Database Administration and Tuning
Mini-Project 2 - Report

Henrique Rocha - 68621
Ludijor Barros - 68626
Fábio Martins - 71073

May 17, 2015

1 Transaction Isolation Levels

Consider a relational database for information related to cheese products, with the following tables:

CHEESE(cheeseID, type, producer, calories, proteins)
REGION(regionID, name, country)
PRODUCTION (productionID, cheeseID, season, amount)
PROVENANCE (productionID, regionID)

Assume that the following three stored procedures can run concurrently in a given application that is supported by the relational database:

1. **insert_cheese**: creates new records in the CHEESE table, for new cheeses that are produced. This procedure uses only the CHEESE table.

2. **update_production**: inserts new records, or modifies existing PRODUCTION and PROVENANCE records. This procedure writes to the PRODUCTION and PROVENANCE tables, updates the tuples in the PROVENANCE table, and may be reading from the REGION and CHEESE tables.

3. **delete_region**: deletes a given region from the REGION table.

1.1

Give a scenario that leads to a possible dirty read in the concurrent execution of operations from this group of stored procedures, or explain why a dirty read cannot happen in this group of stored procedures.

Consider two transactions both using **update_production** stored procedure over the same PRODUCTION tuples.

T1	T2
BEGIN	
R(REGION)	
R(CHEESE)	
R(PRODUCTION_X)	
W(PRODUCTION_X)	
	BEGIN
	R(REGION)
	R(CHEESE)
	R(PRODUCTION_X)
	W(PRODUCTION_X)
	R(PROVENANCE)
	W(PROVENANCE)
	COMMIT
R(PROVENANCE)	
ABORT	

Figure 1: An example of dirty read

T2 commits and T1 aborts after, rolling back its changes, the PRODUCTION tuples that T2 read from T1 are dirty reads because they reflect changes of an aborted transaction (T1).

1.2

Give a scenario that leads to a possible non-repeatable read in the concurrent execution of operations from this group of stored procedures, or explain why a non-repeatable read cannot happen in this group of stored procedures.

Consider that transaction one executes two **update_production** stored procedures over the same REGION tuple, the second transaction deletes this REGION tuple.

T1	T2
BEGIN	
R(REGION_X)	
R(CHEESE)	
R(PRODUCTION)	
W(PRODUCTION)	
R(PROVENANCE)	
W(PROVENANCE)	
	BEGIN
	DELETE(REGION_X)
	COMMIT
R(REGION_X)	
R(CHEESE)	
R(PRODUCTION)	
W(PRODUCTION)	
R(PROVENANCE)	
W(PROVENANCE)	
COMMIT	

Figure 2: An example a non-repeatable read

The second time the REGION tuple is read, it doesn't exists, leading to a non-repeatable read.

1.3

Give an example of a possible overwriting of uncommitted data in the concurrent execution of operations from this group of stored procedures, or explain why a phantom read cannot happen in this group of stored procedures.

Consider two transactions both using **update_production** stored procedure over the same PRODUCTION tuples.

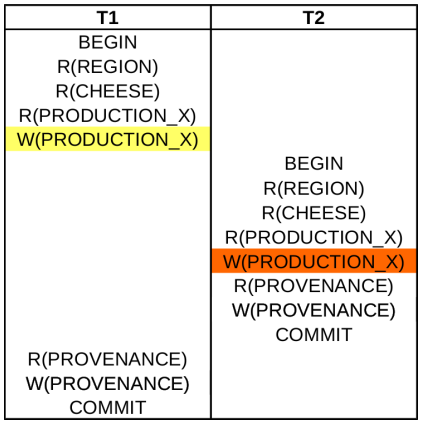

Figure 3: An example overwriting of uncommitted data

T2 commits and T1 PRODUCTION tuple value is lost.

1.4

Indicate what transaction isolation level would you use for executing each of the three procedures above, and why? For each procedure you should use the least restricted transaction isolation level that ensures correctness.

2 Concurrency Control

3 Recovery System

4 Schema and Index Tuning

5 Query Tuning

6 Database Tuning