## 

Tickets Booking

1.12.2024

**─**

John Protopsaltis 4999

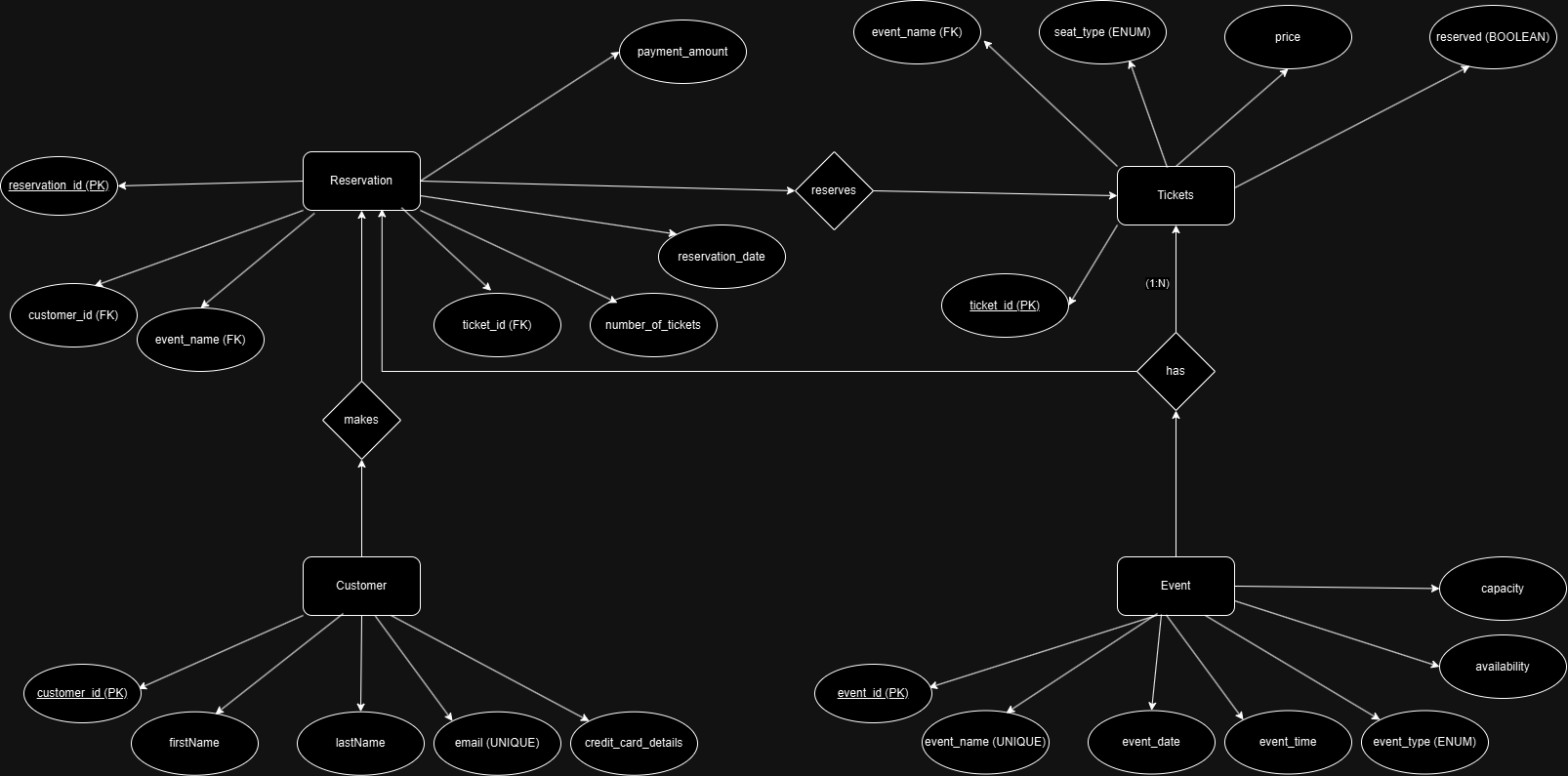
Theodoros Papaliagkas 4980

Stefanos Avramakis 2707

# Contents

1. ER Diagram
2. Entities (Names,Types,Keys,Cardinalities)
3. Relational Model
4. Keys and Functional Dependencies
5. Επεξηγήσεις για τα μη προφανή γνωρίσματα
6. Μετάφραση μοντέλου σε σχεσιακό
7. Εντολές της γλώσσας ορισμού δεδομένων για τις σχέσεις που προκύπτουν
8. Περιορισμοί ακεραιότητας
9. Λειτουργικές εξαρτήσεις

