Project Report
Big Data Management Analytics

ERD - Ticketing Database Management

Submitted to: Prof. Amarnath Mitra
FORE School of Management, New Delhi

Submitted by: Jagriti (055016)
Harsh Jain (055018

B-18, Qutub Institutional Area
New Delhi – 110016

Table of Contents

Entity-Relationship Diagram - Ticketing Database Management

The ERD describes the database structure for a Ticketing Database Management, including entities, their attributes, primary keys (PK), foreign keys (FK), and relationships between tables.

(A) Schema for the chosen database:

The database schema is designed to efficiently manage event venues, ticketing, customer interactions, and payments. The schema consists of multiple tables, each serving a specific function within the event management system. It includes primary keys (PK) for unique identification, foreign keys (FK) to maintain relationships, and constraints to ensure data integrity.

- **Venues**: Stores venue details such as name, address, and capacity.
- **Events**: Manages events, linking them to venues.
- **Tickets**: Tracks ticket information, including price and type.
- **Customers**: Maintains customer information with unique email identifiers.
- **Orders**: Records customer orders for tickets.
- **OrderTickets**: Links tickets to orders.
- **Payments**: Stores payment details for orders.
- **Promotions**: Manages event promotions and discounts.
- **EventStaff**: Tracks staff assigned to specific events.
- **Feedback**: Records customer reviews and ratings.
- **Waitlist**: Maintains a waitlist for events.
- **TicketAvailability**: Tracks ticket availability per event.

This schema ensures seamless integration between event organization, ticket sales, customer management, and financial transactions.

Schema Constraints and Optimizations

- **Primary Keys (PK)**: VenueID, EventID, TicketID, CustomerID, OrderID, OrderTicketID, PaymentID, PromotionID, StaffID, FeedbackID, WaitlistID, AvailabilityID
- **Foreign Keys (FK)**:
  - Events references Venues
  - Tickets references Events
  - Orders references Customers
  - OrderTickets references Orders and Tickets
  - Payments references Orders
  - Promotions references Events
  - EventStaff references Events
  - Feedback references Customers and Events
  - Waitlist references Customers and Events
  - TicketAvailability references Events
- **Unique Constraints**:
  - Email in Customers (to ensure unique user accounts)

- **Indexes**:
    - Index on VenueID in Events for efficient event lookup
    - Index on EventID in Tickets for quick retrieval of tickets
    - Index on CustomerID in Orders for customer order tracking
    - Index on OrderID in Payments for order-payment linkage

Data Relationships and Indexing

- One-to-Many (1:N): Venues → Events, Events → Tickets, Customers → Orders, Orders → Payments
- Many-to-Many (M:M): Orders ↔ Tickets (via OrderTickets), Customers ↔ Events (via Waitlist)
- Indexes on primary keys for faster retrieval

(B) Entities and Attributes

1. **Venues Table**
    - VenueID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - VenueName (VARCHAR(255))
    - Address (VARCHAR(255))
    - Capacity (INT)

2. **Events Table**
    - EventID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - EventName (VARCHAR(255))
    - EventDate (DATETIME)
    - Description (TEXT)
    - VenueID (INT, FOREIGN KEY referencing Venues(VenueID))

3. **Tickets Table**
    - TicketID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - EventID (INT, FOREIGN KEY referencing Events(EventID))
    - Price (DECIMAL(10,2))
    - SeatNumber (VARCHAR(20))
    - TicketType (VARCHAR(50))

4. **Customers Table**
    - CustomerID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - FirstName (VARCHAR(255))
    - LastName (VARCHAR(255))
    - Email (VARCHAR(255), UNIQUE)
    - PhoneNumber (VARCHAR(20))

5. **Orders Table**
    - OrderID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - CustomerID (INT, FOREIGN KEY referencing Customers(CustomerID))
    - OrderDate (DATETIME, DEFAULT CURRENT_TIMESTAMP)
    - TotalAmount (DECIMAL(10,2))

6. **OrderTickets Table**
    - OrderTicketID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - OrderID (INT, FOREIGN KEY referencing Orders(OrderID))
    - TicketID (INT, FOREIGN KEY referencing Tickets(TicketID))
    - Quantity (INT)

7. **Payments Table**
    - PaymentID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - OrderID (INT, FOREIGN KEY referencing Orders(OrderID))
    - PaymentDate (DATETIME, DEFAULT CURRENT_TIMESTAMP)
    - PaymentMethod (VARCHAR(50))
    - Amount (DECIMAL(10,2))

8. **Promotions Table**
    - PromotionID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - PromotionName (VARCHAR(255))
    - DiscountPercentage (DECIMAL(5,2))
    - StartDate (DATE)
    - EndDate (DATE)
    - EventID (INT, FOREIGN KEY referencing Events(EventID))

9. **EventStaff Table**
    - StaffID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - FirstName (VARCHAR(255))
    - LastName (VARCHAR(255))

- Role (VARCHAR(100))
- EventID (INT, FOREIGN KEY referencing Events(EventID))

10. **Feedback Table**
    - FeedbackID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - CustomerID (INT, FOREIGN KEY referencing Customers(CustomerID))
    - EventID (INT, FOREIGN KEY referencing Events(EventID))
    - Rating (INT)
    - Comment (TEXT)
    - FeedbackDate (DATETIME, DEFAULT CURRENT_TIMESTAMP)

11. **Waitlist Table**
    - WaitlistID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - CustomerID (INT, FOREIGN KEY referencing Customers(CustomerID))
    - EventID (INT, FOREIGN KEY referencing Events(EventID))
    - WaitlistDate (DATETIME, DEFAULT CURRENT_TIMESTAMP)

12. **TicketAvailability Table**
    - AvailabilityID (INT, PRIMARY KEY, AUTO_INCREMENT)
    - EventID (INT, FOREIGN KEY referencing Events(EventID))
    - TicketType (VARCHAR(50))
    - AvailableTickets (INT)

(C) Relationship between Tables

1. Venues and Events:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Venues: 1 (Each event must be held at one venue)
    - Events: 0..* (A venue can host zero or many events)
- **Description:** Each event is associated with one specific venue. One venue can host multiple events. The VenueID in the Events table is a foreign key referencing the VenueID in the Venues table.

