

Exercises 11

The purpose of these exercises is to get some experience with using, and reasoning about, transactions. To disable autocommit in MySQL type: `set autocommit=0;` in the Query Browser, or call `conn.setAutoCommit(false)` if you are connecting via JDBC. The default isolation level is `REPEATABLE READ` – this can be changed using the standard syntax, e.g., `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;` MySQL uses row level locking by default for InnoDB tables. Whole tables can be locked using the syntax:

```
LOCK TABLES T1 READ, T2 WRITE, ...
```

Tables must be explicitly unlocked at the end of a transaction using `UNLOCK TABLES.`

Fun with transactions

Start two concurrent MySQL connections (e.g. two different windows or tabs in Query Browser).

a) Create the Authors relation, and issue updates as described below. Explain the observed behavior.

In the following we consider the below transactions on the Authors(`auID`,`name`) relation.

Time	User A	User B
1	INSERT INTO Authors VALUES (42,'Donald Knuth');	
2		INSERT INTO Authors VALUES (43,'Guy Threepwood');
3		DELETE FROM Authors WHERE name LIKE 'Don%';
4		INSERT INTO Authors VALUES (44,'Donald E. Knuth');
5	DELETE FROM Authors WHERE name LIKE 'Guy%';	
6	COMMIT;	
7		COMMIT;

b) Suppose that Authors is initially empty, that the transactions are run at isolation level `READ COMMITTED`, and that the commands are issued in the order indicated above. What is the content of Authors after the execution?

c) Suppose that Authors is initially empty. What are the possible contents of Authors after each *serial* execution of the two transactions?

d) Declare `auID` a unique key. Then issue a sequence of insertions that result in a deadlock, even at isolation level `READ COMMITTED`.

Problem 4 from exam in Databasesystemer, June 2005 (15 %)

The relation `Seats(seatID, class, reserved)` is used to handle seat reservations in an airplane. It contains one tuple for each seat, and the attribute `reserved` is 0 or 1 depending on whether a seat is free or booked. 15 minutes before departure, the booking of business class seats is closed, by transferring any free business class seats to “economy plus” customers. The below transaction (written in Oracle SQL) makes the transfer of seats, using the two relations `FreeBusinessSeats(seatID, reserved)` and `Upgrades(seatID)` to store intermediate results.

1. `DELETE FROM FreeBusinessSeats;`
2. `DELETE FROM Upgrades;`
3. `INSERT INTO FreeBusinessSeats (SELECT seatID, reserved FROM Seats
WHERE class='business' AND reserved=0);`
4. `INSERT INTO Upgrades (SELECT *
FROM (SELECT seatID FROM Seats
WHERE class='economy plus' AND reserved=1
WHERE rownum<=(SELECT COUNT(*) FROM FreeBusinessSeats));`
5. `UPDATE FreeBusinessSeats SET reserved=1
WHERE rownum<=(SELECT COUNT(*) FROM Upgrades);`
6. `UPDATE Seats SET reserved=0 WHERE seatID IN (SELECT * FROM Upgrades);`
7. `UPDATE Seats SET reserved=1
WHERE (seatID,1) IN (SELECT seatID,reserved FROM FreebusinessSeats);`

Explanation: The first two lines delete any old intermediate results. Line 3 inserts the free business class seats in the relation `FreeBusinessSeats`. Line 4 chooses reservations from “economy plus” that are to be upgraded. The number of upgrades is kept below the number of free seats by use of the `rownum` variable, which returns the row number of the current row in the relation. Line 5 marks the right number of free seats in `FreeBusinessSeats` as reserved. In line 6 and 7, the information on the new reservations are transferred to the `Seats` relation.

a) Assume that the above transaction runs at SQL isolation level `READ COMMITTED`. Argue that if, at the same time, a reservation is made for a business class seat (i.e. a transaction that changes a value of `reserved` from 0 to 1), there may be a double booking, that is, the number of reserved seats is smaller than the number of passengers.

Because of the above problem, it seems like a good idea to consider a higher isolation level. We consider SQLs `REPEATABLE READ` og `SERIALIZABLE`, as well as “snapshot isolation” described in the article *A Critique of ANSI SQL Isolation Levels*.

b) Consider for each of the three mentioned isolation levels whether a double booking may occur. Argue for your answer.