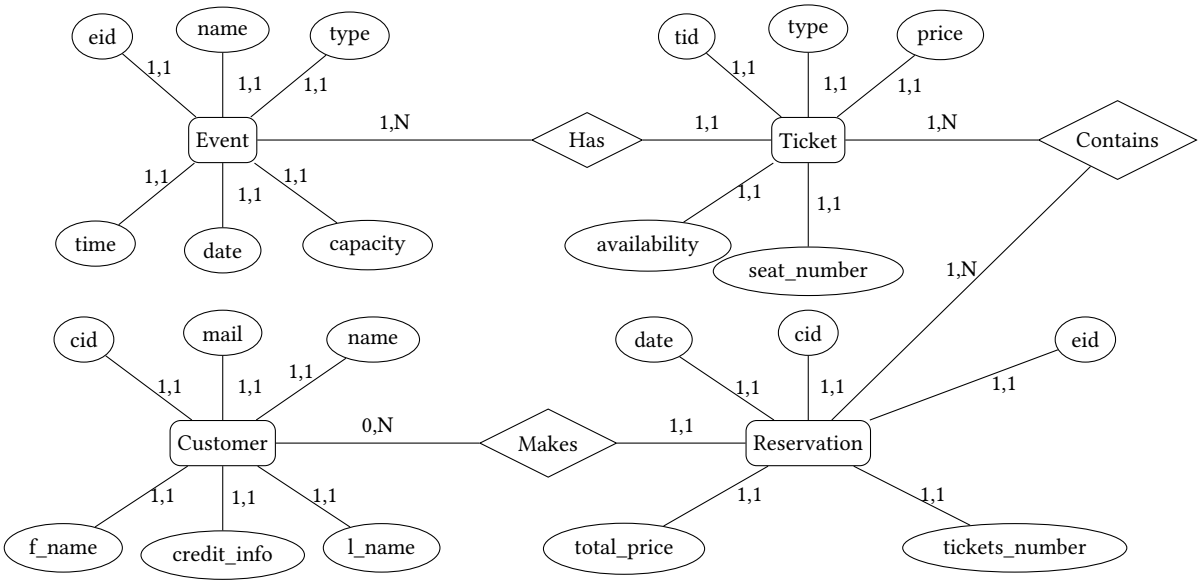
Project HY-360

Δρακάκης Ραφαήλ csd5310
Άγγελος-Τίτος Δήμογλης csd5078
Κωνσταντίνος Κουναλάκης csd5058

1η φάση

Η πρώτη φάση αφορά την δημιουργία ενός πλήρους εννοιολογικού μοντέλου.

E-R Διάγραμμα



Στο διάγραμμα φαίνονται τα γνωρίσματα όλων των οντοτήτων και σχέσεων και τα πρωτεύοντα κλειδιά
Σχετικά με τα γνωρίσματα και τις σχέσεις έχουμε:
Μια σχέση contains ανάμεσα στο ticket και το reservation, και μια σχέση Makes ανάμεσα στον customer και το reservation.
Οι περιορισμοί για τις πληθικότητες φαίνονται στο σχήμα

Μετάφραση στο σχεσιακό μοντέλο

Παρακάτω φαίνονται οι πίνακες για το σχεσιακό μοντέλο

Customer				
cid	mail	credit_info	f_name	l_name

Event					
eid	name	type	time	date	capacity

Ticket				
tid	type	price	availability	seat_number

Reservation					
rid	eid	cid	date	total_price	tickets_number

Contains	
eid	tid

Makes	
cid	rid

Has	
tid	eid

Εντολές SQL για τις σχέσεις που προκύπτουν

```
CREATE TABLE Customer (
    cid INT PRIMARY KEY,
    mail VARCHAR(255) NOT NULL,
    credit_info VARCHAR(255),
    f_name VARCHAR(100),
    l_name VARCHAR(100)
);

CREATE TABLE Event (
    eid INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    type VARCHAR(100),
    time TIME,
```

```
    date DATE,
    capacity INT
);

CREATE TABLE Ticket (
    tid INT PRIMARY KEY,
    type VARCHAR(100),
    price DECIMAL(10, 2),
    availability BOOLEAN DEFAULT TRUE,
    seat_number INT
);

CREATE TABLE Reservation (
    rid INT PRIMARY KEY,
    eid INT,
    cid INT,
    date DATE,
    total_price DECIMAL(10, 2),
    tickets_number INT,
    FOREIGN KEY (eid) REFERENCES Event(eid),
    FOREIGN KEY (cid) REFERENCES Customer(cid)
);

CREATE TABLE Contains (
    eid INT,
    tid INT,
    PRIMARY KEY (eid, tid),
    FOREIGN KEY (eid) REFERENCES Event(eid),
    FOREIGN KEY (tid) REFERENCES Ticket(tid)
);

CREATE TABLE Makes (
    cid INT,
    rid INT,
    PRIMARY KEY (cid, rid),
    FOREIGN KEY (cid) REFERENCES Customer(cid),
    FOREIGN KEY (rid) REFERENCES Reservation(rid)
);

CREATE TABLE Has (
    tid INT,
    eid INT,
    PRIMARY KEY (tid, eid),
    FOREIGN KEY (tid) REFERENCES Ticket(tid),
    FOREIGN KEY (eid) REFERENCES Event(eid)
);
```

