**UNIVERSITY OF KIGALI**

**SCHOOL OF COMPUTING**

**FACULTY: BBIT**

**DATABASE MANAGEMENT SYSTEM**

**DR BUGINGO Emmanuel**

**REG: 2305000338**

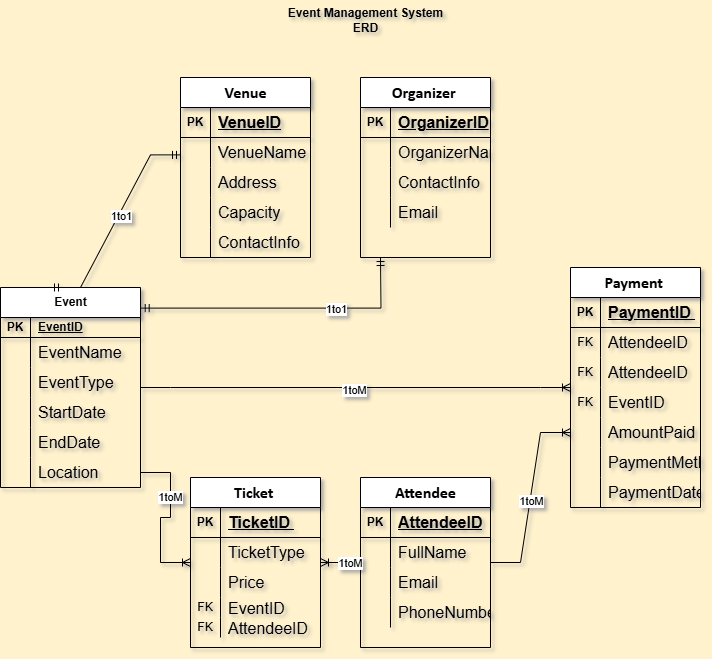
**DATE: 31/01/2024**

**PROJECT: Event Management System**

**2. Description of My Project**

The Event Management System is a database designed to manage events, including: attendee, ticketing, and payment processes. The system ensures efficient organization by tracking venues, event schedules, participants, and financial transactions. Its purpose is to simplify event planning and improve the overall experience for organizers and attendees.

**3.ERD**



**4.** **Logical Data Model (LDM)**

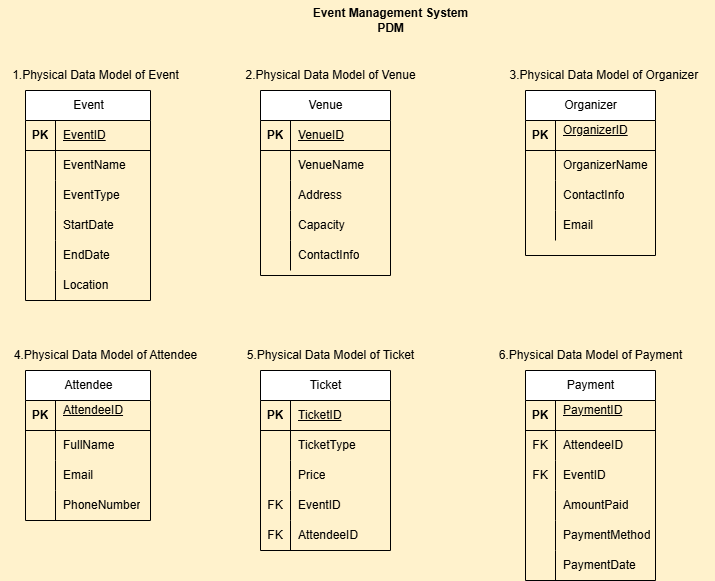
**Entities and Their Attributes:**

1. **Event** (PK: EventID)
   * EventName
   * EventType
   * StartDate
   * EndDate
   * Location
2. **Venue** (PK: VenueID)
   * VenueName
   * Address
   * Capacity
   * ContactInfo
3. **Organizer** (PK: OrganizerID)
   * OrganizerName
   * ContactInfo
   * Email
4. **Attendee** (PK: AttendeeID)
   * FullName
   * Email (Unique)
   * PhoneNumber
5. **Ticket** (PK: TicketID, FK: EventID#, FK: AttendeeID#)
   * TicketType
   * Price
6. **Payment** (PK: PaymentID, FK: AttendeeID#, FK: EventID#)
   * AmountPaid
   * PaymentMethod
   * PaymentDate

**Relationships in LDM:**

* One **Event** has **many Tickets**.
* One **Event** has **many Payments**.
* One **Attendee** can buy **many Tickets**.
* One **Attendee** can make **many Payments**.

