# Case Study

PAPAJON’S PIZZA is a pizza restaurant chain that offers “made to order” pizzas to its customers, i.e. customers choose the pizza base and the topping(s) they would like on their pizza(s). PAPAJON’S PIZZA offers “eat in” and “home delivery” service to its customers from a number of outlets nationwide.

The pizza bases that customers can choose from differ on their type and on their size. Currently PAPAJON’S PIZZA offers three types of pizza bases, namely “thin crust”, “deep pan”, and “stuffed crust”, which can be available in 5”, 9” and 15” size. Both the type of pizza base and size offered may change over time and area of operation, i.e. not all pizza bases/sizes combinations are available in every outlet. The combination of size and type of pizza base that an outlet offers depend on the area of operation and target customer market. PAPAJON’S PIZZA offers its customers a variety of toppings; again, not all toppings are available in each outlet. The price of a pizza is calculated based on the toppings and the pizza base used. Each of the pizza bases has a different price that depends on its size and type; the price of each topping depends on the size of the pizza base and the topping itself.

Although the PAPAJON’S PIZZA is a profitable company, the recent lockdowns and social distancing rules had a negative impact on the revenue the “eat in” part of the business has generated. On the other hand, there was an increase in the revenue of the “home delivery” service. The above has led the company’s management to believe that the revenue of the “home delivery” service can be increased further if the service is to be better organised. Among the changes the management wishes to introduce are (a) an online ordering service, where customers will be able to place (and track) collection, or delivery orders online; (b) customers to be able to place orders in advance, i.e. place orders and will be collected / delivered at a date and time agreed in advance; and (c) a new nationwide telephone number that customers can call to place their “home delivery” orders. The proposed service will operate are as follows:

Before a customer places a delivery order (online or through the call centre), s/he would be asked to provide details of the delivery address postcode to establish the outlet that will prepare and deliver the order, lead times and to check the availability of the various items ordered. For collection orders customers will need to provide details of the outlet they want to collect the order from. Customers will also be asked if they are registered, in which case their details will be retrieved from the customers’ database; otherwise, they will be asked to provide their name and a mobile contact number (to be used only in relation with their order). Customers who want to place an in advance will be asked to provide for the date and time they want their offer to be collected/delivered. To encourage customers to register their details, the company promises that their registered customers will experience an even speedier ordering service (details yet to be defined) and have extra benefits such delivery updates and credit vouchers for late orders.

Each outlet will serve a number of postcodes. For each placed order the information system should record the number of pizzas required; for each ordered pizza its type, its base size and choice of topping(s) will also be recorded. It should also record the delivery address (for non-registered customers, or if different from the registered one), the date/time the order has placed and the amount of the order. Customers, who place collection orders will be given a 10% collection discount. Finally, customers will be allowed to use any discount vouchers they may have for partial payment of their order (only one discount voucher can be allowed per order); in such case the discount voucher’s number will need to be entered to check its validity, before the discount is accepted. Payment details that will be recorded will include the credit/debit card number, cardholder’ name and address, expiry date, security number, card issue date and issue number for debit cards and authorisation number.

When the payment of an order is confirmed, the customer will be given an order reference and an estimated date/time of delivery/collection which cannot be more than 20 minutes late from the time quoted when the order has been confirmed; (b) the order will be passed on to the outlet that serves the particular delivery address for preparation. Registered customers and those with orders in advance will also receive an SMS (to their registered mobile number) that will contain an acknowledgement of their order, their order’s reference number, and (for delivery orders) the expected time of delivery. Registered customers will be able to track their order(s) online or through the call centre using their order number.

For their deliveries PAPAJON’S PIZZA will use a number of motorbikes (the registration number, and engine size of which are recorded) and will employ a number of motorbike drivers (the name, address, date of birth, and driving license number of each driver are recorded). Each delivery trip will involve a motorbike, a motorbike driver and a number of orders. The departure and arrival times of each delivery trip will be recorded, as well as the time at which each order is delivered. A motorbike driver will be able to use any available motorbike. Drivers will use an application to record the collection and delivery of orders. If an order is delivered 20 mins later than the quoted delivery time, then the customer will be sent by SMS a 15% voucher for their next order (the voucher’s number and expiry date need to be recorded). You are a member of the IT team working on the above information system and one of your responsibilities is the design of a data model that will allow processing and monitoring of customers’ orders and the monitoring of revenue and efficiency of service.