Περιορισμοί Ακεραιότητας

Οι περιορισμοί ακεραιότητας για την βάση δεδομένων περιλαμβάνουν τα εξής:

- Πρωτεύοντα Κλειδιά
  - Customer(cid)
  - Event(eid)
  - Ticket(tid)
  - Reservation(rid)
- Ξένα Κλειδιά
  - Στον πίνακα Reservation, το πεδίο eid αναφέρεται στον πίνακα Event.
  - Στον πίνακα Reservation, το πεδίο cid αναφέρεται στον πίνακα Customer.
  - Στον πίνακα Contains, το πεδίο tid αναφέρεται στον πίνακα Ticket.
- Μοναδικότητα: Το πεδίο mail στον πίνακα Customer είναι μοναδικό για κάθε πελάτη.
- Υποχρεωτικά Πεδία: Τα πεδία cid στον πίνακα Customer, eid στον πίνακα Event, και tid στον πίνακα Ticket δεν επιτρέπουν κενές τιμές.
- Επιτρεπόμενες Τιμές: Οι τιμές του price στον πίνακα Ticket πρέπει να είναι θετικές.

Συναρτησιακές Εξαρτήσεις

Οι συναρτησιακές εξαρτήσεις μεταξύ των πεδίων των πινάκων είναι οι εξής:

- cid →mail, credit_info, f_name, l_name (από τον πίνακα Customer)
- eid →name, type, time, date, capacity (από τον πίνακα Event)
- tid →type, price, availability, seat_number (από τον πίνακα Ticket)
- rid →eid, cid, date, total_price, tickets_number (από τον πίνακα Reservation)
- eid, tid →tickets_number (από τη σχέση Contains)

Μετατροπή σε Τρίτη Κανονική Μορφή

1. Πίνακας: Πελάτης Customer

- Κύριο Κλειδί: cid
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Όλα τα χαρακτηριστικά εξαρτώνται πλήρως από το κύριο κλειδί cid.
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Ο πίνακας Customer βρίσκεται σε 3NF.

2. Πίνακας: Εκδήλωση Event

- Κύριο Κλειδί: eid
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Όλα τα χαρακτηριστικά εξαρτώνται πλήρως από το κύριο κλειδί eid.
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Ο πίνακας Event βρίσκεται σε 3NF.

3. Πίνακας: Εισιτήριο Ticket

- Κύριο Κλειδί: tid
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Όλα τα χαρακτηριστικά εξαρτώνται πλήρως από το κύριο κλειδί tid).
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Ο πίνακας Ticket βρίσκεται σε 3NF.

4. Πίνακας: Κράτηση Reservation

- Κύριο Κλειδί: rid
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Όλα τα χαρακτηριστικά εξαρτώνται πλήρως από το κύριο κλειδί rid.
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Ο πίνακας Reservation βρίσκεται σε 3NF.

5. Συσχέτιση: Περιέχει Contains

- Κύριο Κλειδί: Σύνθετο κλειδί eid, tid)
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Δεν υπάρχουν μερικές εξαρτήσεις.
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Η συσχέτιση Contains βρίσκεται σε 3NF.

6. Συσχέτιση: Δημιουργεί Makes

- **Κύριο Κλειδί:** Σύνθετο κλειδί cid, rid)
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Δεν υπάρχουν μερικές εξαρτήσεις.
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Η συσχέτιση Makes βρίσκεται σε 3NF.

7. Συσχέτιση: Έχει Has

- **Κύριο Κλειδί:** Σύνθετο κλειδί tid, eid)
- 1NF: Όλες οι τιμές είναι ατομικές.
- 2NF: Δεν υπάρχουν μερικές εξαρτήσεις.
- 3NF: Δεν υπάρχουν μεταβατικές εξαρτήσεις.

