Questions 2-7

```
/* 2 */
DROP TABLE IF EXISTS contact CASCADE;
DROP TABLE IF EXISTS passengers CASCADE;
DROP TABLE IF EXISTS creditcard CASCADE;
DROP TABLE IF EXISTS reservation CASCADE;
DROP TABLE IF EXISTS flights CASCADE;
DROP TABLE IF EXISTS weekday CASCADE;
DROP TABLE IF EXISTS weekly_schedule CASCADE;
DROP TABLE IF EXISTS route CASCADE;
DROP TABLE IF EXISTS destination CASCADE;
DROP TABLE IF EXISTS profitfactor CASCADE;

CREATE TABLE passengers
(
  id INT NOT NULL AUTO_INCREMENT,
  reservation_id INT NOT NULL,
  passnr INT NOT NULL,
  fullname VARCHAR(30) NOT NULL,
  ticketnr INT,
  PRIMARY KEY(id)
);

CREATE TABLE reservation
(
  id INT NOT NULL AUTO_INCREMENT,
  ppl_amount INT NOT NULL,
  flights_id INT NOT NULL,
  isBooked BOOLEAN NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE creditcard
(
  id INT NOT NULL AUTO_INCREMENT,
  num BIGINT NOT NULL,
  name VARCHAR(30) NOT NULL,
  reservation_id INT NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY (reservation_id) REFERENCES reservation(id)
);

CREATE TABLE contact
(
  id INT NOT NULL,
  email VARCHAR(30) NOT NULL,
  phonenr BIGINT NOT NULL,
```

```
  reservation_id INT NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY (id) REFERENCES passengers(id),
  FOREIGN KEY (reservation_id) REFERENCES reservation(id)
);

CREATE TABLE flights
(
  id INT NOT NULL AUTO_INCREMENT,
  week INT NOT NULL,
  weekly_schedule_id INT NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE weekly_schedule
(
  id INT NOT NULL AUTO_INCREMENT,
  weekday VARCHAR(10) NOT NULL,
  dept_time TIME NOT NULL,
  route INT NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE destination
(
  id VARCHAR(3) NOT NULL,
  name VARCHAR(30) NOT NULL,
  country VARCHAR(30) NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE route
(
  id INT NOT NULL AUTO_INCREMENT,
  year INT NOT NULL,
  price DOUBLE NOT NULL,
  dest VARCHAR(3) NOT NULL,
  origin VARCHAR(3) NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY (dest) REFERENCES destination(id),
  FOREIGN KEY (origin) REFERENCES destination(id)
);

CREATE TABLE weekday
(
  name VARCHAR(10) NOT NULL,
  year INT NOT NULL,
  factor DOUBLE NOT NULL,
  PRIMARY KEY(name, year)
```

```
);

CREATE TABLE profitfactor
(
  year INT NOT NULL,
  factor DOUBLE NOT NULL,
  PRIMARY KEY(year)
);

ALTER TABLE passengers
ADD CONSTRAINT fk_pass_reservation
FOREIGN KEY (reservation_id) REFERENCES reservation(id);

ALTER TABLE reservation
ADD FOREIGN KEY (flights_id) REFERENCES flights(id);

ALTER TABLE flights
ADD FOREIGN KEY (weekly_schedule_id) REFERENCES weekly_schedule(id);

ALTER TABLE weekly_schedule
ADD FOREIGN KEY (route) REFERENCES route(id);

/* 3 */
DELIMITER //

DROP PROCEDURE IF EXISTS addYear;
DROP PROCEDURE IF EXISTS addDay;
DROP PROCEDURE IF EXISTS addDestination;
DROP PROCEDURE IF EXISTS addRoute;
DROP PROCEDURE IF EXISTS addFlight;

CREATE PROCEDURE addYear(IN year INT, IN factor DOUBLE)
BEGIN
  INSERT INTO profitfactor
  VALUES(year, factor);
END;
//

CREATE PROCEDURE addDay(IN year INT, IN day VARCHAR(10), IN factor DOUBLE)
BEGIN
  INSERT INTO weekday
  values(day, year, factor);
END;
//

CREATE PROCEDURE addDestination(IN airport_code VARCHAR(3), IN airport_name
VARCHAR(30), IN country VARCHAR(30))
BEGIN
  INSERT INTO destination
```

```
  VALUES(airport_code, airport_name, country);
END;
//

CREATE PROCEDURE addRoute(IN departure_airport_code VARCHAR(3), IN
arrival_airport_code VARCHAR(3), IN year INT, IN routeprice DOUBLE)
BEGIN
  INSERT INTO route(year, price, dest, origin)
  VALUES(year, routeprice, arrival_airport_code, departure_airport_code);
END;
//

CREATE PROCEDURE addFlight(IN departure_airport_code VARCHAR(3), IN
arrival_airport_code VARCHAR(3), IN year INT, IN day VARCHAR(10), IN departure_time
TIME)
BEGIN
  DECLARE i INT;
  INSERT INTO weekly_schedule(weekday, dept_time, route)
  VALUES(day, departure_time,