# Data Model for Papajon’s Online Delivery Case Study

## Entities and Attributes:

1. Entity name: OUTLET

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| outletId | INTEGER | PRIMARY KEY, NOT NULL |
| postcode | VARCHAR(8) | NOT NULL |
| name | VARCHAR(100) | NULL |

**name:** It refers to the name of an outlet.

1. Entity name: PIZZA

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| baseId | INTEGER | PRIMARY KEY, NOT NULL |
| pizzaBase | VARCHAR(50) | NOT NULL |
| size | VARCHAR(6) | NOT NULL |
| basePrice | DECIMAL(10, 2) | NOT NULL |

**basePrice:** It is the price for a combination of pizzaBase and size.

1. Entity name: TOPPING

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| toppingId | INTEGER | PRIMARY KEY, NOT NULL |
| name | VARCHAR(100) | NULL |
| toppingPrice | DECIMAL(10, 2) | NOT NULL |

**toppingPrice:** It is the decided price for each topping.

1. Entity name: ITEM

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| itemId | INTEGER | PRIMARY KEY, NOT NULL |
| quantity | INTEGER | NOT NULL |
| itemPrice | DECIMAL(10, 2) | NOT NULL |
| totalPrice | DECIMAL(10, 2) | NOT NULL |

**itemPrice:** It is the sum of **basePrice** and **toppingPrice** for each item.

**totalPrice:** It is the sum of (**itemPrice \* quantity**).

1. Entity name: ORDER

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| orderId | INTEGER | PRIMARY KEY, NOT NULL |
| delivery | BOOLEAN | NOT NULL |
| inAdvance | BOOLEAN | NOT NULL |
| orderDateTime | DATETIME | NOT NULL |
| leadTime | DATETIME | NULL |
| advanceDateTime | DATETIME | NULL |
| deliveryAddress | VARCHAR(100) | NULL |
| collectionAddress | VARCHAR(100) | NULL |
| totalAmount | DECIMAL(10, 2) | NOT NULL |
| finalAmount | DECIMAL(10, 2) | NOT NULL |

**delivery:** ‘Yes’ means it is a delivery order and ‘No’ means it is a collection order.

**inAdvance:** ‘Yes’ means order is for advance booking an ‘No’ means it is current/real-time order.

**orderDateTime**: it is the time when order is confirmed.

**leadTime**: It is the expected time for order delivery at customer’s door. It is assumed to be calculated based on orderDateTime, estimated time for preparing order, distance between outlet and customer postcode/ address etcetera.

**advanceDateTime:** it will only be filled if inAdvance is ‘yes’.

**deliveryAddress:** It will only be filled if it’s a delivery order.

**collectionAddress:** It will only be filled if it’s a collection order.

**totalAmount:** It represents the overall price of an order, encompassing the cost of all ordered items.

**finalAmount:** It will be the new amount (subtracting discounted amount from totalAmount) for an order if any voucher is applied. Otherwise, it will be same as of totalAmount.

1. Entity name: VOUCHER

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| voucherId | INTEGER | PRIMARY KEY, NOT NULL |
| expiryDateTime | DATETIME | NOT NULL |
| valid | BOOLEAN | NOT NULL |
| discountPercentage | INTEGER | NOT NULL |

**valid:** ‘Yes’ means a voucher is valid and ‘No’ means it is invalid. It will be checked based on expiryDateTime. Also, once the voucherId is used in any order this table will be updated to mark valid as ‘No’.

**discountPercentage:** It will mention the percentage of a voucher like 10, 20, 50 etcetera. Based on these values calculations will be applied in query to calculate total amount for an order.

1. Entity name: CUSTOMER

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| customerId | INTEGER | PRIMARY KEY, NOT NULL |
| name | VARCHAR(100) | NOT NULL |
| mobileNumber | VARCHAR(15) | NOT NULL |
| registrationStatus | BOOLEAN | NOT NULL |

1. Entity name: PAYMENT

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| paymentId | INTEGER | PRIMARY KEY, NOT NULL |
| cardType | VARCHAR(50) | NOT NULL |
| cardNumber | INTEGER | NOT NULL |
| cardholderName | VARCHAR(100) | NOT NULL |
| cardholderAddress | VARCHAR(100) | NULL |
| expiryDate | DATETIME | NOT NULL |
| securityNumber | INTEGER | NOT NULL |
| cardIssueDate | DATETIME | NOT NULL |
| cardIssueNumber | INTEGER | NULL |
| authorisationNumber | INTEGER | NULL |

1. Entity name: DELIVERY

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| deliveryId | INTEGER | NOT NULL |
| arrivalTime | DATETIME | NOT NULL |
| departureTime | DATETIME | NOT NULL |
| numberOfOrders | INTEGER | NOT NULL |
| deliveryTime | DATETIME | NOT NULL |

**arrivalTime:** When driver reaches at the outlet to collect delivery order.

**departureTime:** When he starts the journey to deliver the order at customer’s door.

**deliveryTime:** When delivery is completed.

1. Entity name: DRIVER

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| driverId | INTEGER | PRIMARY KEY, NOT NULL |
| name | VARCHAR(100) | NOT NULL |
| address | VARCHAR(100) | NULL |
| dateOfBirth | DATE | NULL |
| licenseNumber | VARCHAR(50) | NOT NULL |

