Βάσεις Δεδομένων 2017-18

Εξαμηνιαία εργασία e-Hotels

Κώδικας SQL

Ομάδα:

Διαμαντίδης Θεόδωρος (03115007)

Λούκας Νικόλαος (03115188)

Ξανθόπουλος Βασίλειος-Χρήστος (03115186)

---

1. Κατασκευή βάσης

Η βάση δεδομένων που χρησιμοποιήθηκε για την υποστήριξη της εφαρμογής έχει τη μορφή που φαίνεται στο σχεσιακό διάγραμμα (βλ. Αναφορά έργου).

Για τη δημιουργία αυτής της βάσης δεδομένων εκτελέστηκαν τα παρακάτω DDL statements (όπου ο χρήστης *e-hotels* έχει δημιουργηθεί σύμφωνα με την ενότητα 5 της Αναφοράς έργου):

```
CREATE DATABASE IF NOT EXISTS ehotels;

GRANT ALL ON ehotels.* TO 'e-hotels'@'localhost';

USE ehotels;

CREATE TABLE Hotel_group (
    Hotel_group_ID int(10) UNSIGNED AUTO_INCREMENT NOT NULL,
    Number_of_hotels smallint(4) UNSIGNED DEFAULT 0,
    Address_Street varchar(42) NOT NULL,
    Address_Number smallint(4) UNSIGNED NOT NULL,
    Address_City varchar(42) NOT NULL,
    Address_Postal_Code mediumint(6) UNSIGNED NOT NULL,
    Hotel_group_Name varchar(42) NOT NULL,
    PRIMARY KEY (Hotel_group_ID)
);

CREATE TABLE Hotel_group_Email_Address (
    Hotel_group_ID int(10) UNSIGNED NOT NULL,
    Email_Address varchar(42) NOT NULL,
    FOREIGN KEY (Hotel_group_ID) REFERENCES Hotel_group(Hotel_group_ID) ON UPDATE CASCADE ON
DELETE CASCADE,
    PRIMARY KEY (Hotel_group_ID, Email_Address)
);

CREATE TABLE Hotel_group_Phone_Number (
    Hotel_group_ID int(10) UNSIGNED NOT NULL,
    Phone_Number bigint(10) UNSIGNED NOT NULL,
    FOREIGN KEY (Hotel_group_ID) REFERENCES Hotel_group(Hotel_group_ID) ON UPDATE CASCADE ON
DELETE CASCADE,
    PRIMARY KEY (Hotel_group_ID, Phone_Number)
);
```

```sql
CREATE TABLE Hotel (
    Hotel_ID int(10) UNSIGNED AUTO_INCREMENT NOT NULL,
    Hotel_group_ID int(10) UNSIGNED NOT NULL,
    Stars tinyint(1) UNSIGNED NOT NULL,
    Address_Street varchar(42) NOT NULL,
    Address_Number smallint(4) UNSIGNED NOT NULL,
    Address_City varchar(42) NOT NULL,
    Address_Postal_Code mediumint(6) UNSIGNED NOT NULL,
    Number_of_rooms smallint(4) UNSIGNED DEFAULT 0,
    Hotel_Name varchar(42) NOT NULL,
    FOREIGN KEY (Hotel_group_ID) REFERENCES Hotel_group(Hotel_group_ID) ON UPDATE CASCADE ON
DELETE CASCADE,
    PRIMARY KEY (Hotel_ID)
);

-- For quick searches by city
CREATE INDEX hotel_city
ON Hotel (Address_City);

CREATE TABLE Hotel_Phone_Number (
    Hotel_ID int(10) UNSIGNED NOT NULL,
    Phone_Number bigint(10) UNSIGNED NOT NULL,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (Hotel_ID, Phone_Number)
);

CREATE TABLE Hotel_Email_Address (
    Hotel_ID int(10) UNSIGNED NOT NULL,
    Email_Address varchar(42) NOT NULL,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (Hotel_ID, Email_Address)
);

CREATE TABLE Employee (
    Employee_IRS int(9) ZEROFILL UNSIGNED NOT NULL,
    Social_Security_Number bigint(11) ZEROFILL UNSIGNED UNIQUE NOT NULL
CHECK(Social_Security_Number >= 01010000000),
    Last_Name varchar(42) NOT NULL,
    First_Name varchar(42) NOT NULL,
    Address_Street varchar(42) NOT NULL,
    Address_Number smallint(4) UNSIGNED NOT NULL,
    Address_City varchar(42) NOT NULL,
    Address_Postal_Code mediumint(6) UNSIGNED NOT NULL,
    PRIMARY KEY (Employee_IRS)
);

CREATE INDEX employee_fullname
ON Employee (Last_Name, First_Name);

CREATE TABLE Works (
    Employee_IRS int(9) UNSIGNED NOT NULL,
    Hotel_ID int(10) UNSIGNED NOT NULL,
    Start_Date date NOT NULL,
