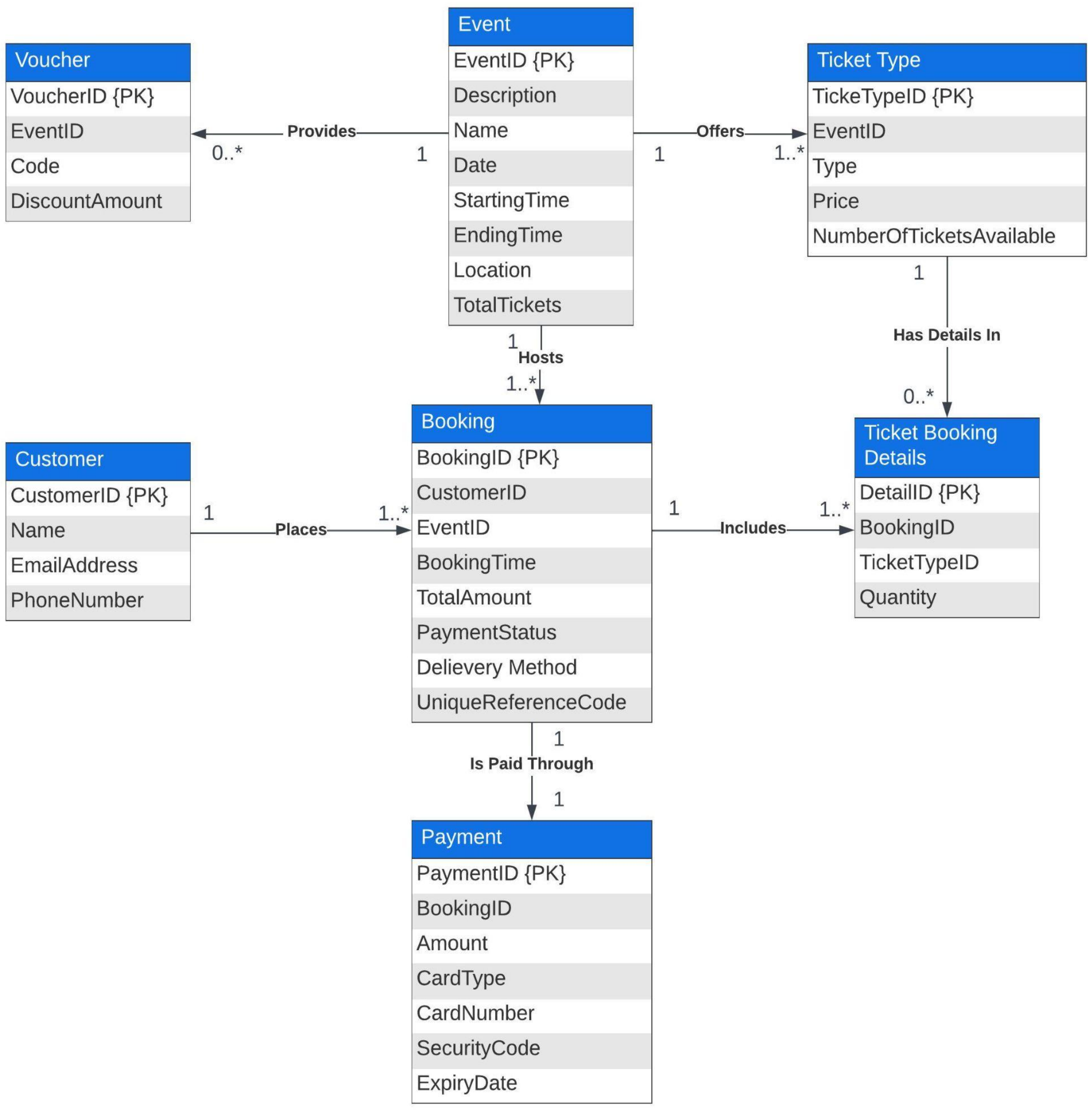
Report for Ticket Booking Database system



1. Conceptual Model Design

Customer:

Description of Attributes:

- *CustomerID {PK}* - a unique identifier for each customer.
- *Name* - The Full Name of the customer.
- *EmailAddress* - Email address of the customer.
- *PhoneNumber* - Phone number of the customer.

It was assumed that Customer Entity would have a unique number such as CustomerID, which is the Primary Key to identify customers in the system. Attributes such as Name, EmailAddress, PhoneNumber are used for customer management, such as collecting tickets via email if chosen or future marketing communications.

Relationships:

1. The Customer-Booking relationship is identified as One-to-Many. It means that each customer can make at least one or several bookings, reflecting a real-world scenario where customers can make several bookings for different events.

Event:

Description of Attributes:

- *EventID {PK}* - a unique identifier for each event.
- *Description* - A brief overview of the event.
- *Name* - The name of the event.
- *Date* - The day in which the event takes place.
- *StartingTime* and *EndingTime* - The times when the event starts and finishes.
- *Location* - The venue of the event.
- *TotalTickets* - The total number of tickets available for the event.

It was assumed that Event Entity would have a unique number such as EventID, which is Primary Key to identify events in the system. Attributes such as Description, Name, Date, StartingTime, EndingTime and Location are used for event management and retrieving general information about the event. Each event has a limited number of tickets which are demonstrated in the TotalTickets attribute.

Relationships:

1. The Event-Booking relationship is a One-to-Many relationship which indicates that each event can have multiple bookings associated with a specific customer, however a booking can correspond to only one event.
2. The Event-TicketType relationship is a One-to-Many relationship which indicates that each event can have various ticket types, however one ticket type corresponds to only one event.
3. The Event-Voucher relationship is a One-to-Many relationship which indicates that each event can have various vouchers, however each voucher corresponds to only one event.