Συμπέρασμα: Η συσχέτιση Has βρίσκεται σε 3NF

Ερωτήματα SQL

Ακολουθούν παραδείγματα SQL ερωτημάτων για την βάση δεδομένων:

- **Ερώτημα για την εύρεση όλων των κρατήσεων ενός πελάτη:**

```
SELECT * FROM Reservation
WHERE cid = 164;
```

- **Ερώτημα για την εύρεση όλων των διαθέσιμων εισιτηρίων για ένα γεγονός:**

```
SELECT * FROM Ticket
WHERE availability = TRUE AND eid = 163;
```

- **Ερώτημα για την εύρεση του συνολικού κόστους μιας κράτησης:**

```
SELECT SUM(price) FROM Ticket
JOIN Reservation ON Ticket.eid = Reservation.eid
WHERE Reservation.rid = 924;
```

Ψευδοκώδικας Διαδικασιών

Ακολουθεί ψευδοκώδικας για την διαδικασία δημιουργίας μιας νέας κράτησης:

```
PROCEDURE ADD_CUSTOMER(email, credit_info, first_name, last_name):
    IF email is not empty AND first_name is not empty
                AND last_name is not empty:
        INSERT INTO Customer (email, credit_info, first_name, last_name)
        RETURN success
    ELSE:
        RETURN "Missing required fields"

PROCEDURE CREATE_RESERVATION(customer_id, event_id, ticket_count, total_price):
    IF ticket_count > 0 AND total_price > 0:
        CHECK IF event_capacity(event_id) >= ticket_count:
            INSERT INTO Reservation (customer_id, event_id,
                        ticket_count, total_price)
            IF reservation is successful:
                UPDATE ticket availability for the event
                RETURN reservation ID
            ELSE:
                RETURN "Reservation failed"
        ELSE:
            RETURN "Not enough capacity"
    ELSE:
        RETURN "Invalid ticket count or price"

PROCEDURE UPDATE_TICKET_AVAILABILITY(ticket_id, availability):
    IF ticket_id exists AND availability is valid:
        UPDATE Ticket
        SET availability = availability
        WHERE ticket_id = ticket_id
        RETURN success
    ELSE:
        RETURN error "Invalid ticket ID or availability"

PROCEDURE CHECK_EVENT_CAPACITY(event_id):
    SELECT capacity
    FROM Event
    WHERE event_id = event_id
    RETURN capacity

PROCEDURE ASSIGN_TICKET_TO_EVENT(ticket_id, event_id):
    IF ticket_id exists AND event_id exists:
        INSERT INTO Has (ticket_id, event_id)
        RETURN success
    ELSE:
```

```
        RETURN "Invalid ticket or event"

PROCEDURE CANCEL_RESERVATION(reservation_id):
    IF reservation_id exists:
        DELETE FROM Reservation
        WHERE reservation_id = reservation_id
        UPDATE ticket availability
        RETURN success
    ELSE:
        RETURN "Reservation not found"

PROCEDURE UPDATE_EVENT(event_id, new_capacity):
    IF event_id exists AND new_capacity > 0:
        UPDATE Event
        SET capacity = new_capacity
        WHERE event_id = event_id
        RETURN success
    ELSE:
        RETURN "Invalid event ID or capacity"

PROCEDURE GET_CUSTOMER_RESERVATIONS(customer_id):
    SELECT *
    FROM Reservation
    WHERE customer_id = customer_id
    RETURN reservation details

PROCEDURE GET_EVENT_TICKETS(event_id):
    SELECT *
    FROM Ticket
    WHERE event_id = event_id AND availability = TRUE
    RETURN available tickets

PROCEDURE GET_TOTAL_COST(reservation_id):
    SELECT SUM(price)
    FROM Ticket
    JOIN Reservation
        ON Ticket.event_id = Reservation.event_id
    WHERE Reservation.reservation_id = reservation_id
    RETURN total cost
```

2η φάση

Ενδεικτικά αποτελέσματα από την εκτέλεση των διαδικασιών

Οι σχετικές εικόνες θα εισαχθούν αργότερα

Εγχειρίδιο χρήσης της εφαρμογης

Για να χρησιμοποιηθεί η εφαρμογή πρέπει να γίνουν τα ακόλουθα:
Αρχικά, εάν θέλουμε μια καθαρή βάση, θα πρέπει να διαγράψουμε το αρχείο eventManagement.db το οποίο περιέχει τις προηγούμενες εγγραφές που έγιναν στην βάση. Έπειτα, τρέχουμε python setupDB.py ώστε να δημιουργήσουμε μια νέα βάση. Στην συνέχεια, στην διεύθυνση http://127.0.0.1:5000/, θα δούμε το διαχειριστικό περιβάλλον της εφαρμογής μας. Μπορούμε να εκτελέσουμε λειτουργίες όπως:

- Εγγραφή νέου πελάτη
- Δημιουργία νέας εκδήλωσης
- Εμφάνιση δεδομένων για πελάτες, εκδηλώσεις, εισητήρια, κρατήσεις
- Εμφάνιση αριθμού διαθέσιμων εισητηρίων
- Εμφάνιση αριθμού κρατήσεων που έχουν γίνει
- Εμφάνιση συνολικού κέρδους
- Διαγραφή δεδομένων για πελάτες, εκδηλώσεις, εισητήρια, κρατήσεις

Περιγραφή των περιορισμών της υλοποίησης