PROJECT REPORT(732A57)

Bryan Air Database

Saman Zahid (Samza595)
Rabnawaz Jansher (rabsh696)

8. Answer the following theoretical questions:

a) How can you protect the credit card information in the database from hackers?
In order to protect the credit card information from hackers, the following measures should be taken:

1. Every account should be password protected, that is there should be no user without password and the password should be strong and password hashes should be encrypted.
2. There should be defined privileges/roles for every user. Only a limited and trusted users should have access to credit card information.
3. Encrypt the credit card number.
4. Web application and database firewalls should be used to avoid any sort of attack from web such as SQL injection.
5. Database activity monitoring should be done in order to monitor any sort of unusual activity on database server.

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 front-end of the system (in for example java-script on a web-page)?

The advantages of using stored procedures over writing the function on front are as follows:

1. Pre-compiled: Stored procedures are pre-compiled objects. That is it is compiled only first time into the memory then it is used from memory afterwards.
2. Optimization: Stored procedures are bunch of queries which is dynamically optimized based on the input provided to it on runtime.
3. Reusability: In case of database change, it is easier to reuse the same stored procedures by making small changes. It is also easier to use in case of update.
4. Security: A well written stored procedure protects against SQL injection.

9. Open two MySQL sessions. We call one of them A and the other one B. Write *START TRANSACTION;* in both terminals.

a) In session A, add a new reservation.

Terminal A

```
mysql> start transaction;
Query OK, 0 rows affected (0.06 sec)

mysql> CALL addReservation("MIT","HOB",2010,1,"Monday","09:00:00",1,@f);
Query OK, 1 row affected (0.01 sec)

mysql> select * from reservation;
+--------------------+-------------------+--------+
| reservation_number | reservation_seats | flight |
+--------------------+-------------------+--------+
|                  1 |                 3 |      1 |
|                  2 |                 3 |      1 |
|                  3 |                 1 |      1 |
|                  4 |                 1 |      1 |
|                  5 |                 1 |      1 |
+--------------------+-------------------+--------+
5 rows in set (0.00 sec)
```

Terminal B

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from reservation;
+--------------------+-------------------+--------+
| reservation_number | reservation_seats | flight |
+--------------------+-------------------+--------+
|                  1 |                 3 |      1 |
|                  2 |                 3 |      1 |
|                  3 |                 1 |      1 |
|                  4 |                 1 |      1 |
+--------------------+-------------------+--------+
4 rows in set (0.00 sec)
```

b) Is this reservation visible in session B? Why? Why not?

No, the reservation in session "A" is not visible in session "B". It is because of the concept isolation that is in order to maintain concurrency. Here, transaction B starts before the end of transaction A thus the modifications made there are not visible in session B.

c) What happens if you try to modify the reservation from A in B? Explain what happens and why this happens and how this relates to the concept of isolation of transactions.

Transactions will not be modified in terminal B from terminal A until terminal A commits the modifications, because of repeatable read isolation. MySQL uses repeatable read isolation by default which introduces shared locks on all data used by multiple transactions. So in this case when terminal A tries to modify the data in terminal B, it won't be allowed (lock timeout occurs) as this data is used by transaction in terminal A and thus is locked until the transaction is completed. But if terminal A commits the modifications, then it will be visible in terminal B.

There is one more scenario where newly added data in terminal B can be modified by terminal A, its because of phantom reads which is a flaw in repeatable read.

10. Is your BryanAir implementation safe when handling multiple concurrent transactions? Let two customers try to simultaneously book more seats than what are available on a flight and see what happens. This is tested by executing the testscripts available on the course-page using two different MySQL sessions. *Note that you should not use explicit transaction control unless this is your solution on 10c.*

a) Did overbooking occur when the scripts were executed? If so, why? If not, why not?

No, overbooking will not occur. It is because when two customers try to book simultaneously, then one customer's payment procedure will run before the other customer's payment procedure. The payment procedures will not run in parallel thus avoiding the overbooking.

b) Can an overbooking theoretically occur? If an overbooking is possible, in what order must the lines of code in your procedures/functions be executed.

Yes. Overbooking can occur if both customers create reservation, then both customers dd contact information and then both are able to make payment at the same time.

c) Try to make the theoretical case occur in reality by simulating that multiple sessions call the procedure at the same time. To specify the order in which the lines of code are executed use the MySQL query *SELECT sleep(5);* which makes the session sleep for 5 seconds. Note that it is not always possible to make the theoretical case occur, if not, motivate why.

Running this procedure on multiple sessions can cause overbooking:

```
Delimiter //
CREATE PROCEDURE myProcedure(reservation_nr INT)
BEGIN

DECLARE flightnumber INT;
DECLARE total_price DOUBLE;