# ER Diagram



# 

# 

# Entities

#### **1. Events**

* **event\_id** (PK, INT, AUTO\_INCREMENT)
* event\_name (VARCHAR(255), UNIQUE, NOT NULL)
* event\_date (DATE, NOT NULL)
* event\_time (TIME, NOT NULL)
* event\_type (ENUM: CONCERT, SPORTS, THEATER, SEMINAR, CONFERENCE, MEETING, OTHER, NOT NULL)
* availability (INT, NOT NULL)
* capacity (INT, NOT NULL)

#### **2. Customers**

* **customer\_id** (PK, INT, AUTO\_INCREMENT)
* firstName (VARCHAR(255), NOT NULL)
* lastName (VARCHAR(255), NOT NULL)
* email (VARCHAR(255), UNIQUE, NOT NULL)
* credit\_card\_details (BIGINT, NOT NULL)

#### **3. Tickets**

* **ticket\_id** (PK, INT, AUTO\_INCREMENT)
* event\_name (FK, VARCHAR(255), NOT NULL → Events.event\_name)
* seat\_type (ENUM: VIP, Regular, Student, NOT NULL)
* price (DECIMAL(10, 2), NOT NULL)
* reserved (BOOLEAN, NOT NULL, DEFAULT FALSE)

#### **4. Reservations**

* **reservation\_id** (PK, INT, AUTO\_INCREMENT)
* customer\_id (FK, INT, NOT NULL → Customers.customer\_id)
* event\_name (FK, VARCHAR(255), NOT NULL → Events.event\_name)
* ticket\_id (FK, INT, NOT NULL → Tickets.ticket\_id)
* number\_of\_tickets (INT, NOT NULL)
* reservation\_date (DATE, NOT NULL, DEFAULT CURRENT\_DATE)
* payment\_amount (DECIMAL(10, 2), NOT NULL)

# Relations

**Events** ↔ **Tickets**:

* **One-to-Many** (Each event can have multiple tickets, identified via event\_name).

**Customers** ↔ **Reservations**:

* **One-to-Many** (Each customer can make multiple reservations).

**Tickets** ↔ **Reservations**:

* **One-to-Many** (Each ticket can be part of multiple reservations).

**Events** ↔ **Reservations**:

* **One-to-Many** (Each event can have multiple reservations, identified via event\_name).

# Keys and Functional Dependencies

#### **1. Events Table**

* **Primary Key (PK)**: event\_id
* **Candidate Keys**: event\_id, event\_name (since event\_name is unique)
* **Functional Dependencies**:
  + event\_id → event\_name, event\_date, event\_time, event\_type, availability, capacity
  + event\_name → event\_id, event\_date, event\_time, event\_type, availability, capacity

#### **2. Customers Table**

* **Primary Key (PK)**: customer\_id
* **Candidate Keys**: customer\_id, email (since email is unique)
* **Functional Dependencies**:
  + customer\_id → firstName, lastName, email, credit\_card\_details
  + email → customer\_id, firstName, lastName, credit\_card\_details

#### **3. Tickets Table**

* **Primary Key (PK)**: ticket\_id
* **Candidate Key**: ticket\_id
* **Foreign Key (FK)**: event\_name references Events(event\_name)
* **Functional Dependencies**:
  + ticket\_id → event\_name, seat\_type, price, reserved
  + event\_name, seat\_type → price (Assuming seat\_type determines the price of tickets for a specific event)

#### **4. Reservations Table**

* **Primary Key (PK)**: reservation\_id
* **Foreign Keys (FKs)**:
  + customer\_id references Customers(customer\_id)
  + event\_name references Events(event\_name)
  + ticket\_id references Tickets(ticket\_id)
* **Functional Dependencies**:
  + reservation\_id → customer\_id, event\_name, ticket\_id, number\_of\_tickets, reservation\_date, payment\_amount
  + customer\_id, event\_name → reservation\_id, number\_of\_tickets, reservation\_date, payment\_amount
  + ticket\_id → event\_name, seat\_type, price