    Finish_Date date NOT NULL,
    Position varchar(42) NOT NULL,
    FOREIGN KEY (Employee_IRS) REFERENCES Employee(Employee_IRS) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (Employee_IRS, Start_Date)
);

CREATE TABLE Hotel_Room (
    Room_ID smallint(4) UNSIGNED AUTO_INCREMENT NOT NULL,
    Hotel_ID int(10) UNSIGNED NOT NULL,
    Capacity tinyint(2) UNSIGNED NOT NULL,
    View boolean DEFAULT 0,
    Expandable varchar(15) DEFAULT '',
    Repairs_need boolean DEFAULT 0,
    Price decimal(5, 2) UNSIGNED NOT NULL,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (Room_ID, Hotel_ID)
);
```

```sql
CREATE TABLE Room_Amenities (
    Room_ID smallint(4) UNSIGNED NOT NULL,
    Hotel_ID int(10) UNSIGNED NOT NULL,
    amenity varchar(42) NOT NULL,
    FOREIGN KEY (Room_ID, Hotel_ID) REFERENCES Hotel_Room(Room_ID, Hotel_ID) ON UPDATE CASCADE
ON DELETE CASCADE,
    PRIMARY KEY (Room_ID, Hotel_ID, amenity)
);

CREATE TABLE Customer (
    Customer_IRS int(9) ZEROFILL UNSIGNED NOT NULL,
    Social_Security_Number bigint(11) ZEROFILL UNSIGNED UNIQUE NOT NULL
CHECK(Social_Security_Number >= 01010000000),
    Last_Name varchar(42) NOT NULL,
    First_Name varchar(42) NOT NULL,
    Address_Street varchar(42) NOT NULL,
    Address_Number smallint(4) UNSIGNED NOT NULL,
    Address_City varchar(42) NOT NULL,
    Address_Postal_Code mediumint(6) UNSIGNED NOT NULL,
    First_Registration date,
    PRIMARY KEY (Customer_IRS)
);

CREATE INDEX customer_fullname
ON Customer (Last_Name, First_Name);

CREATE TABLE Reserves (
    Room_ID smallint(4) UNSIGNED,
    Hotel_ID int(10) UNSIGNED,
    Customer_IRS int(9) UNSIGNED NOT NULL,
    Start_Date date NOT NULL,
    Finish_Date date NOT NULL,
    Paid boolean DEFAULT 0,
    UNIQUE (Room_ID, Hotel_ID, Start_Date),
    FOREIGN KEY (Room_ID) REFERENCES Hotel_Room(Room_ID) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (Customer_IRS) REFERENCES Customer(Customer_IRS) ON UPDATE CASCADE
);

CREATE TABLE Rents (
    Room_ID smallint(4) UNSIGNED,
    Hotel_ID int(10) UNSIGNED,
    Customer_IRS int(9) UNSIGNED NOT NULL,
    Employee_IRS int(9) UNSIGNED NOT NULL,
    Start_Date date NOT NULL,
    Finish_Date date NOT NULL,
    Rent_ID int(10) UNSIGNED AUTO_INCREMENT NOT NULL,
    UNIQUE (Room_ID, Hotel_ID, Start_Date),
    FOREIGN KEY (Room_ID) REFERENCES Hotel_Room(Room_ID) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (Hotel_ID) REFERENCES Hotel(Hotel_ID) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (Customer_IRS) REFERENCES Customer(Customer_IRS) ON UPDATE CASCADE,
    FOREIGN KEY (Employee_IRS) REFERENCES Employee(Employee_IRS) ON UPDATE CASCADE,
    PRIMARY KEY (Rent_ID)
);

-- Rent-Transaction is a 1:1 relationship, so Rent_ID is both a foreign and a primary key in
Payment_Transaction
CREATE TABLE Payment_Transaction (
    Rent_ID int(10) UNSIGNED NOT NULL,
    Payment_Amount decimal(7, 2) UNSIGNED NOT NULL,
    Payment_Method varchar(12) NOT NULL,
    PRIMARY KEY (Rent_ID),
    FOREIGN KEY (Rent_ID) REFERENCES Rents(Rent_ID) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

2. Κώδικας SQL για υποστήριξη εφαρμογής

2.1. Triggers

Εκτός του κώδικα για τη δημιουργία της βάσης, χρησιμοποιήθηκε επιπλέον κώδικας για τον ορισμό *TRIGGER*, με σκοπό την επαλήθευση δεδομένων, καθώς και τη διαχείριση εισαγωγών, ενημερώσεων και διαγραφών.

Ο κώδικας για τα triggers που ορίστηκαν παρουσιάζεται παρακάτω:

