that links back to the original **Customer** table.

The new schema (with the new tables with the redundancy removed):

**Customer**(customerID, firstname, lastName, address, status)

**Primary Key** customerID

**PhoneCustomer**(customerID，phoneNumber)

**Primary Key** customerID, phoneNumber

**Foreign Key** customerID **references customer** (customerID)

**ON UPDATE** CASCADE **ON DELETE** CASCADE

After creating the new table, the tables will now look like this (Every phone value now have their own record/row)

^^^

These now are FKs that reference Customer

**Customer**(customerID, firstname, lastName, address, status)

**Primary Key** customerID

I**t is already in 2NF because** “firstname”, “lastName”, “address” , “status” completely rely on “customerID”. Every non-candidate key attribute is fully functionally dependent on a candidate key.

I**t is already in 3NF because** attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

I**t is already in BCNF because** for every functional dependency X 🡪 Y, X is a candidate key (we have no non-candidate keys that determine any of the other attributes/candidate keys/Primary key)

**Supplier Relation**

**Supplier**(supplierNo, name, address, phone, email, contactPerson)

**Primary Key** supplierNo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| supplierNo | name | address | phone | email | contactPerson |
| A001 | Star | Heaton street | 0416042299 | 422352@wang.com | Jack |
| 0415032261 | 2423421@qs.com |

**It is not in 1th normal form** because of each supplier can have more than one phone number and email.

Original Supplier(with the phone and email as a multivalued attributes):

**Supplier**(supplierNo, name, address, phone, email, contactPerson)

**Primary Key** supplierNo

--------------------------------

**Functional Dependencies:**

**X 🡪 Y**

**FD1:** supplierNo 🡪 name, address, phone, email, contactPerson

**FD2:** supplierNo🡪 phone, email (phone and email are repeating group/not a single atomic value)

--------------------------------

**Normalization to 1NF:**

The phone and email fields are repeating groups that need to be separated so that different phone numbers and emails are in their individual fields.

**Supplier**(supplierNo, name, address, phone, email, contactPerson)

\* Remove phone and email from the original table

**Supplier**(supplierNo, name, address, contactPerson)

A new table will be made to store the phone and email values in:

**PhoneEmail**(supplierNo, phone, email)

\* supplierNo , phone and email become a composite primary key.

\* supplierNo is also a foreign key that links back to the original **Supplier** table.

The new schema (with the new tables with the redundancy removed):

**Supplier**(supplierNo, name, address, contactPerson)

**Primary Key** supplierNo

**PhoneEmail**(supplierNo, phone, email)

**Primary Key** supplierNo, phone email

**Foreign Key** supplierNo **references Supplier** (supplierNo)

**ON UPDATE** CASCADE **ON DELETE** CASCADE

After creating the new table, the tables will now look like this (Every phone value and email value now have their own record/row)

^^^

These now are FKs that reference Supplier

**Supplier**(supplierNo, name, address, contactPerson)

**Primary Key** supplierNo

I**t is already in 2NF because** “name”, “address”, “contactPerson” completely rely on “supplierNo”. Every non-candidate key attribute is fully functionally dependent on a candidate key.

I**t is already in 3NF because** attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

I**t is already in BCNF because** for every functional dependency X 🡪 Y, X is a candidate key (we have no non-candidate keys that determine any of the other attributes/candidate keys/Primary key)

**Ingredient Relation:**

**Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time, shelf life)

**Primary Key** code

It is **Conform the 1th normal form** because each attribute value is a single, atomic value from its domain.

**Conform the 2th** **normal form** because “name”, “type”, “description”, “stockLevelAtCurrentPeriod”, “StockLevelAtLastPeriod”, “SuggestedStockLevel”, “reorder Level”, “put in time”, “shelf life” completely rely on “code”.

--------------------------------

**Functional Dependencies:**

**FD1**: code -> name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time, shelf life

**X 🡪 Y**

When a non-candidate key determines another non-candidate key, then it is a transitive dependency. This is when a ‘regular’ attribute (that is not part of the key) determines another ‘regular’ attribute.