# Επεξηγήσεις για τα μη προφανή γνωρίσματα

#### **1. Events Table**

* **Feature**: event\_type as an ENUM type
  + Αυτό διασφαλίζει ότι μόνο προκαθορισμένες κατηγορίες (π.χ. ΣΥΝΑΥΛΙΑ, ΑΘΛΗΤΙΚΑ, κ.λπ.) μπορούν να αντιστοιχιστούν σε μια εκδήλωση. Αυτός ο περιορισμός αποφεύγει τις ασυνέπειες δεδομένων που προκαλούνται από την εισαγωγή ελεύθερου κειμένου.
* **Non-obvious Relationship**: event\_name → event\_id
  + Εφόσον το event\_name είναι μοναδικό, μπορεί να χρησιμεύσει ως εναλλακτικό κλειδί. Ωστόσο, το event\_id (ως ακέραιος αριθμός AUTO\_INCREMENT) είναι το πραγματικό πρωτεύον κλειδί επειδή είναι πιο αποτελεσματικό για την ευρετηρίαση και πιο γρήγορο στην αναφορά σε σχέσεις.

#### 

#### **2. Customers Table**

* **Feature**: email as a unique identifier
  + Ενώ το customer\_id είναι το κύριο κλειδί, το email είναι επίσης μοναδικό και λειτουργεί ως φυσικό κλειδί για την αναγνώριση πελατών σε σενάρια πραγματικού κόσμου (π.χ. για σύνδεση ή αποστολή ειδοποιήσεων). Ωστόσο, δεδομένου ότι οι χρήστες μπορούν να ενημερώσουν τα email τους, το customer\_id διατηρείται ως το κύριο κλειδί για να διασφαλιστεί η αμετάβλητη.
* **Feature**: credit\_card\_details
  + Αυτό το πεδίο είναι ευαίσθητο και θα πρέπει να αποθηκευτεί με ασφάλεια (π.χ. κρυπτογραφημένο ή διακριτικό) για συμμόρφωση με κανονισμούς όπως το PCI DSS. Αν και είναι ένα μη προφανές πεδίο για απευθείας συμπερίληψη, επιτρέπει ταχύτερη εσωτερική αναφορά για κρατήσεις.

#### **3. Tickets Table**

* **Feature**: seat\_type as an ENUM
  + Αυτό διασφαλίζει ότι τα εισιτήρια κατηγοριοποιούνται με σαφήνεια (VIP, Κανονικό, Φοιτητικό) χωρίς ασάφεια. Η σχέση μεταξύ seat\_type και τιμής προϋποθέτει ότι οι τιμές των εισιτηρίων εξαρτώνται από τον τύπο της θέσης για μια δεδομένη εκδήλωση.
* **Non-obvious Relationship**: event\_name as a foreign key
  + Αντί για αναφορά event\_id (το κύριο κλειδί των Events), το event\_name χρησιμοποιείται εδώ για καλύτερη αναγνωσιμότητα και φιλικότητα προς τον χρήστη. Ωστόσο, υποθέτει ότι το event\_name είναι μοναδικό σε όλα τα συμβάντα (όπως διασφαλίζεται από τον UNIQUE περιορισμό του).

#### **4. Reservations Table**

* **Feature**: reservation\_date with a default of CURRENT\_DATE
  + Η αυτόματη ρύθμιση της ημερομηνίας κράτησης απλοποιεί τη διαδικασία κράτησης μειώνοντας τα πεδία που πρέπει να εισαχθούν χειροκίνητα.
* **Feature**: payment\_amount
  + Η αποθήκευση του payment\_amount στον πίνακα Κρατήσεις βοηθά στην παρακολούθηση του συνολικού ποσού που καταβλήθηκε για τη συγκεκριμένη κράτηση, ακόμα κι αν οι τιμές των εισιτηρίων ή οι εκπτώσεις αλλάξουν στο μέλλον.
* **Non-obvious Relationship**: customer\_id, event\_name → reservation\_id
  + Ένας συνδυασμός customer\_id και event\_name προσδιορίζει μοναδικά τις κρατήσεις για έναν συγκεκριμένο πελάτη. Ωστόσο, το reservation\_id εξακολουθεί να απαιτείται ως το κύριο κλειδί για την ευρετηρίαση και τον χειρισμό περιπτώσεων όπου ένας πελάτης μπορεί να κάνει πολλές κρατήσεις για το ίδιο συμβάν.