**5.** **PDM**



**6.Data Dictionary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| **Event** | EventID | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for an event |
|  | EventName | VARCHAR(255) | NOT NULL | Name of the event |
|  | EventType | VARCHAR(100) |  | Type of event (e.g., conference, workshop) |
|  | StartDate | DATE | NOT NULL | Start date of the event |
|  | EndDate | DATE |  | End date of the event |
|  | Location | VARCHAR(255) |  | Location where the event will take place |
| **Venue** | VenueID | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for a venue |
|  | VenueName | VARCHAR(255) | NOT NULL | Name of the venue |
|  | Address | TEXT | NOT NULL | Address of the venue |
|  | Capacity | INT | NOT NULL | Maximum capacity of the venue |
|  | ContactInfo | VARCHAR(100) |  | Contact details for the venue |
| **Organizer** | OrganizerID | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for an organizer |
|  | OrganizerName | VARCHAR(255) | NOT NULL | Name of the organizer |
|  | ContactInfo | VARCHAR(100) |  | Contact details of the organizer |
|  | Email | VARCHAR(255) | UNIQUE | Email address of the organizer |
| **Attendee** | AttendeeID | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for an attendee |
|  | FullName | VARCHAR(255) | NOT NULL | Full name of the attendee |
|  | Email | VARCHAR(255) | UNIQUE, NOT NULL | Email of the attendee |
|  | PhoneNumber | VARCHAR(20) |  | Contact number of the attendee |
| **Ticket** | TicketID | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for a ticket |
|  | TicketType | VARCHAR(50) | NOT NULL | Type of ticket (e.g., VIP, Regular) |
|  | Price | DECIMAL(10,2) | NOT NULL | Price of the ticket |
|  | EventID | INT | FOREIGN KEY REFERENCES Event(EventID) ON DELETE CASCADE | Associated event for the ticket |
|  | AttendeeID | INT | FOREIGN KEY REFERENCES Attendee(AttendeeID) ON DELETE SET NULL | Attendee who purchased the ticket |
| **Payment** | PaymentID | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for a payment |
|  | AttendeeID | INT | FOREIGN KEY REFERENCES Attendee(AttendeeID) ON DELETE CASCADE | Attendee who made the payment |
|  | EventID | INT | FOREIGN KEY REFERENCES Event(EventID) ON DELETE CASCADE | Event associated with the payment |
|  | AmountPaid | DECIMAL(10,2) | NOT NULL | Amount paid for the ticket |
|  | PaymentMethod | VARCHAR(50) |  | Payment method used (e.g., Credit Card, PayPal) |
|  | PaymentDate | DATE | NOT NULL | Date the payment was made |

**7.SQL(CRUD/COUNT,AVG,SUM) for each table**

**DATABASE CREATION**

CREATE DATABASE Event\_Management\_System;

**TABLES CREATON**

**1. Event Table**

CREATE TABLE Event (

EventID INT PRIMARY KEY AUTO\_INCREMENT,

EventName VARCHAR(255) NOT NULL,

EventType VARCHAR(100),

StartDate DATE NOT NULL,

EndDate DATE,

Location VARCHAR(255)

);

**2. Venue Table**

CREATE TABLE Venue (

VenueID INT PRIMARY KEY AUTO\_INCREMENT,

VenueName VARCHAR(255) NOT NULL,

Address TEXT NOT NULL,

Capacity INT NOT NULL,

ContactInfo VARCHAR(100)

);

**3. Organizer Table**

CREATE TABLE Organizer (

OrganizerID INT PRIMARY KEY AUTO\_INCREMENT,

OrganizerName VARCHAR(255) NOT NULL,

ContactInfo VARCHAR(100),

Email VARCHAR(255) UNIQUE

);

**4. Attendee Table**

CREATE TABLE Attendee (

AttendeeID INT PRIMARY KEY AUTO\_INCREMENT,

FullName VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

PhoneNumber VARCHAR(20)

);

**5. Ticket Table**

CREATE TABLE Ticket (

TicketID INT PRIMARY KEY AUTO\_INCREMENT,

TicketType VARCHAR(50) NOT NULL,

Price DECIMAL(10,2) NOT NULL,

EventID INT,

AttendeeID INT,

FOREIGN KEY (EventID) REFERENCES Event(EventID) ON DELETE CASCADE,

FOREIGN KEY (AttendeeID) REFERENCES Attendee(AttendeeID) ON DELETE SET NULL

);