**FD2**: type -> shelf life

• type is a non-candidate key, it does not uniquely identify a record in the table.

• Every time it has a type appearing, It will always have the same shelf life. **Normalization to 3NF:**

Decomposing Ingredient Relation

**X 🡪 Y**

For FD2 **(**which was **–** type -> shelf life)

shelf life is redundant data, so it needs to be separated into a different table.

**X** = type

**Y** = shelf life

**XY** = type, shelf life

The original table/relation (R):

**Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time, shelf life)

\* Remove Y from the original table (shelf life)

Relation1 (R-Y) is:

**Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time)

Then, create a new table (Relation 2) to store XY in:

**IngredientType**(type, shelf life)

\* shelf life is normal attribute.

\* type becomes a Primary Key.

In the original table, code is still the Primary Key, and type becomes a foreign key that links to the Primary key of the new IngredientType table.

The new schema (with the new tables with the redundancy removed):

**IngredientType**(type, shelf life)

**Primary Key** type

**Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time)

**Primary Key** code

**Foreign Key** type **references** **IngredientType**(type)

**ON UPDATE** CASCADE **ON DELETE** NO ACTION

I**t is already in BCNF because** for every functional dependency X 🡪 Y, X is a candidate key (we have no non-candidate keys that determine any of the other attributes/candidate keys/Primary key)

**MenuItem Relation**

**MenuItem**(itemNo, name, size, CurrentSellingPrice, description)

itemNo 🡪 name, size, CurrentSellingPrice, description

X🡪Y, Y is functionally dependent on X, but not on any proper subset of X

All the attributes are fully functionally dependent on the PK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| itemNo | name | size | CurrentSellingPrice | description |
| 001 | king Pizza | Small | $11 | Pork and Beef inside |
| Medium | $16 |
| Large | $21 |

**It is not in 1th normal form** because of each MenuItem can have multiple sizes and different prices.

Original MenuItem(with the size and CurrentSellingPrice as a multivalued attributes):

**MenuItem**(itemNo, name, size, CurrentSellingPrice, description)

**Primary Key** itemNo

--------------------------------

**Functional Dependencies:**

**X 🡪 Y**

**FD1:** itemNo 🡪 name, size, CurrentSellingPrice, description

**FD2:** itemNo🡪 size, CurrentSellingPrice (size and CurrentSellingPrice are repeating group/not a single atomic value)

--------------------------------

**Normalization to 1NF:**

The size and CurrentSellingPrice fields are repeating groups that need to be separated so that different size and CurrentSellingPrice are in their individual fields.

**MenuItem**(itemNo, name, size, CurrentSellingPrice, description)

\* Remove size and CurrentSellingPrice from the original table

**MenuItem**(itemNo, name, description)

A new table will be made to store the size and CurrentSellingPrice values in:

**SizeCurrentSellingPrice**(itemNo, size, CurrentSellingPrice)

\* itemNo, size, CurrentSellingPrice become a composite primary key.

\* itemNo is also a foreign key that links back to the original **MenuItem** table.

The new schema (with the new tables with the redundancy removed):

**MenuItem**(itemNo, name, description)

**Primary Key** itemNo

**SizeCurrentSellingPrice**(itemNo, size, CurrentSellingPrice)

**Primary Key** itemNo, size, CurrentSellingPrice

**Foreign Key** itemNo **references MenuItem** (itemNo)

**ON UPDATE** CASCADE **ON DELETE** CASCADE

After creating the new table, the tables will now look like this (Every size and CurrentSellingPrice now have their own record/row)

**MenuItem**

|  |  |  |
| --- | --- | --- |
| **itemNo** | **name** | **description** |
| 001 | King Pizza | With beef, pork…. |

**SizeCurrentSellingPrice**

|  |  |  |
| --- | --- | --- |
| **itemNo** | **size** | **CurrentSellingPrice** |
| 001 | Small | $11 |
| 001 | Large | $16 |

**MenuItem**(itemNo, name, description)

**Primary Key** itemNo

I**t is already in 2NF because** “name”, “description” completely rely on “itemNo”. Every non-candidate key attribute is fully functionally dependent on a candidate key.