    (SELECT id FROM route WHERE dest=arrival_airport_code AND
origin=departure_airport_code AND route.year=year));
  SET i = 0;
  REPEAT
    SET i = i + 1;

    INSERT INTO flights(week, weekly_schedule_id)
    VALUES(i,
      (
        SELECT id FROM weekly_schedule WHERE route IN (
          SELECT id FROM route WHERE
            dest=arrival_airport_code AND
            origin=departure_airport_code AND
            route.year=year
        ) AND dept_time=departure_time
      )
    );

    UNTIL i = 52
  END REPEAT;
END;
//
/* 4 */
DROP FUNCTION IF EXISTS calculateFreeSeats;
DROP FUNCTION IF EXISTS calculatePrice;

CREATE FUNCTION calculateFreeSeats(flightnumber INT) RETURNS INT DETERMINISTIC
BEGIN
  DECLARE bookedSeats INT;
  SELECT (40 - SUM(r4.ppl_amount)) INTO bookedSeats FROM reservation AS r4 WHERE
```

```
    r4.flights_id = flightnumber AND r4.isBooked = 1;

  IF bookedSeats IS NULL THEN
    RETURN 40;
  END IF;
  RETURN bookedSeats;
END
//

CREATE FUNCTION calculatePrice(flightnumber INT) RETURNS DOUBLE DETERMINISTIC
BEGIN
  /* TotalPrice =
      RoutePrice * WeekdayFactor * (BookedPassengers + 1)/40 * profitFactor */
  DECLARE routePrice DOUBLE;
  DECLARE weekdayFactor DOUBLE;
  DECLARE bookedPassengers INT;
  DECLARE profitFactor DOUBLE;

  /* Route price */
  SELECT route.price INTO routePrice FROM route WHERE id IN (
    SELECT route FROM weekly_schedule WHERE id IN (
      SELECT weekly_schedule_id FROM flights WHERE id=flightnumber
    )
  );

  /* Weekday factor */
  SELECT factor INTO weekdayFactor FROM weekday WHERE name IN (
    SELECT weekday FROM weekly_schedule WHERE id IN (
      SELECT weekly_schedule_id FROM flights WHERE id=flightnumber
    )
  ) AND year IN (
    SELECT year FROM route WHERE id IN (
      SELECT route FROM weekly_schedule WHERE id IN (
        SELECT weekly_schedule_id FROM flights WHERE id=flightnumber
      )
    )
  );

  /* Booked passengers */
  SELECT (SUM(ppl_amount) + 1) INTO bookedPassengers FROM reservation WHERE
    flights_id=flightnumber AND isBooked=1;

  IF bookedPassengers IS NULL THEN
    SET bookedPassengers = 1;
  END IF;

  /* Profit factor */
  SELECT factor INTO profitFactor FROM profitfactor WHERE year IN (
    SELECT year FROM route WHERE id IN (
```

```
      SELECT route FROM weekly_schedule WHERE id IN (
        SELECT weekly_schedule_id FROM flights WHERE id=flightnumber
      )
    )
  );

  RETURN ROUND((routePrice * weekdayFactor * (bookedPassengers / 40) * profitFactor), 3);
END
//
/* 5 */
DROP TRIGGER IF EXISTS booking_rand;

CREATE TRIGGER booking_rand BEFORE UPDATE ON reservation
FOR EACH ROW BEGIN
  DECLARE res_id INT;
  IF NEW.isBooked = 1 AND OLD.isBooked = 0 THEN
    SET res_id = NEW.id;

    UPDATE passengers
    SET ticketnr = FLOOR(1 + RAND() * 100000)
    WHERE res_id = reservation_id;
  END IF;
END;
//

/* 6 */
DROP PROCEDURE IF EXISTS addReservation;
DROP PROCEDURE IF EXISTS addPassenger;
DROP PROCEDURE IF EXISTS addContact;
DROP PROCEDURE IF EXISTS addPayment;

CREATE PROCEDURE addReservation(IN departure_airport_code VARCHAR(3), IN
arrival_airport_code VARCHAR(3), IN year INT, IN week INT, IN day VARCHAR(10), IN time
TIME, IN number_of_passengers INT, OUT reservation_id INT)
addreserv:BEGIN
  DECLARE flight_id INT;

  SELECT id INTO flight_id FROM flights WHERE weekly_schedule_id IN (
    SELECT id FROM weekly_schedule WHERE route IN (
      SELECT id FROM route WHERE
        dest=arrival_airport_code AND
        origin=departure_airport_code AND
        year=route.year
    ) AND dept_time=time AND weekday=day
  ) AND flights.week = week;

  IF flight_id IS NULL THEN
    SELECT "There exist no flight for the given route, date and time" AS "Error";
    LEAVE addreserv;
```

```
  END IF;

  IF number_of_passengers > 40 THEN
    SELECT "There are not enough seats available on the chosen flight" AS "Error";
    LEAVE addreserv;