**6. Payment Table**

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY AUTO\_INCREMENT,

AttendeeID INT NOT NULL,

EventID INT NOT NULL,

AmountPaid DECIMAL(10,2) NOT NULL,

PaymentMethod VARCHAR(50),

PaymentDate DATE NOT NULL,

FOREIGN KEY (AttendeeID) REFERENCES Attendee(AttendeeID) ON DELETE CASCADE,

FOREIGN KEY (EventID) REFERENCES Event(EventID) ON DELETE CASCADE

);

**Insert Data**

INSERT INTO Event (EventName, EventType, StartDate, EndDate, Location) VALUES ('Tech Conference', 'Technology', '2025-06-01', '2025-06-03', 'Kigali Convention Center');

INSERT INTO Venue (VenueName, Address, Capacity, ContactInfo) VALUES ('Kigali Convention Center', 'KG 2 Roundabout, Kigali', 500, 'venue@example.com');

INSERT INTO Organizer (OrganizerName, ContactInfo, Email) VALUES ('John Doe', '0781234567', 'john@example.com');

INSERT INTO Attendee (FullName, Email, PhoneNumber) VALUES ('Jane Smith', 'jane@example.com', '0788765432');

INSERT INTO Ticket (TicketType, Price, EventID, AttendeeID) VALUES ('VIP', 50.00, 1, 1);

INSERT INTO Payment (AttendeeID, EventID, AmountPaid, PaymentMethod, PaymentDate) VALUES (1, 1, 50.00, 'Credit Card', '2025-06-01');

**Read Data**

SELECT \* FROM Event;

SELECT \* FROM Venue;

SELECT \* FROM Organizer;

SELECT \* FROM Attendee;

SELECT \* FROM Ticket;

SELECT \* FROM Payment;

**Update Data**

UPDATE Event SET Location = 'Radisson Blu Hotel' WHERE EventID = 1;

UPDATE Attendee SET PhoneNumber = '0781112233' WHERE AttendeeID = 1;

UPDATE Ticket SET Price = 55.00 WHERE TicketID = 1;

**Delete Data**

DELETE FROM Payment WHERE PaymentID = 1;

DELETE FROM Ticket WHERE TicketID = 1;

DELETE FROM Attendee WHERE AttendeeID = 1;

**Aggregate Queries**

SELECT COUNT(\*) AS TotalEvents FROM Event;

SELECT AVG(Price) AS AverageTicketPrice FROM Ticket;

SELECT SUM(AmountPaid) AS TotalRevenue FROM Payment;

**8.PL/SQL Views**

CREATE VIEW EventDetails AS

SELECT EventID, EventName, EventType, StartDate, EndDate, Location FROM Event;

CREATE VIEW AttendeeList AS

SELECT AttendeeID, FullName, Email FROM Attendee;

CREATE VIEW TicketSales AS

SELECT TicketType, COUNT(\*) AS TotalSold, SUM(Price) AS TotalRevenue FROM Ticket GROUP BY TicketType;

CREATE VIEW PaymentSummary AS

SELECT EventID, SUM(AmountPaid) AS TotalRevenue FROM Payment GROUP BY EventID;

CREATE VIEW OrganizerDetails AS

SELECT OrganizerID, OrganizerName, Email FROM Organizer;

CREATE VIEW VenueCapacity AS

SELECT VenueName, Capacity FROM Venue;

**PL/SQL Stored Procedures**

DELIMITER $$

CREATE PROCEDURE AddEvent(IN eventName VARCHAR(255), IN eventType VARCHAR(100), IN startDate DATE, IN endDate DATE, IN location VARCHAR(255))

BEGIN

INSERT INTO Event (EventName, EventType, StartDate, EndDate, Location) VALUES (eventName, eventType, startDate, endDate, location);

END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE GetTotalRevenue(OUT total DECIMAL(10,2))

BEGIN

SELECT SUM(AmountPaid) INTO total FROM Payment;

END $$

DELIMITER ;

**PL/SQL Triggers**

DELIMITER $$

CREATE TRIGGER after\_event\_insert

AFTER INSERT ON Event

FOR EACH ROW

BEGIN

INSERT INTO LogTable (Description, CreatedAt) VALUES (CONCAT('New Event Added: ', NEW.EventName), NOW());

END $$

DELIMITER ;

**9.Creating a user and granting permissions**

CREATE USER 'event\_admin'@'localhost' IDENTIFIED BY 'securepassword';

GRANT SELECT, INSERT, UPDATE, DELETE ON EventManagementDB.\* TO 'event\_admin'@'localhost';

FLUSH PRIVILEGES;