#### **5. Relationships Between Tables**

* **Non-obvious Feature**: Cascading Deletes (ON DELETE CASCADE)
  + Εάν διαγραφεί ένα συμβάν, όλα τα σχετικά Εισιτήρια και Κρατήσεις αφαιρούνται επίσης αυτόματα. Ομοίως, η διαγραφή ενός Πελάτη καταργεί τις σχετικές κρατήσεις.
  + Αυτό είναι σημαντικό για τη διατήρηση της ακεραιότητας αναφοράς και την αποφυγή ορφανών εγγραφών (π.χ. εισιτήρια που προορίζονται για διαγραμμένα συμβάντα).
* **Non-obvious Relationship**: Tickets → Reservations
  + Αν και μπορεί να φαίνεται περιττό να αναφέρονται τόσο το ticket\_id όσο και το event\_name στις Κρατήσεις, αυτό διασφαλίζει την ακριβή παρακολούθηση συγκεκριμένων εισιτηρίων που έχουν δεσμευτεί από πελάτες.

#### **6. General Observations**

* **Why AUTO\_INCREMENT for Primary Keys?**
  + Παρόλο που τα φυσικά κλειδιά όπως το event\_name ή το email θα μπορούσαν να λειτουργήσουν, τα υποκατάστατα κλειδιά (AUTO\_INCREMENT ακέραιοι αριθμοί) προτιμώνται για επεκτασιμότητα και απόδοση, ειδικά όταν πρόκειται για μεγάλα σύνολα δεδομένων ή σύνθετα κλειδιά.
* **Why ENUM Types?**
  + Η χρήση του ENUM (για τον τύπο\_συμβάντος και τον τύπο\_θέσης) επιβάλλει αυστηρή επικύρωση σε επίπεδο βάσης δεδομένων, μειώνοντας τον κίνδυνο εισαγωγής μη έγκυρων δεδομένων.

# Μετάφραση μοντέλου σε σχεσιακό

**Events**(event\_id (PK), event\_name (UNIQUE), event\_date, event\_time, event\_type, availability, capacity)

**Primary Key**: event\_id

**Unique Key**: event\_name

**Attributes**:

* event\_id: Auto-incrementing integer for unique identification.
* event\_name: Name of the event, must be unique.
* event\_date: Date of the event.
* event\_time: Time of the event.
* event\_type: ENUM value specifying the event type (e.g., CONCERT, SPORTS).
* availability: Number of tickets available for the event.
* capacity: Total seating capacity for the event.

**Customers**(customer\_id (PK), firstName, lastName, email (UNIQUE), credit\_card\_details)

**Primary Key**: customer\_id

**Unique Key**: email

**Attributes**:

* customer\_id: Auto-incrementing integer for unique identification.
* firstName: Customer's first name.
* lastName: Customer's last name.
* email: Customer's unique email address.
* credit\_card\_details: Customer's credit card number (stored securely).

**Tickets**(ticket\_id (PK), event\_name (FK), seat\_type, price, reserved)

**Primary Key**: ticket\_id

**Foreign Key**: event\_name references Events(event\_name)

**Attributes**:

* ticket\_id: Auto-incrementing integer for unique identification.
* event\_name: Foreign key referencing Events table to associate the ticket with an event.
* seat\_type: ENUM value specifying the seat type (e.g., VIP, Regular, Student).
* price: Decimal value representing the price of the ticket.
* reserved: Boolean value indicating whether the ticket is reserved (default: FALSE).