```
DROP TRIGGER IF EXISTS first_registration;
DELIMITER $$
CREATE TRIGGER first_registration BEFORE INSERT ON Customer
    FOR EACH ROW BEGIN
        IF (NEW.First_Registration IS NULL) THEN
            SET NEW.First_Registration = CURDATE();
        END IF;
    END$$
DELIMITER ;

DROP TRIGGER IF EXISTS add_hotel;
CREATE TRIGGER add_hotel AFTER INSERT ON Hotel
    FOR EACH ROW
        UPDATE Hotel_group SET Number_of_hotels = Number_of_hotels + 1 WHERE Hotel_group_ID =
NEW.Hotel_group_ID;

DROP TRIGGER IF EXISTS delete_hotel;
CREATE TRIGGER delete_hotel AFTER DELETE ON Hotel
    FOR EACH ROW
        UPDATE Hotel_group SET Number_of_hotels = Number_of_hotels - 1 WHERE Hotel_group_ID =
OLD.Hotel_group_ID;

DROP TRIGGER IF EXISTS add_room;
CREATE TRIGGER add_room AFTER INSERT ON Hotel_Room
    FOR EACH ROW
        UPDATE Hotel SET Number_of_rooms = Number_of_rooms + 1 WHERE Hotel_ID = NEW.Hotel_ID;

DROP TRIGGER IF EXISTS delete_room;
CREATE TRIGGER delete_room AFTER DELETE ON Hotel_Room
    FOR EACH ROW
        UPDATE Hotel SET Number_of_rooms = Number_of_rooms - 1 WHERE Hotel_ID = OLD.Hotel_ID;

-- Employee update check
DROP TRIGGER IF EXISTS update_employee;
DELIMITER $$
CREATE TRIGGER update_employee BEFORE UPDATE ON Works
    FOR EACH ROW BEGIN
        IF OLD.Position = 'manager' AND NEW.Position <> 'manager' AND
            (SELECT COUNT(Employee_IRS) FROM Works WHERE
                Hotel_ID = NEW.Hotel_ID AND Position = 'manager' AND
                Start_Date <= NEW.Finish_Date AND NEW.Start_Date <= Finish_Date
            ) >= 1 THEN
            SET @message_text = CONCAT('Error for employee #', NEW.Employee_IRS, ': Can''t
update this employee since the hotel will be left without a manager.');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
        END IF;
    END$$
DELIMITER ;

-- Employee deletion check
DROP TRIGGER IF EXISTS delete_employee;
DELIMITER $$
CREATE TRIGGER delete_employee BEFORE DELETE ON Works
    FOR EACH ROW BEGIN
        IF OLD.Position = 'manager' AND (SELECT COUNT(Employee_IRS) FROM Works WHERE
                Hotel_ID = OLD.Hotel_ID AND Position = 'manager' AND
                Start_Date <= OLD.Finish_Date AND OLD.Start_Date <= Finish_Date
```

```
            ) >= 1 THEN
            SET @message_text = CONCAT('Error for employee #', OLD.Employee_IRS, ': Can''t
delete this employee since the hotel will be left without a manager.');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
        END IF;
    END$$
DELIMITER ;

-- Employee assignment checks
DROP TRIGGER IF EXISTS assign_employee;
DELIMITER $$
CREATE TRIGGER assign_employee BEFORE INSERT ON Works
    FOR EACH ROW BEGIN
        IF NEW.Finish_Date < NEW.Start_Date THEN
            SET @message_text = CONCAT('Error for employee #', NEW.Employee_IRS, ': Finish date
can''t be earlier than start date');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
        END IF;
        IF (SELECT COUNT(Employee_IRS) FROM Works WHERE Employee_IRS = NEW.Employee_IRS AND
            Start_Date <= NEW.Finish_Date AND NEW.Start_Date <= Finish_Date
        ) >= 1 THEN
            SET @message_text = CONCAT('Error for employee #', NEW.Employee_IRS, ': The given
working period conflicts with an existent one.');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
        END IF;
    END$$
DELIMITER ;