I**t is already in 3NF because** attributes not rely on other Non-primary attributes, no non-candidate-key attribute is transitively dependent on a candidate key.

I**t is already in BCNF because** for every functional dependency X 🡪 Y, X is a candidate key (we have no non-candidate keys that determine any of the other attributes/candidate keys/Primary key)

# the relations that are in BCNF

|  |  |
| --- | --- |
| **Order**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo)  **Primary Key** orderNo | **WalkInOrder**(orderNo, orderStatus, approvalNo, pickUpTime, customerID)  **Primary Key** orderNo  **Foreign Key** customerID **references Customer**(customerID)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** orderNo **references Order**(orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** |
| **PhoneOrder**(orderNo, description, customerID)  **Primary Key** orderNo  **Foreign Key** customerID **references Customer**(customerID)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** orderNo **references Order**(orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** | **Delivery**(orderNo, description, deliveryTime)  **Primary Key** orderNo  **Foreign Key** orderNo **references PhoneOrder** (orderNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** |
| **StaffNoDriverInfo**(orderNo, staffNo, driverInfo)  **Primary Key** orderNo, staffNo, driverInfo  **Foreign Key** staffNo **references Driver**(staffNo)  **ON UPDATE CASCADE,ON DELETE NO ACTION** | **PickUp**(orderNo, orderDate&Time, totalAmountDue, orderType, paymentMethod, orderStatus, approvalNo, description, pickUpTime）  **Primary Key** orderNo  **Foreign Key** orderNo **references PhoneOrder** (orderNo)  **ON UPDATE CASCADE,ON DELETE CASCADE** |
| **MenuItem**(itemNo, name, description)  **Primary Key** itemNo  **Alternate Key** name | **SizeCurrentSellingPrice**(itemNo, size, CurrentSellingPrice)  **Primary Key** itemNo, size, CurrentSellingPrice  **Foreign Key** itemNo **references MenuItem** (itemNo)  **ON UPDATE CASCADE ON DELETE CASCADE** |
| **QOrderMenuItem**(orderNo, itemNo, Quantity)  **Primary Key**(orderNo, itemNo)  **Foreign Key** orderNo **references Order**(orderNo)  **ON UPDATE CASCADE, ON DELETE CASCADE**  **Foreign Key** itemNo **references MenuItem**(itemNo)  **ON UPDATE CASCADE, ON DELETE CASCAD** | **Ingredient**(code, name, type, description, stockLevelAtCurrentPeriod, StockLevelAtLastPeriod, SuggestedStockLevel, reorder Level, put in time)  **Primary Key** code  **Foreign Key** type **references** **IngredientType**(type)  **ON UPDATE** CASCADE **ON DELETE** NO ACTION |
| **IngredientType**(type, shelf life)  **Primary Key** type | **QMenuItemIngredient**(itemNo, code，Quantity)  **Primary Key**(itemNo, code)  **Foreign Key** itemNo **references MenuItem**(itemNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** code **references Ingredient**(code)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **IngredientOrder**(orderID, dateOfTheOrder, recevedOrder, totalAmount, status, description, supplierNo)  **Primary Key** orderID  **Foreign Key** supplierNo **references** **Supplier**(supplierNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **QIngredientOrderIngredient**(orderID, code, Quantity)  **Primary Key**(orderID, code)  **Foreign Key** orderID **references** IngredientOrder(orderID)  **ON UPDATE CASCADE, ON DELETE CASCADE**  **Foreign Key** code **references** Ingredient(code)  **ON UPDATE CASCADE, ON DELETE CASCAD** |
| **Customer**(customerID, firstname, lastName, address, status)  **Primary Key** customerID | **PhoneCustomer(**customerID, phoneNumber)  **Primary Key** customerID, phoneNumber  **Foreign Key** customerID **references Customer**(customerID)  **ON UPDATE CASCADE, ON DELETE CASCAD** |
| **Supplier**(supplierNo, name, address, contactPerson)  **Primary Key** supplierNo  **Alternate Key** name | **PhoneEmail**(supplierNo, phone, email)  **Primary Key** supplierNo, phone email  **Foreign Key** supplierNo **references Supplier** (supplierNo)  **ON UPDATE** CASCADE **ON DELETE** CASCADE |