Booking:

Description of Attributes:

- *BookingID {PK}* - A unique identifier for each booking.
- *CustomerID* - A reference to the customer who placed a booking.
- *EventID* - A reference to the specific event which is booked.
- *BookingTime* - The time of when the booking was placed.

- *TotalAmount* - The total cost of the booking.
- *PaymentStatus* - The current status of the booking's payment. Statuses: Paid, Pending, Cancelled.
- *Delivery Method* - The status of the delivery of tickets to the customer. Statuses: Email, Pickup.
- *UniqueReferenceCode* - A unique code for reference and verification purposes.

It was assumed that each booking has a unique identifier BookingID, which is the Primary Key to manage bookings in the system. Attributes such as CustomerID, EventID, BookingTime, TotalAmount, PaymentStatus, Delivery Method and UniqueReferenceCode are used for booking management and retrieving information about the Booking.

Relationships:

1. The Booking-TicketBookingDetails relationship is a One-to-Many relationship allowing to collect detailed information about one or more ticket types within one booking.
2. The Booking-Payment relationship is a One-to-One relationship indicating that one booking results in one payment transaction.

<u>TicketType:</u>

Description of Attributes:

- *TicketTypeID {PK}* - a unique identifier for each ticket type.
- *EventID* - The reference to the event to which the type of ticket belongs.
- *Type* - The type of the ticket. Types: Adult, Child, Gold, Silver, Bronze.
- *Price* - The cost of the ticket.
- *NumberOfTicketsAvailable* - The amount of the type of ticket available to the event.

It was assumed that the TicketType entity has a unique identifier TicketTypeID, which is the Primary Key of the entity. Attributes such as EventID, Type, Price, NumberOfTicketsAvailable are crucial pieces of information to manage ticket types in the system, optimise ticket sales, and check the availability of the tickets.

Relationships:

1. The TicketType-TicketBookingDetails is a One-to-Many relationship, where each ticket type can have multiple booking details but each booking detail corresponds to only one ticket type.

<u>Payment:</u>

Description of Attributes:

- *PaymentID {PK}* - a unique identifier for each payment.
- *BookingID* - Connects the payment to a specific booking.
- *Amount* - The total amount paid.

- *CardType* - The type of card used. Types: Visa, MasterCard.
- *CardNumber* - The number of a credit/debit card.
- *SecurityCode* - The code of the card for transaction verification.
- *ExpiryDate* - The expiry date of the card.

It was assumed that Payment Entity has a unique identifier PaymentID, which is the Primary Key of the entity. Attributes such as BookingID, Amount, CardType, CardNumber, SecurityCode, ExpiryData are used for payment management, ensuring security of transactions and efficient processing.

Relationships:

1. The Booking-Payment relationship is a One-to-One relationship indicating that one booking results in one payment transaction.

Voucher:

Description of Attributes:

- *VoucherID {PK}* - a unique identifier for each voucher.
- *EventID* - Reference the voucher that applies to a specific event.
- *Code* - The unique voucher code for discounts.
- *DiscountAmount* - The amount of discount provided by the voucher.

It was assumed that Voucher Entity has a unique identifier VoucherID which is the Primary Key of the entity. Attributes such as EventID, Code, DiscountAmount are used to manage promotional and discount codes that customers can use to have discounts on bookings of a specific event that is crucial for marketing strategies in the real-world scenario.

Relationships:

1. The Event-Voucher relationship is a One-to-Many relationship which indicates that each event can have various vouchers, however each voucher corresponds to only one event.

TicketBookingDetails:

Description of Attributes:

- *DetailID {PK}* - A unique identifier for booking details of each ticket.
- *BookingID* - References to the booking to which details belong.
- *TicketTypeID* - References to the type of ticket booked.
- *Quantity* - The number of tickets of each type in the booking.

It was assumed that TicketBookingDetails has a unique identifier DetailID which is the primary key of the entity. Attributes such as BookingID, TicketTypeID and Quantity are crucial for detailed information about the booking. The entity is useful for events that have multiple types of tickets such as Gold, Silver, Bronze or Adult, Child. It ensures that the ticket availability for each type of ticket is updated.

Relationships:

1. The Booking-TicketBookingDetails relationship is a One-to-Many relationship allowing to collect detailed information about one or more ticket types within one booking.
2. The TicketType-TicketBookingDetails is a One-to-Many relationship, where each ticket type can have multiple booking details but each booking detail corresponds to only one ticket type.

2. Logical Model Design

- **Customers** (CustomerID, Name, EmailAddress, PhoneNumber)
  Primary Key: CustomerID
- **Event** (EventID, Description, Name, Date, StartingTime, EndingTime, Location, TotalTickets)
  Primary Key: EventID
- **Booking** (BookingID, CustomerID, EventID, BookingTime, TotalAmount, PaymentStatus, DeliveryMethod, UniqueReferenceCode)
  Primary Key: Booking ID
  Foreign Key: CustomerID references Customer(CustomerID), EventID references Event(EventID)
- **TicketType** (TicketTypeID, EventID, Type, Price, NumberOfTicketsAvailable)
  Primary Key: TicketTypeID
  Foreign Key: EventID references Event(EventID)
- **Payment** (PaymentID, BookingID, Amount, CardType, CardNumber, SecurityCode, ExpiryDate)
  Primary Key: PaymentID
  Foreign Key: BookingID references Booking(BookingID)
- **Voucher** (VoucherID, EventID, Code, DiscountAmount)
  Primary Key: VoucherID
  Foreign Key: EventID references Event(EventID)
- **TicketBookingDetails** (DetailID, BookingID, TicketTypeID, Quantity)
  Primary Key: DetailID
  Foreign Key: BookingID references Booking(BookingID), TicketTypeID references TicketType(TicketTypeID)