1. Entity name: DELIVERYVEHICLE

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraints** |
| registrationNumber | INTEGER | PRIMARY KEY, NOT NULL |
| engineSize | INTEGER | NOT NULL |

## Relationship Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Entities** | **Multiplicities** | **Definition** |
| contains | Ternary | OUTLET – PIZZA | One to Many | For each outlet, there is at least one or many (pizza/size) combinations in stock. |
| contains | Ternary | PIZZA – OUTLET | Zero to Many | A specific pizza/size (combination) might not be available at any outlet or different combinations are available across all the outlets. |
| contains | Ternary | OUTLET – TOPPING | One to Many | For each outlet, there is at least one or many toppings in stock. |
| contains | Ternary | TOPPING - OUTLET | Zero to Many | A specific topping might not be available at any outlet or different are available across all the outlets. |
| has | Binary | PIZZA – TOPPING | Zero to Many | A pizza may do not have any topping or multiple topping on it. |
| has | Binary | TOPPING – PIZZA | Zero to Many | A topping may have not been used on any pizza. Or different toppings are used on multiple pizzas. |
| utilize | Binary | PIZZA – ITEM | One to One | Only one pizza/size (combination) can be utilized for an item. |
| utilize | Binary | ITEM – PIZZA | One to One | For an item, only one pizza/size (combination) can be utilized. |
| includes | Binary | ITEM – ORDER | Zero to Many | An item might not be included in any of the orders, or an item is included in multiple orders. |
| includes | Binary | ORDER – ITEM | One to Many | At least one item should be included in an order, however there can be multiple items included in an order. |
| process | Binary | OUTLET – ORDER | Zero to Many | Every outlet can process zero or multiple orders. |
| process | Binary | ORDER - OUTLET | One to One | An order will be processed by just one outlet. |
| places | Binary | ORDER – CUSTOMER | One to One | An order will only be placed by a single customer. |
| places | Binary | CUSTOMER – ORDER | One to Many | A customer can be one-time or regular. |
| transaction | Binary | ORDER – PAYMENT | One to One | For each order one payment much be placed. |
| transaction | Binary | PAYMENT – ORDER | One to One | Only one payment can be placed for an order. |
| applies | Binary | ORDER – VOUCHER | Zero to Many | For an order either zero, one or many vouchers can be applied. |
| applies | Binary | VOUCHER – ORDER | Zero to One | Either a voucher will not be applied to any order or at most it can be applied to one order. |
| assigns | Binary | ORDER – DELIVERY | One to One | For an order only one delivery can be assigned. |
| assigns | Binary | DELIVERY – ORDER | One to Many | For a delivery trip at least one or multiple orders can be assigned. |
| picks | Binary | DELIVERY – DRIVER | One to One | A delivery can be picked by only one driver. |
| picks | Binary | DRIVER – DELIVERY | Zero to many | A drive may not pick any of the deliveries or he can pick multiple deliveries. |
| uses | Binary | DRIVER – DELIVERYVECHICLE | One to One | A driver will only use one vehicle for a delivery. |
| uses | Binary | DELIVERYVECHICLE – DRIVER | Zero to Many | A delivery vehicle may not be used by any of the drivers, or it can be used by many drivers. |

## Enhanced Entity Relationship Diagram (Conceptual E-ERD)

**A diagram of a company

Description automatically generated**

**Figure 1.1** Conceptual E-ERD(Song, Evans and Park, 1995)

## Logical Entity Relationship Diagram

A diagram of a computer program

Description automatically generated with medium confidence

**Figure 1.2** Logical E-ERD(Song, Evans and Park, 1995)

## Working Details

A postcode provided by the customer will be kept temporary rather than storing in any database. Based on that postcode outlet table will be searched to decide on the nearest outlet based on the distance between customer’s provided postcode and outlet postcode. Also based on that distance and some other factors like traffic and cooking time etcetera lead time will be calculated and stored temporarily until the order is confirmed. Once the order is confirmed the lead time will be updated in the order table. Simultaneously, using both foreign keys (baseId and toppingId) both PIZZA and TOPPING tables will be searched for available items based on filtered outlet postcode.

If chosen items are available, customer is satisfied with the leadtime/waiting time and he is ready to move forward with the order, then he will be asked for the quantity of each chosen item to decide on price. The calculated price for each item will be stored in itemPrice corresponding to each item. Further it will be multiplied with quantity for each item. Minimum quantity will be one for each chosen item. Further totalPrice of all the items for an order will be the sum of all the itemPrice multiplied with the quantity for each item. Working is shown in Below.