  END IF;

  INSERT INTO reservation(ppl_amount, flights_id, isBooked)
  VALUES(number_of_passengers, flight_id, 0);

  SELECT LAST_INSERT_ID() INTO reservation_id;
END;
//

CREATE PROCEDURE addPassenger(IN reservation_id INT, IN passnr INT, IN name
VARCHAR(30))
addpass:BEGIN
  IF reservation_id NOT IN (SELECT id FROM reservation) THEN
    SELECT "The given reservation number does not exist" AS "Error";
    LEAVE addpass;
  END IF;

  IF (SELECT isBooked FROM reservation WHERE id=reservation_id) THEN
    SELECT "The booking has already been payed and no futher passengers can be added" AS
"Error";
    LEAVE addpass;
  END IF;

  INSERT INTO passengers(reservation_id, passnr, fullname)
  VALUES(reservation_id, passnr, name);
END;
//

CREATE PROCEDURE addContact(IN reservation_id INT, IN passnr INT, IN email
VARCHAR(30), IN phone BIGINT)
addcont:BEGIN
  DECLARE passenger_id INT;

  IF reservation_id NOT IN (SELECT id FROM reservation) THEN
    SELECT "The given reservation number does not exist" AS "Error";
    LEAVE addcont;
  END IF;

  SELECT id INTO passenger_id FROM passengers AS p WHERE
    p.reservation_id=reservation_id AND
    p.passnr=passnr;

  IF passenger_id IS NULL THEN
    SELECT "The person is not a passenger of the reservation" AS "Error";
```

```
    LEAVE addcont;
  END IF;

  INSERT INTO contact(id, email, phonenr, reservation_id)
  VALUES(passenger_id, email, phone, reservation_id);
END;
//

CREATE PROCEDURE addPayment(IN reservation_id_in INT, IN cardname VARCHAR(30), IN
cardnum BIGINT)
addpay:BEGIN
  DECLARE flight_id INT;

  IF reservation_id_in NOT IN (SELECT r1.id FROM reservation AS r1) THEN
    SELECT "The given reservation number does not exist" AS "Error";
    LEAVE addpay;
  END IF;

  IF (SELECT c1.id FROM contact AS c1 WHERE c1.reservation_id=reservation_id_in) IS NULL
THEN
    SELECT "The reservation has no contact yet" AS "Error";
    LEAVE addpay;
  END IF;

  IF (SELECT COUNT(p1.id) FROM passengers AS p1 WHERE
p1.reservation_id=reservation_id_in) > calculateFreeSeats(flight_id) THEN
    SELECT "There are not enough seats available on the flight anymore, deleting reservation" AS
"Error";

    DELETE FROM creditcard WHERE reservation_id=reservation_id_in;
    DELETE FROM contact WHERE reservation_id=reservation_id_in;
    DELETE FROM passengers WHERE reservation_id=reservation_id_in;
    DELETE FROM reservation WHERE id=reservation_id_in;
    LEAVE addpay;
  END IF;

  INSERT INTO creditcard(num, name, reservation_id)
  VALUES(cardnum, cardname, reservation_id_in);

  UPDATE reservation AS r3
  SET r3.isBooked = 1
  WHERE r3.id = reservation_id_in;
END;
//

/* 7 */
DELIMITER ;

DROP VIEW IF EXISTS allFlights;
```

```
CREATE VIEW allFlights AS
  /* departure_city_name, destination_city_name, departure_time,
    departure_day, departure_week, departure_year, nr_of_free_seats,
    current_price_per_seat */
  SELECT o.name AS departure_city_name,
       d.name AS destination_city_name,
       w.dept_time AS departure_time,
       w.weekday AS departure_day,
       f.week AS departure_week,
       r.year AS departure_year,
       calculateFreeSeats(f.id) AS nr_of_free_seats,
       calculatePrice(f.id) AS current_price_per_seat
  FROM
    destination AS o, destination AS d, weekly_schedule AS w, flights AS f, route AS r
  WHERE
    o.id IN (
      SELECT origin FROM route WHERE id IN (
        SELECT route FROM weekly_schedule WHERE id IN (
          SELECT weekly_schedule_id FROM flights
        )
      )
    ) AND d.id IN (
      SELECT dest FROM route WHERE id IN (
        SELECT route FROM weekly_schedule WHERE id IN (
          SELECT weekly_schedule_id FROM flights
        )
      )
    ) AND w.id IN (
      SELECT weekly_schedule_id FROM flights
    ) AND r.id IN (
      SELECT route FROM weekly_schedule WHERE id IN (
        SELECT weekly_schedule_id FROM flights
      )
    )
  ;
```