**Reservations**(reservation\_id (PK), customer\_id (FK), event\_name (FK), ticket\_id (FK), number\_of\_tickets, reservation\_date, payment\_amount)

**Primary Key**: reservation\_id

**Foreign Keys**:

* customer\_id references Customers(customer\_id)
* event\_name references Events(event\_name)
* ticket\_id references Tickets(ticket\_id)

**Attributes**:

* reservation\_id: Auto-incrementing integer for unique identification.
* customer\_id: Foreign key referencing Customers table to link the reservation to a customer.
* event\_name: Foreign key referencing Events table to link the reservation to an event.
* ticket\_id: Foreign key referencing Tickets table to link the reservation to a specific ticket.
* number\_of\_tickets: Integer representing the number of tickets reserved.
* reservation\_date: Date of the reservation (default: CURRENT\_DATE).
* payment\_amount: Decimal value representing the total payment amount.

# Εντολές της γλώσσας ορισμού δεδομένων για τις σχέσεις που προκύπτουν

| CREATE TABLE IF NOT EXISTS Events (  event\_id INT AUTO\_INCREMENT PRIMARY KEY,  event\_name VARCHAR(255) NOT NULL UNIQUE,  event\_date DATE NOT NULL,  event\_time TIME NOT NULL,  event\_type ENUM('CONCERT', 'SPORTS', 'THEATER', 'SEMINAR', 'CONFERENCE', 'MEETING', 'OTHER') NOT NULL,  availability INT NOT NULL,  capacity INT NOT NULL ); |
| --- |

| CREATE TABLE IF NOT EXISTS Customers (  customer\_id INT AUTO\_INCREMENT PRIMARY KEY,  firstName VARCHAR(255) NOT NULL,  lastName VARCHAR(255) NOT NULL,  email VARCHAR(255) UNIQUE NOT NULL,  credit\_card\_details BIGINT NOT NULL ); |
| --- |

| CREATE TABLE IF NOT EXISTS Tickets (  ticket\_id INT AUTO\_INCREMENT PRIMARY KEY,  event\_name VARCHAR(255) NOT NULL,  seat\_type ENUM('VIP', 'Regular', 'Student') NOT NULL,  price DECIMAL(10, 2) NOT NULL,  reserved BOOLEAN NOT NULL DEFAULT FALSE,  FOREIGN KEY (event\_name) REFERENCES Events(event\_name) ON DELETE CASCADE ); |
| --- |

| CREATE TABLE IF NOT EXISTS Reservations (  reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,  customer\_id INT NOT NULL,  event\_name VARCHAR(255) NOT NULL,  ticket\_id INT NOT NULL,  number\_of\_tickets INT NOT NULL,  reservation\_date DATE NOT NULL DEFAULT CURRENT\_DATE,  payment\_amount DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id) ON DELETE CASCADE,  FOREIGN KEY (event\_name) REFERENCES Events(event\_name) ON DELETE CASCADE,  FOREIGN KEY (ticket\_id) REFERENCES Tickets(ticket\_id) ON DELETE CASCADE ); |
| --- |

# Περιορισμοί ακεραιότητας

**Πρωτεύοντα κλειδιά**: Προσδιορίστε μοναδικά τις εγγραφές σε κάθε πίνακα.

Μοναδικοί περιορισμοί: Βεβαιωθείτε ότι ορισμένα πεδία (όπως το event\_name και το email) είναι μοναδικά σε όλες τις σειρές.

Περιορισμοί ξένου κλειδιού: Επιβολή ακεραιότητας αναφοράς διασφαλίζοντας ότι οι εγγραφές σε σχετικούς πίνακες παραμένουν συνεπείς όταν αναφέρονται ή διαγράφονται.

Περιορισμοί Enum: Οριακές τιμές για event\_type και seat\_type σε ένα συγκεκριμένο σύνολο αποδεκτών επιλογών.

NOT NULL Περιορισμοί: Βεβαιωθείτε ότι τα κρίσιμα πεδία δεν μπορούν να μείνουν άδεια.

# Λειτουργικές εξαρτήσεις

Πίνακας συμβάντων: event\_id → όλα τα άλλα χαρακτηριστικά, event\_name → όλα τα άλλα χαρακτηριστικά.

Πίνακας πελατών: customer\_id → όλα τα άλλα χαρακτηριστικά, email → όλα τα άλλα χαρακτηριστικά.

