[TDDD37] Project

## Description and brief overview

**create\_flights (IN start, IN days)**

Creates flights from a start date for a given amount of dates, based on the schedule made by the weekly\_flights. These are to be considered abstract flights in a schedule, while the flights generated into regular\_flights are considered concrete, actual flights.

**create\_single\_flight (IN cur\_date)**

Creates a single set of flights for a given date. These flights are created from the weekly\_schedule for the weekly\_flights.

**reserve\_seats(IN seats, IN flight\_number)**

Reserves the given number of seats on the given flight, if there is room on the flight. By this, the number of paid bookings are only considered since a person is not guaranteed a seat until the flight has been paid. Until the booking is paid, seats are only reserved, and this does not guarantee a seat on the plain.   
This is done by creating the amount of reservations as rows in the passenger\_on table, where the customer field is left NULL and the payed field is left FALSE.

**add\_passenger(IN booking\_number, IN pid, IN title, IN first\_name, IN last\_name, IN age)**

Adds a passenger to a booking, given a booking number and the passengers credentials. This is a convenience method for not first inserting the customer into the customer table and then call add\_existing\_passenger().

**add\_existing\_passenger(IN booking\_number, IN pid)**

Given the personal number for a person registered in the database, this procedure adds that person to a booking. This can be done at any time, after the booking is made (at this point called a reservation). It could either be done before or after the payment is done, but the person is only guaranteed a place if the booking is payed for.

**add\_creditcard (IN card\_number, IN type, IN expires\_month, IN expires\_year, IN customer)**

Adds a credit card to a customer, given the credentials of the credit card and the customer. Checks are performed whether the customer defined by the personal id passed to the procedure is actually in the Brian Air database. Further on, if the customer not already is in the database as a creditcard\_customer, he/she will be inserted there. The name on the credit card is assumed to be the same as the person signing up with that card.

**pay\_booking(IN booking\_number, IN creditcard\_number)**

Pays the given booking with the given creditcard. The price is calculated as defined in the assignment. A check is made if the reserved seats still are available. If that is not the case, the booking is cancelled and removed, as well as all the passenger from the passenger\_on relation.

**get\_flight\_price(IN flight\_num, OUT price)**

Calculates the current price for a flight.

**get\_seats\_taken(IN flight, OUT payed\_seats)**

Returns how many seats that are reserved, booked and payed for on a given flight.

**flights\_to(IN dest, IN date)**

## C:\Users\Anton\Downloads\Project - ER v2 (1).png

## Assignment 5

|  |  |
| --- | --- |
| **SESSION A** | **SESSION B** |
| **mysql> SET autocommit=0;**  **Query OK, 0 rows affected (0.00 sec)**  **mysql> SELECT \* FROM booking;**  **+----------------+-------------------+-------------+-----------+**  **| booking\_number | ticket\_number | contact | flight |**  **+----------------+-------------------+-------------+-----------+**  **| 1 | 6ce8966c2268a6309 | 600510-6493 | BA1411250 |**  **+----------------+-------------------+-------------+-----------+**  **1 row in set (0.00 sec)** |  |
|  | **mysql> SET autocommit=0;**  **Query OK, 0 rows affected (0.00 sec)**  **mysql> SELECT \* FROM booking;**  **+----------------+-------------------+-------------+-----------+**  **| booking\_number | ticket\_number | contact | flight |**  **+----------------+-------------------+-------------+-----------+**  **| 1 | 6ce8966c2268a6309 | 600510-6493 | BA1411250 |**  **+----------------+-------------------+-------------+-----------+**  **1 row in set (0.00 sec)** |
| **mysql> CALL reserve\_seats(6, 'BA1411250');**  **Query OK, 1 row affected (0.01 sec)**  **mysql> SELECT \* FROM booking;**  **+----------------+-------------------+-------------+-----------+**  **| booking\_number | ticket\_number | contact | flight |**  **+----------------+-------------------+-------------+-----------+**  **| 1 | 6ce8966c2268a63a6 | 600510-6493 | BA1411250 |**  **| 2 | NULL | NULL | BA1411250 |**  **+----------------+-------------------+-------------+-----------+**  **2 rows in set (0.00 sec)** |  |
|  | **mysql> SELECT \* FROM booking;**  **+----------------+-------------------+-------------+-----------+**  **| booking\_number | ticket\_number | contact | flight |**  **+----------------+-------------------+-------------+-----------+**  **| 1 | 6ce8966c2268a6309 | 600510-6493 | BA1411250 |**  **+----------------+-------------------+-------------+-----------+**   1. **row in set (0.00 sec)** |

1. No, the booking is not visible in session B. This is because of the issued command
2. **SET autocommit=0**which, according to the MySQL reference manual *“changes to transaction-safe tables (such as those for InnoDB, […]) are not made permanent immediately. You must use COMMIT to store your changes to disk…”*. So, changes are only stored locally until they are commited, and since no commit has been made, session B cannot see the changes in the database.

|  |  |
| --- | --- |
| **SESSION A** | **SESSION B** |
| **mysql> SELECT \* FROM booking;**  **+----------------+-------------------+-------------+-----------+**  **| booking\_number | ticket\_number | contact | flight |**  **+----------------+-------------------+-------------+-----------+**  **| 1 | 6ce8966c2268a63a6 | 600510-6493 | BA1411250 |**  **| 2 | NULL | NULL | BA1411250 |**  **+----------------+-------------------+-------------+-----------+**  **2 rows in set (0.00 sec)**  **mysql> CALL add\_existing\_passenger(2, '810217-8485');**  **Query OK, 1 row affected (0.01 sec)** |  |
|  | **mysql> CALL add\_existing\_passenger(2, '400316-1354');**  **ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction** |

1. A deadlock occours when session B tries to write to the table. After some 60 seconds of trying to write to the table, the timeout is exceeded.