| **QSupplierIngredient**(supplierNo, code, Quantity)  **Primary Key** supplierNo, code  **Foreign Key** supplierNo **references Supplier**(supplierNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** code **references Ingredient**(code)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **Employee**(staffNo, firstname, lastName, postalAddress, taxFileNumber, bankDetails, paymentRate, dateOfBirth, status)  **Primary Key** staffNo |
| **EmployeeContactNo(**staffNo, contactNumber)  **Primary Key** staffNo, contactNumber  **Foreign Key** staffNo **references Employee**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **Driver**(staffNo, firstname, lastName, postalAddress, taxFileNumber, bankDetails, paymentRate, dateOfBirth, status, Description, driver’s License Number)  **Primary Key** staffNo  **Foreign Key** staffNo **references Employee**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **DriverContact (**staffNo, contactNumber)  **Primary Key** staffNo, contactNumber  **Foreign Key** staffNo **references Driver** (staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **DriverLicenseNo(**staffNo, driver’s License Number, dateOfBirth, firstname, lastName)  **Primary Key** staffNo driver’s License Number, dateOfBirth, firstname, lastName  **Foreign Key** staffNo **references Driver** (staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **In Store Staff**(staffNo, firstname, lastName, postalAddress, taxFileNumber, bankDetails, paymentRate, dateOfBirth, status, Description, hurlySalary)  **Primary Key** staffNo  **Foreign Key** staffNo **references Employee**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **InStoreStaffContact(**staffNo, contactNumber)  **Primary Key** staffNo, contactNumber  **Foreign Key** staffNo **references In Store Staff**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **Shift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType)  **Primary Key** ShiftNo |
| **DriverShift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType, deliveryQuantity, deliveryRate, ordersADriverDeliversDuringAShift, staffNo)  **Primary Key** ShiftNo  **Foreign Key** staffNo **references Driver**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** ShiftNo **references Shift**(shiftNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **InStoreStaffShift**(ShiftNo, StartDate, StartTime, endDate, endTime, shiftType, hourlyRate, numberOfWorkingHours, staffNo)  **Primary Key** ShiftNo  **Foreign Key** staffNo **references InStoreStaff**(staffNo)  **ON UPDATE CASCADE,ON DELETE CASCAD**  **Foreign Key** ShiftNo **references Shift**(shiftNo)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **Payment**(paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate, bankDetails)  **Primary Key** paymentRecordID | **DriverPayment**(paymentRecordID, paymentPeriodStartDate, paymentPeriodEndDate,  deliveryQuantity, moneyForEachDelivery)  **Primary Key** paymentRecordID  **Foreign Key** paymentRecordID **references Payment**(paymentRecordID)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **DriverPaymentStaff(**paymentRecordID, staffNo, bankDetails)  **Primary Key** paymentRecordID, staffNo, bankDetails  **Foreign Key** paymentRecordID **references DriverPayment** (paymentRecordID)  **ON UPDATE CASCADE,ON DELETE CASCAD** | **InStoreStaffPayment**(paymentRecordID, grossPayment, taxWithheld, totalAmountPaid, paymentDate, paymentPeriodStartDate, paymentPeriodEndDate,  numberOfWorkingHours)  **Primary Key** paymentRecordID  **Foreign Key** paymentRecordID **references Payment**(paymentRecordID)  **ON UPDATE CASCADE,ON DELETE CASCAD** |
| **InStoreStaffPaymentStaff(**paymentRecordID, staffNo, hourlyRate, bankDetails)  **Primary Key** paymentRecordID, staffNo, bankDetails  **Foreign Key** paymentRecordID **references InStoreStaffPayment** (paymentRecordID)  **ON UPDATE CASCADE,ON DELETE CASCAD** |  |