Πίνακας εισιτηρίων: ticket\_id → όλα τα άλλα χαρακτηριστικά, event\_name, seat\_type → όλα τα άλλα χαρακτηριστικά.

Πίνακας κρατήσεων: reservation\_id → όλα τα άλλα χαρακτηριστικά, customer\_id, event\_name, ticket\_id → όλα τα άλλα χαρακτηριστικά.

# Custom SQL Queries: Κατάσταση διαθέσιμων και κρατουμένων θέσεων ανά εκδήλωση:

**SELECT**

**e.event\_name,**

**e.capacity AS total\_seats,**

**(e.capacity - e.availability) AS reserved\_seats,**

**e.availability AS available\_seats**

**FROM**

**events e;**

==========================================================================

Έσοδα από πωλήσεις ανά εκδήλωση:

**SELECT**

**e.event\_name,**

**COALESCE(SUM(t.price), 0) AS total\_revenue**

**FROM**

**Events e**

**LEFT JOIN**

**Tickets t**

**ON**

**e.event\_name = t.event\_name**

**WHERE**

**t.reserved = TRUE**

**GROUP BY**

**e.event\_name;**

==========================================================================

Δημοφιλέστερη εκδήλωση βάσει κρατήσεων:

**SELECT**

**r.event\_name,**

**SUM(r.number\_of\_tickets) AS total\_tickets\_reserved**

**FROM**

**Reservations r**

**GROUP BY**

**r.event\_name**

**ORDER BY**

**total\_tickets\_reserved DESC**

**LIMIT 1;**

==========================================================================

Εκδήλωση με τα περισσότερα έσοδα σε ένα χρονικό εύρος:

**SELECT**

**e.event\_name,**

**SUM(r.payment\_amount) AS total\_revenue**

**FROM**

**Reservations r**

**JOIN**

**Events e ON r.event\_name = e.event\_name**

**WHERE**

**e.event\_date BETWEEN '2024-12-30' AND '2025-05-05'**

**GROUP BY**

**e.event\_name**

**ORDER BY**

**total\_revenue DESC**

**LIMIT 1;**

==========================================================================  
Προβολή κρατήσεων ανά χρονική περίοδο:

**SELECT**

**r.reservation\_id AS Κωδικός\_Κράτησης,**

**CONCAT(c.firstName, ' ', c.lastName) AS Πελάτης,**

**r.event\_name AS Εκδήλωση,**

**e.event\_date AS Ημερομηνία\_Εκδήλωσης,**

**r.ticket\_id AS Εισιτήριο,**

**r.number\_of\_tickets AS Πλήθος\_Εισιτηρίων,**

**r.reservation\_date AS Ημερομηνία\_Κράτησης,**

**r.payment\_amount AS Πληρωμή**

**FROM**

**Reservations r**

**JOIN**

**Customers c ON r.customer\_id = c.customer\_id**

**JOIN**

**Events e ON r.event\_name = e.event\_name**

**WHERE**

**e.event\_date BETWEEN '1000-01-01' AND '2026-12-31'**

**ORDER BY**

**e.event\_date ASC;**

==========================================================================

Τα συνολικά έσοδα από την πώληση VIP ή γενικών εισιτηρίων ανά εκδήλωση ή συνολικά:

**SELECT**

**e.event\_name AS Εκδήλωση,**

**SUM(r.payment\_amount) AS Συνολικά\_Έσοδα**

**FROM**

**Reservations r**

**JOIN**

**Tickets t ON r.ticket\_id = t.ticket\_id**

**JOIN**

**Events e ON r.event\_name = e.event\_name**

**WHERE**

**t.seat\_type IN ('VIP', 'Regular')**

**GROUP BY**

**e.event\_name**

**UNION ALL**

**SELECT**

**'Συνολικά' AS Εκδήλωση,**

**SUM(r.payment\_amount) AS Συνολικά\_Έσοδα**

**FROM**

**Reservations r**

**JOIN**

**Tickets t ON r.ticket\_id = t.ticket\_id**

**JOIN**

**Events e ON r.event\_name = e.event\_name**

**WHERE**

**t.seat\_type IN ('VIP', 'Regular')**

**ORDER BY**

**Συνολικά\_Έσοδα DESC;**