Transaction processing in Oracle DBMS

(Snapshot isolation protocol)

Transaction scope

Transaction starts from the first executable SQL statement

Transaction ends with either `COMMIT`, `ROLLBACK`, or DDL statement: `CREATE`, `DROP`, `ALTER`

Transaction also ends with disconnection (auto-commit) or process failure (`ROLLBACK`)

Read consistency levels

Statement-level read consistency

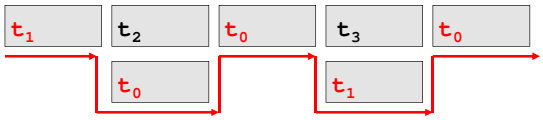
All data read by a query (`SELECT`) come from a single point in time (`READ COMMITTED`)

Transaction-level read consistency

All queries in a transaction read data that come from a single point in time (`SERIALIZABLE`)

Rollback/Undo segments

Rollback segments consist of data blocks that contain the old values of data that have been changed by the uncommitted or recently committed transactions



Segments read by a transaction T with a timestamp t_i such that $t_2, t_3 > t_i > t_0, t_1$

Locking

Locking is performed automatically by the system

It is also possible to lock data items manually

There are two types of locks:
(1) shared,
(2) exclusive

A transaction holds exclusive row locks for all rows inserted, updated, or deleted within the transaction

Locking

Transactions use row-level locking to ensure database consistency

The system releases all locks acquired by a transaction when the transaction either commits or rolls back

A transaction must wait if it tries to change a row updated by an uncommitted transaction

Locking

A deadlock occurs when two or more users are waiting for data locked by each other

The system automatically detects a deadlock and rolls back one of the statements involved in the deadlock

It is possible to perform manual locking with `LOCK TABLE` statement

Setting isolation levels in Oracle DBMS

```
SET TRANSACTION ISOLATION READ ONLY
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```


READ COMMITTED isolation level

At `READ COMMITTED` isolation level each query executes with respect to its own materialized view time, thereby permitting nonrepeatable reads and phantoms for multiple executions of a query

`READ COMMITTED` isolation level is recommended when few transactions are likely to conflict

`READ COMMITTED` is a default isolation level

READ COMMITTED isolation level

Transaction 1	Transaction 2
	SELECT budget FROM Department WHERE name = 'SALES'
	2000
UPDATE DEPARTMENT SET budget = budget + 1000 WHERE NAME = 'Sales';	
	SELECT budget FROM Department WHERE name = 'SALES'
	2000

READ COMMITTED isolation level

Transaction 1	Transaction 2
	SELECT budget FROM Department WHERE name = 'SALES'
	2000
UPDATE DEPARTMENT SET budget = budget + 1000 WHERE name = 'Sales';	
COMMIT	
	SELECT budget FROM Department WHERE name = 'SALES'
	3000

READ COMMITTED isolation level

Transaction 1	Transaction 2
SELECT budget FROM DEPARTMENT WHERE name = 'SALES'	
3000	
	UPDATE DEPARTMENT SET budget = budget + 10 WHERE name = 'SALES'
UPDATE DEPARTMENT SET budget = budget + 1000 WHERE name = 'Sales';	
Wait	
	COMMIT
...	...

READ COMMITTED isolation level

Transaction 1	Transaction 2
...	...
SELECT budget FROM DEPARTMENT WHERE name = 'SALES'	
4010	

READ COMMITTED isolation level

Transaction 1	Transaction 2
SELECT budget FROM DEPARTMENT WHERE name = 'SALES'	
3000	
	UPDATE DEPARTMENT SET budget = budget + 10 WHERE name = 'SALES'
UPDATE DEPARTMENT SET budget = budget + 1000 WHERE name = 'Sales';	
Wait	
	ROLLBACK
...	...

READ COMMITTED isolation level

Transaction 1	Transaction 2
...	...
SELECT budget FROM DEPARTMENT WHERE name = 'SALES'	
4000	

READ COMMITTED isolation level

Transaction 1	Transaction 2
UPDATE DEPARTMENT SET budget = (SELECT budget FROM DEPARTMENT WHERE name = 'Sales') WHERE name = 'Finance';	
	UPDATE DEPARTMENT SET budget = 500 WHERE name = 'Sales'; COMMIT;
UPDATE DEPARTMENT SET budget = budget + (SELECT budget FROM DEPARMENT WHERE name = 'Sales') WHERE name = 'Finance'; COMMIT;	Transaction 1 corrupts a database, because budget(Finance) <> 2 *(budget(Sales))

SERIALIZABLE isolation level

If transaction T running at SERIALIZABLE isolation level tries to update or delete data modified by a transaction that commits after the serializable transaction T began then the system aborts transaction T

If a serializable transaction fails then it is possible to:
(1) commit the work executed to that point,
(2) execute additional (but different) statements,
(3) rollback the entire transaction

SERIALIZABLE isolation level

Transaction 1	Transaction 2
	UPDATE DEPARTMENT SET budget = budget + 10 WHERE name = 'SALES'
UPDATE DEPARTMENT SET budget = budget + 1000 WHERE name = 'Sales';	
Wait	
	COMMIT
ERROR at line 2: ORA-08177: can't serialize access for this transaction	

SERIALIZABLE isolation level

Transaction 1	Transaction 2
	SELECT budget FROM Department WHERE name = 'SALES'
	2000
UPDATE DEPARTMENT SET budget = budget + 1000 WHERE name = 'Sales';	
COMMIT	
	SELECT budget FROM Department WHERE name = 'SALES'
	2000

References

http://www.uow.edu.au/~jrg/235/HOMEWORK
10 How to control database transactions ?