Martin Forsberg
Emilia Bylund Månsson

Assignment 4

Task 2 – 7 as seen in code above

Task 8

a) How can you protect the credit card information in the database from hackers?

You can encrypt the credit card information or separate the credit card numbers in different files two make it harder for hackers to access all information necessary.

b) Give three advantages of using stored procedures in the database (and thereby execute them on the server) instead of writing the same functions in the frontend of the system (in for example java-script on a web-page)?

1. The performance is better compared to writing it in the front end, due to the quick response and the storage in executable form.
2. You can group the stored procedures and execute them all at once, this isn't possible to the same extent with for instance java-script. Instead of writing the same query over and over again the user can call the procedure. This can reduce code complexity, epecially if the same query is used in many parts of the code.
3. If the query has to be changed as the development evolves, the query only has to be changed in one place, which is more OOP.

Task 9

b) No, this is not visible in section B because we haven't written a commit yet. This means that the two transactions are isolated from each other.
c) Since the transactions are executed in isolation due to the lack of commit-statement, B can't see what happens in A and can't modify it. After A has committed, B will be able to see the transaction.

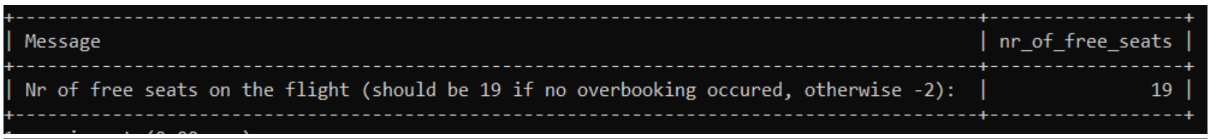
Task 10

a) No, overbooking did not occur due to the fact that we got an IF-statement that controls the number of passengers that the reservation is trying to book. The first transaction is then preventing the second from booking too many passengers.

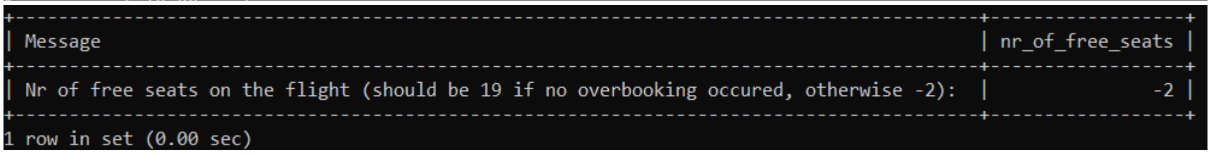
b) Overbooking is theoretically possible if both transactions are going through the IF-statement in addPayment before the trigger that generates the ticket number starts. If the code inside the IF-statement is too slow and B is going through the condition before the execution of the ticket then an overbooking is possible.
   i) IF-statement starts for transaction A
   ii) IF-statement starts for transaction B and B is "inside" the IF-statement
   iii) Ticket-number is generated by the trigger for A
   iv) Ticket-number is generated by the trigger for B
   → Overbooking occurs

c) By adding SELECT SLEEP (15) the second call to addPayment is able to enter the if-statement before the first one is getting the ticket and actually booking, this will thereby create an overbooked flight.

Martin Forsberg
Emilia Bylund Månsson



```
+----------------------------------------------------------------------------------+------------------+
| Message                                                                          | nr_of_free_seats |
+----------------------------------------------------------------------------------+------------------+
| Nr of free seats on the flight (should be 19 if no overbooking occured, otherwise -2):  |               19 |
+----------------------------------------------------------------------------------+------------------+
```



```
+----------------------------------------------------------------------------------+------------------+
| Message                                                                          | nr_of_free_seats |
+----------------------------------------------------------------------------------+------------------+
| Nr of free seats on the flight (should be 19 if no overbooking occured, otherwise -2):  |               -2 |
+----------------------------------------------------------------------------------+------------------+
1 row in set (0.00 sec)
```

d) When doing a *LOCK TABLES ticket READ* the other transaction is not able to read and use the ticket table and therefore a booking can't get through. The second transaction then needs to wait until the table has been UNLOCKED again and then it tries to book, but this time we already have a booking and not enough seats which will cancel the booking.

<u>Index</u>

A secondary index is used to get access to specific data faster. By having a table's id:s in a separate, already sorted table, that points to the original tuple, the user does not have to search through a random array of indexes to find the data they are looking for. This will increase the efficiency of querying the database.

In this project we could have used a secondary index for the passengers fo example. We would create another table only consisting of the p-number och all passengers in the passenger table. The new table would be sorted by size and be a fk of the p-number in passenger. If we assume MySQL uses an algorithm to search for a tuple and not just try at random, this method should be faster.