SELECT reservation_seats INTO reserved_seats FROM reservation WHERE reservation_number=reservation_nr;

SELECT c.passport_number as passport_number from passenger p inner join contact c on p.passport_number =
c.passport_number where reservation_number = reservation_nr;

SET flightnumber = (SELECT flight FROM reservation WHERE reservation_number=reservation_nr);
SELECT calculateFreeSeats(flightnumber) INTO free_seats;


IF passport_number IS NULL
THEN
SELECT "The reservation has no contact yet";
END IF;

IF free_seats < reserved_seats
THEN
SELECT 'There are not enough seats available on the flight anymore, deleting reservation.';


ELSE

SELECT sleep(5);

INSERT INTO credit_card(card_number,holder_name) VALUES(credit_card_number, cardholder_name)

ON DUPLICATE KEY UPDATE card_number = credit_card_number;

SET total_price = calculatePrice(flightnumber);
INSERT INTO booking (reservation_number,total_price,card_number,flight_number,status) VALUES (reservation_nr,
total_price, credit_card_number,flightnumber,TRUE);

INSERT INTO ticket (booking_number, passenger_number) values(reservation_nr,passport_number);

END IF;

END;
//
Delimiter ;
```

d) Modify the testscripts so that overbookings are no longer possible using (some of) the commands START TRANSACTION, COMMIT, LOCK TABLES, UNLOCK TABLES, ROLLBACK, SAVEPOINT, and SELECT…FOR UPDATE. Motivate why your solution solves the issue, and test that this also is the case using the sleep implemented in 10c. Note that it is not ok that one of the sessions ends up in a deadlock scenario. Also, try to hold locks on the common resources for as short time as possible to allow multiple sessions to be active at the same time.

```
/**************************************************************************************
***
 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);
SELECT SLEEP(5);


SELECT "Making payment, supposed to work for one session and be denied for the other" as "Message";

LOCK TABLES contact read,  weekly_schedule read, route read, airport read, weekday read, year read;

LOCK TABLES flight write, reservation write, passenger write, booking write, credit_card write, ticket write;

start transaction;
```

CALL addPayment (@a, "Sauron",7878787878);
commit;
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 and flight_no=1) as "nr_of_free_seats";

Terminal A

```
mysql> source /Users/rabnawazjansher/Desktop/database_project/q10.sql;
+-------------------------------------------------------------------------------------+
| Message                                                                             |
+-------------------------------------------------------------------------------------+
| Testing script for Question 10, Adds a booking, should be run in both terminals |
+-------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

+-----------------------------------+
| Message                           |
+-----------------------------------+
| Adding a reservations and passengers |
+-----------------------------------+
1 row in set (0.00 sec)

+--------------------+
| MESSAGE            |
+--------------------+
| Reservation seats |
+--------------------+
1 row in set (0.01 sec)

Query OK, 1 row affected, 3 warnings (0.08 sec)

Query OK, 1 row affected, 1 warning (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)
```

```
Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected, 1 warning (0.00 sec)

+----------+
| SLEEP(5) |
+----------+
|        0 |
+----------+
1 row in set (5.00 sec)

+-------------------------------------------------------------------------------+
| Message                                                                       |
+-------------------------------------------------------------------------------+
| Making payment, supposed to work for one session and be denied for the other |
+-------------------------------------------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected, 1 warning (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

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

Terminal B

```
mysql> source /Users/rabnawazjansher/Desktop/database_project/q10.sql;


+-------------------------------------------------------------------------------+
| Message                                                                       |
+-------------------------------------------------------------------------------+
| Testing script for Question 10, Adds a booking, should be run in both terminals |
+-------------------------------------------------------------------------------+
1 row in set (0.00 sec)

+-------------------------------------+
| Message                             |
+-------------------------------------+
| Adding a reservations and passengers |
+-------------------------------------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.00 sec)


+----------+
| SLEEP(5) |
+----------+
|        0 |
+----------+
1 row in set (5.00 sec)

+-------------------------------------------------------------------------------+
| Message                                                                       |
+-------------------------------------------------------------------------------+
| Making payment, supposed to work for one session and be denied for the other |
+-------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

+-----------------------------------+
| Message                           |
+-----------------------------------+
| Please add a contact before payment |
+-----------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

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

Q.11) Identify one case where a secondary index would be useful. Design the index, describe and motivate your design. (Do not implement this.)

Secondary Index will be useful if we need to perform search with respect to the destination or arrival airport's country name. As primary index in the table is airport code thus the data is unsorted with respect to country name and binary search will not be performed. Linear search will decrease the performance. Thus secondary index will be effective in this case.