-- Reserve check
DROP TRIGGER IF EXISTS reserve_room;
DELIMITER $$
CREATE TRIGGER reserve_room BEFORE INSERT ON Reserves
    FOR EACH ROW BEGIN
        IF NEW.Start_Date > NEW.Finish_Date THEN
            SET @message_text = CONCAT('Error for reserve (', NEW.Room_ID, ', ', NEW.Hotel_ID,
', ', NEW.Start_Date , '): Finish date can''t be earlier than start date.');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
        END IF;
        IF (SELECT COUNT(Room_ID)
            FROM Reserves
            WHERE
                Room_ID = NEW.Room_ID AND
                Hotel_ID = NEW.Hotel_ID AND
                Start_Date <= NEW.Finish_Date AND
                NEW.Start_Date <= Finish_Date
            ) >= 1 THEN
            SET @message_text = CONCAT('Error for reserve (', NEW.Room_ID, ', ', NEW.Hotel_ID,
', ', NEW.Start_Date , ') : Reserve period conflicts with an existent one.');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message_text;
        END IF;
    END$$
DELIMITER ;

-- Payment for room rental
DROP TRIGGER IF EXISTS pay_room_rent;
DELIMITER $$
CREATE TRIGGER pay_room_rent AFTER INSERT ON Payment_Transaction
    FOR EACH ROW BEGIN
        DECLARE s_date DATE;
        DECLARE r_id, h_id INT;
        SET s_date = (SELECT Start_Date FROM Rents WHERE Rent_ID = NEW.Rent_ID);
        SET r_id = (SELECT Room_ID FROM Rents WHERE Rent_ID = NEW.Rent_ID);
        SET h_id = (SELECT Hotel_ID FROM Rents WHERE Rent_ID = NEW.Rent_ID);
        UPDATE Reserves SET Paid = 1 WHERE Start_Date = s_date AND Room_ID = r_id AND Hotel_ID
= h_id;
    END$$
DELIMITER ;
```

2.2 Ερωτήματα SQL στο backend

Επιπλέον των προηγουμένων, η εφαρμογή εκτελεί ερωτήματα SQL μέσω της PHP (επέκταση MySQLi) που αφορούν την εισαγωγή, επεξεργασία, διαγραφή ή αναζήτηση καταχωρήσεων στους πίνακες της βάσης δεδομένων.

Τα εν λόγω ερωτήματα μπορούν να βρεθούν στον κώδικα της εφαρμογής που συμπεριλαμβάνεται στο αρχείο υποβολής. Ενδεικτικά παρατίθενται κάποια παραδείγματα:

- Ερώτημα για την εύρεση του ιστορικού κρατήσεων/ενοικιάσεων για ένα δωμάτιο σε χρονολογική σειρά (νεότερες προς παλαιότερες):

```
SELECT Reserves.*,
       Rents.Rent_ID,
       Payment_Transaction.Payment_Method, Payment_Transaction.Payment_Amount
FROM   Reserves
LEFT JOIN Rents ON Rents.Room_ID = Reserves.Room_ID AND Rents.Hotel_ID =
Reserves.Hotel_ID AND Rents.Start_Date = Reserves.Start_Date
LEFT JOIN Payment_Transaction ON Payment_Transaction.Rent_ID = Rents.Rent_ID
WHERE Reserves.Room_ID = {$room_id} AND Reserves.Hotel_ID = {$hotel_id}
ORDER BY Reserves.Start_Date DESC
```

- Ερώτημα για την εύρεση των υπαλλήλων που εργάζονται αυτήν τη στιγμή σε ένα ξενοδοχείο, ενοποιημένο με πληροφορίες για την εργασία:

```
SELECT Employee.*, Works.*
FROM   Employee
INNER JOIN Works ON Works.Employee_IRS = Employee.Employee_IRS
WHERE Works.Hotel_ID = {$hotel_id} AND CURDATE() BETWEEN Works.Start_Date AND
Works.Finish_Date
```

- Ερώτημα για την εισαγωγή μίας νέας κράτησης:

```
INSERT INTO Reserves (Customer_IRS, Room_ID, Hotel_ID, Start_Date, Finish_Date)
VALUES ({$customer_irs}, {$room_id}, {$hotel_id}, DATE('{$start_date}'),
DATE('{$finish_date}'));
```

Για πολύ συχνά είδη ερωτημάτων, όπως είναι η εισαγωγή/επεξεργασία ή διαγραφή Ξενοδοχείων, Δωματίων, Υπαλλήλων κλπ., δημιουργήθηκαν οι γενικές μέθοδοι *`Model::create()`*, *`Model::update()`*, *`Model::delete()`* (κώδικας στο αρχείο *`/src/models/Model.php`*) οι οποίες κληρονομούνται σε καθένα από τα Models της εφαρμογής, άρα και στα προαναφερθέντα.