2. Events and Tickets:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Events: 1 (Each ticket is for one specific event)
    - Tickets: 0..* (An event can have zero or many tickets)
- **Description:** Each ticket is associated with one specific event. One event can have multiple tickets. The EventID in the Tickets table is a foreign key referencing the EventID in the Events table.

3. Customers and Orders:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Customers: 1 (Each order is placed by one customer)
    - Orders: 0..* (A customer can place zero or many orders)
- **Description:** Each order is associated with one specific customer. One customer can place multiple orders. The CustomerID in the Orders table is a foreign key referencing the CustomerID in the Customers table.

4. Orders and OrderTickets:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Orders: 1 (Each OrderTicket record is part of one order)
    - OrderTickets: 0..* (An order can contain zero or many OrderTickets)
- **Description:** Each OrderTicket record is associated with one specific order. One order can contain multiple OrderTicket records. The OrderID in the OrderTickets table is a foreign key referencing the OrderID in the Orders table.

5. Tickets and OrderTickets:

- **Relationship:** One-to-many
- **Cardinality:** 1:N

    - Tickets: 1 (Each OrderTicket record is for one ticket)
    - OrderTickets: 0..* (A ticket can be part of zero or many OrderTickets)
- **Description:** Each OrderTicket record is associated with one specific ticket. One ticket can be part of multiple orders. The TicketID in the OrderTickets table is a foreign key referencing the TicketID in the Tickets table.

6. Orders and Payments:

- **Relationship:** One-to-one
- **Cardinality:** 1:1
    - Orders: 1 (Each order has one payment)
    - Payments: 1 (Each payment is for one order)
- **Description:** Each payment is associated with one specific order. One order has one payment. The OrderID in the Payments table is a foreign key referencing the OrderID in the Orders table.

7. Events and Promotions:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Events: 1 (Each promotion is for one event)
    - Promotions: 0..* (An event can have zero or many promotions)
- **Description:** Each promotion is associated with one specific event. One event can have multiple promotions. The EventID in the Promotions table is a foreign key referencing the EventID in the Events table.

8. Events and EventStaff:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Events: 1 (Each staff member is assigned to one event)
    - EventStaff: 0..* (An event can have zero or many staff members)
- **Description:** Each staff member is associated with one specific event. One event can have multiple staff members assigned to it. The EventID in the EventStaff table is a foreign key referencing the EventID in the Events table.

9. Customers and Feedback:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Customers: 1 (Each feedback record is from one customer)
    - Feedback: 0..* (A customer can give zero or many feedback records)
- **Description:** Each feedback record is associated with one specific customer. One customer can provide multiple feedback records. The CustomerID in the Feedback table is a foreign key referencing the CustomerID in the Customers table.

10. Events and Feedback:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Events: 1 (Each feedback record is for one event)
    - Feedback: 0..* (An event can have zero or many feedback records)
- **Description:** Each feedback record is associated with one specific event. One event can have multiple feedback records. The EventID in the Feedback table is a foreign key referencing the EventID in the Events table.

11. Customers and Waitlist:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Customers: 1 (Each waitlist record is from one customer)
    - Waitlist: 0..* (A customer can join the waitlist for zero or many events)
- **Description:** Each waitlist record is associated with one specific customer. One customer can join the waitlist for multiple events. The CustomerID in the Waitlist table is a foreign key referencing the CustomerID in the Customers table.

12. Events and Waitlist:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Events: 1 (Each waitlist record is for one event)
    - Waitlist: 0..* (An event can have zero or many customers on its waitlist)
- **Description:** Each waitlist record is associated with one specific event. One event can have multiple customers on its waitlist. The EventID in the Waitlist table is a foreign key referencing the EventID in the Events table.

13. Events and TicketAvailability:

- **Relationship:** One-to-many
- **Cardinality:** 1:N
    - Events: 1 (Each TicketAvailability record is for one event)
    - TicketAvailability: 0..* (An event can have zero or many TicketAvailability records)
- **Description:** Each TicketAvailability record is associated with one specific event. One event can have multiple TicketAvailability records. The EventID in the TicketAvailability table is a foreign key referencing the EventID in the Events table.

(D) Primary and Foreign Key Table

Entity	Primary Key (PK)	Foreign Keys (FK)
Venues	VenueID	-
Events	EventID	VenueID
Tickets	TicketID	EventID
Customers	CustomerID	-
Orders	OrderID	CustomerID
OrderTickets	OrderTicketID	OrderID, TicketID
Payments	PaymentID	OrderID
Promotions	PromotionID	EventID
EventStaff	StaffID	EventID
Feedback	FeedbackID	CustomerID, EventID
Waitlist	WaitlistID	CustomerID, EventID
TicketAvailability	AvailabilityID	EventID

(E) ERD Diagram - Ticketing Database Management System