CSCI235 Database Systems

Transaction Processing in Oracle DBMS

Dr Janusz R. Getta

School of Computing and Information Technology - University of Wollongong

Transaction Processing in Oracle DBMS

Outline

Transaction scope

Transaction starts from the first executable SQL statement after connection, `COMMIT`, or `ROLLBACK` statement

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

Transaction also ends with disconnection (auto-commit or auto-rollback depending on default or set up parameters) or process failure followed by automatic `ROLLBACK` statement

Transaction Processing in Oracle DBMS

Outline

Isolation levels and read consistency levels

Oracle DBMS implements three isolation levels

READ COMMITTED - a transaction may exhibit:

- non-repeatable read phenomenon,
- phantom phenomenon

SERIALIZABLE - a transaction may exhibit:

- none of the phenomena

READ ONLY - a transaction consists only of `read` operations

At READ COMMITTED isolation level all data read by a query (`SELECT` statement) come from a single point in time (statement-level read consistency)

At SERIALIZABLE isolation level all queries (`SELECT` statements) in a transaction read data that come from a single point in time (transaction-level read consistency)

Transaction Processing in Oracle DBMS

Outline

Rollback/undo segments

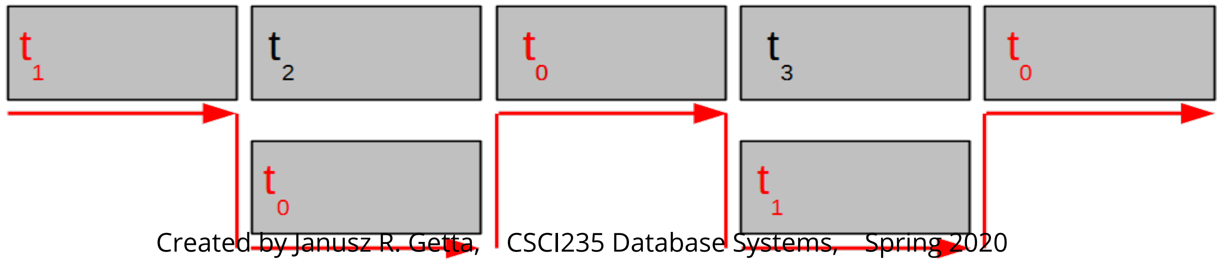
Rollback/undo segments consist of data blocks that contain the old values of data that have been changed by the uncommitted or recently committed transactions

Each time a row in a data a block is changed a new version of the block together with a timestamp is added to rollback/undo segment

A new version of a data block obtains a timestamp higher than the previous version

A data block may have many versions updated in the different moments in time

See below the data blocks read by a transaction T with a timestamp ti such that t2 , t3 > ti > t0 , t1



Transaction Processing in Oracle DBMS

Outline

READ COMMITTED versus SERIALIZABLE levels

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

If a 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:

- commit the work executed to that point,
- execute additional (but different) statements,
- rollback the entire transaction

Transaction Processing in Oracle DBMS

Outline

Locking

Locking is performed automatically by the system

It is also possible to lock data items manually

There are two types of locks:

- shared (read locks),
- exclusive (write locks)

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

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

The system automatically converts a lock of lower restrictiveness to one of higher restrictiveness when it is appropriate

For example, `SELECT` statement with `FOR UPDATE` clause initially locks the rows in shared mode, then when `UPDATE` is performed the locks are upgraded to exclusive locks

Locking

Both READ COMMITTED and SERIALIZABLE transactions use row-level locking to ensure database consistency

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

If a transaction rollbacks then the waiting transactions regardless of its isolation mode can proceed to change the previously locked row

If a transaction commits than any other transaction at READ COMMITTED level waiting for a locked row can proceed to change the previously locked row

If a transaction commits than any other transaction at SERIALIZABLE level waiting for a locked row fails with the error "Cannot serialize access", because the other transaction has committed a change that was made since the serializable transaction began

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 of entire relational table with `LOCK TABLE` statement

Transaction Processing in Oracle DBMS

Outline

READ COMMITTED isolation level

Setting READ COMMITTED isolation level is performed at the beginning of transaction with the following statement

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

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

Sample concurrent processing of database transactions at READ COMMITTED isolation level that does not corrupt a database

Concurrent processing of database transactions at 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
```

Transaction 2 cannot read uncommitted modifications

READ COMMITTED isolation level

Sample concurrent processing of database transactions at READ COMMITTED isolation level that does not corrupt a database

Concurrent processing of database transactions at 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`

Transaction 2 can only read committed modifications

READ COMMITTED isolation level

Sample concurrent processing of database transactions at READ COMMITTED isolation level that does not corruptt a database

```
                 Concurrent processing of database transactions at 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;

SELECT budget
FROM DEPARTMENT
WHERE name = 'SALES'

4010
```

READ COMMITTED isolation level

```
  Transaction 1                  Transaction 2            Processing at READ COMMITTED isolation level

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;

SELECT budget
FROM DEPARTMENT
WHERE name = 'SALES'

4010
```

Transaction 1 must wait until Transaction 2 either commits or rolls back its update

READ COMMITTED isolation level

Sample concurrent processing of database transactions at READ COMMITTED isolation level that does not corrupt a database

```
  Transaction 1                  Transaction 2          Processing at READ COMMITTED isolation level

SSELECT 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;

SELECT budget
FROM DEPARTMENT
WHERE name = 'SALES'

4000
```

READ COMMITTED isolation level

```
  Transaction 1                Transaction 2         Processing at READ COMMITTED isolation level

SSELECT 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;

SELECT budget
FROM DEPARTMENT
WHERE name = 'SALES'

4000
```

Transaction 1 must wait until Transaction 2 either commits or rolls back its update

READ COMMITTED isolation level

A sample processing of database transactions at READ COMMITTED isolation level that corrupts a database

```
                                         Processing at 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 a budget of department `Finance` is not equal to 2 * budget of department `Sales`

Transaction Processing in Oracle DBMS

Outline

SERIALIZABLE isolation level

If a 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`

A sample processing of database transactions at SERIALIZABLE isolation level that fails one of the transactions

Processing database transactions at 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

A sample processing of database transactions at SERIALIZABLE isolation level that ends successfully for both transactions

Processing database transactions at 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`

Transaction 1 creates a new version of a row for Sales department

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 22.5 Concurrency Control and Recovery in Oracle, Pearson Education Ltd, 2015