A yellow and black label with black text

Description automatically generated

**Case 1.** If it is a current delivery order(in this case delivery will be ‘Yes’ and inAdvance will be ‘No’) then full deliveryAddress will be filled along with the postcode. In this case advanceDateTime and collectionAddress will be null. The totalAmount for an order will be fetched from ITEM table using itemId.

A yellow and black tag with black text

Description automatically generated

If any voucher is applied to an order the totalAmount will be subtracted by the discounted amount (this subtraction will be done by query), and calculated amount will be store in finalAmount. Simultaneously, the voucherId used for this order will be changed to invalid (using query).

**Case 2.** If it is a current collection order(in this case both delivery and inAdvance will be ‘No’) then collectionAddress will be updated with the serving outlet name and postcode. In this case advanceDateTime and deliveryAddress will be null. Rest of the process will be repeated. However, as it is a collection order totalAmount will be discounted by 10% (using query) along with any other discount vouchers if applicable. (**Assumption)** Or else, if a customer does not want to redeem the 10% collection discount, a discount voucher of 10% will be shared with them using any mode of communication like( SMS, Customer Profile, Outlet application etcetera). Simultaneously, VOUCHER table will be updated with all the details about newly distributed voucher. It is obvious, delivery trip will not be initiated for this kind of order.

**Case 3.** If it is an inAdvance delivery order, then collectionAddress will be null. The advanceDateTime will be the agreed date and time for the delivery. And, leadTime will be updated considering the advanceDateTime. Further, delivery trip will be organized as per the updated times.

**Case 4.** If it is an inAdvance collection order, then deliveryAddress will be null. The advanceDateTime will be the agreed date and time for the collection. The same way as Case 2. Voucher procedure will be repeated if applicable.

Once the payment is completed for the finalAmount, a delivery trip will be organized. For executing a delivery trip both (deliveryId and orderId) are used as a Primary Key. This is done to maintain the record of deliveryTime for each order served in a single delivery trip.

A yellow and black sign with black text

Description automatically generated

**Assumption:** leadTime and deliveryTime is used to track if the order is delivered late.

A yellow and black box with black text

Description automatically generated

Now, it can be noticed that order having orderId 115 is delivered more than 20 minutes late than the expected leadTime. In this case, a voucher of 15% will be sent to the customer by SMS also the share voucher will be updated in the table with required details.

**Assumption:**

* To ensure all the registered and new registered customers get faster delivery, delivery vehicle with big engineSize will be used. The bigger the engine size, the more power a vehicle driver has.
* The available stocks for each location and its price will be updated by the management regularly in PIZZA and TOPPING tables.
* orderDateTime can be used to track and validate the trigger to send a message (order number and estimated date time of delivery/collection) to the customer within 20 minutes of order confirmation after payment.

## Relational Schema:

## List of Relations:

**OUTLET** (outletId, baseId, toppingId, postcode, name)

**PIZZA** (baseId, toppingId, pizzaBase, size, basePrice)

**TOPPING** (toppingId, name, toppingPrice)

**ITEM** (itemId, baseId, quantity, ItemPrice, totalPrice)

**ORDER** (orderId, outletId, itemId, customerId, paymentId, voucherId, deliveryId, delivery, inAdvance, orderDateTime, leadTime, advanceDateTime, deliveryAddress, collectionAddress, totalAmount, finalAmount)

**VOUCHER** (voucherId, expiryDateTime, valid, discountPercentage)

**CUSTOMER** (customerId, name, mobileNumber, registrationStatus)

**PAYMENT** (paymentId, cardType, cardNumber, cardholderName, cardholderAddress, expiryDate, securityNumber, cardIssueDate, cardIssueNumber, authorisationNumber)

**DELIVERYVECHICLE** (registrationNumber, engineSize)

**DRIVER** (driverId, registrationNumber, name, address, dateOfBirth, licenseNumber)

**DELIVERY** (deliveryId, orderId, driverId, arrivalTime, departureTime, numberOfOrders, deliveryTime)

## List of Primary and Foreign Keys:

|  |  |  |
| --- | --- | --- |
| **Entity Name** | **Primary Key** | **Foreign Key** |
| OUTLET | outletId | baseId  toppingId |
| PIZZA | baseId | toppingId |
| TOPPING | toppingId | NA |
| ITEM | itemId | baseId |
| ORDER | orderId | outletId  itemId  customerId  paymentId  voucherId  deliveryId |
| PAYMENT | paymentId | NA |
| CUSTOMER | customerId | NA |
| VOUCHER | voucherId | NA |
| DELIVERY | deliveryId  orderId | driverId |
| DRIVER | driverId | registrationNumber |
| DELIVERYVEHICLE | registrationNumber | NA |