Questions 8-10 (no queries)

8)

a) You can encrypt the credit card info and protect your database from SQL injections.

b) 1: Less strain on the client-side.

2: Can protect from false parameter inputs.

3: Easier to change/bugfix code.

9)

b) Not visible. Because it hasn't been commited yet, only stored in the A-instance.

c) B waits for a COMMIT. It does this to prevent synchronization errors.

10)

a) Yes, because we don't have any transaction control.

b) Yes, it's possible if both instances check if they can add more before either adds more (so the tables are empty for both).

c) The case occurs depending on how the OS the database is running on functions, in what timing the queries are executed, and other eventual processes the database OS is running (and what they are doing).

d) The LOCK TABLES implementation SHOULD work by not allowing the tables to be modified by another thread until the thread currently having the locks releases them. Since we also individually lock every instance of every table we use, there should be no chance for synchronization errors to happen. The lock surrounds the addPayment procedure call since we can get a synchronization error in the addPayment procedure which would lead to overbooking.

Secondary Index)

CREATE INDEX origin ON route(origin);

For searching purposes, it'll be far more efficient to have a way to check all the planes that leave from a certain airport (so the customer can know where he/she can fly).

Question 10 (queries)

```
/
**********************************************************************************
**********
 Question 10, concurrency
 This is the second of two scripts that tests that the BryanAir database can handle concurrency.
 This script sets up a valid reservation and tries to pay for it in such a way that at most
 one such booking should be possible (or the plane will run out of seats). This script should
 be run in both terminals, in parallel.
```

**********/

```
SELECT "Testing script for Question 10, Adds a booking, should be run in both terminals" as
"Message";
SELECT "Adding a reservations and passengers" as "Message";
CALL addReservation("MIT","HOB",2010,1,"Monday","09:00:00",21,@a);
CALL addPassenger(@a,00000001,"Saruman");
CALL addPassenger(@a,00000002,"Orch1");
CALL addPassenger(@a,00000003,"Orch2");
CALL addPassenger(@a,00000004,"Orch3");
CALL addPassenger(@a,00000005,"Orch4");
CALL addPassenger(@a,00000006,"Orch5");
CALL addPassenger(@a,00000007,"Orch6");
CALL addPassenger(@a,00000008,"Orch7");
CALL addPassenger(@a,00000009,"Orch8");
CALL addPassenger(@a,00000010,"Orch9");
CALL addPassenger(@a,00000011,"Orch10");
CALL addPassenger(@a,00000012,"Orch11");
CALL addPassenger(@a,00000013,"Orch12");
CALL addPassenger(@a,00000014,"Orch13");
CALL addPassenger(@a,00000015,"Orch14");
CALL addPassenger(@a,00000016,"Orch15");
CALL addPassenger(@a,00000017,"Orch16");
CALL addPassenger(@a,00000018,"Orch17");
CALL addPassenger(@a,00000019,"Orch18");
CALL addPassenger(@a,00000020,"Orch19");
CALL addPassenger(@a,00000021,"Orch20");
CALL addContact(@a,00000001,"saruman@magic.mail",080667989);
DO SLEEP(5);
SELECT "Making payment, supposed to work for one session and be denied for the other" as
```

```
"Message";


LOCK TABLES creditcard WRITE,
        contact WRITE,
        passengers WRITE,
        reservation WRITE,
        contact AS c1 WRITE,
        passengers AS p1 WRITE,
        reservation AS r1 WRITE,
        reservation AS r3 WRITE,
        reservation AS r4 WRITE;
CALL addPayment (@a, "Sauron",7878787878);
UNLOCK TABLES;


SELECT "Nr of free seats on the flight (should be 19 if no overbooking occured, otherwise -2): " as
"Message", (SELECT nr_of_free_seats from allFlights where departure_week = 1) as
"nr_of_free_seats;"
```

EER-Diagram (slightly updated)