U O W

Transaction Processing in ANSI SQL

(Phenomenons and Isolation levels)

CSIT882: Data Management Systems

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Outline

- Phenomena
- Isolation levels

Phenomena

Dirty read phenomenon

Read operations may access *dirty data*, i.e. data written by uncommitted transactions

Non-repeatable read phenomenon

Different reads by a single transaction to the same data will not be repeatable, i.e. they may return different values

Phantom phenomenon

A set of rows that transaction reads once might be different from a set of rows when the transaction attempts to read it again

Dirty read phenomenon

Read operations may access dirty data, i.e. data written by uncommitted transactions

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'`

	3000
`ROLLBACK;`	
	???

Non-repeatable read phenomenon

Different reads by a single transaction to the same data will not be repeatable

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
	???

Phantom phenomenon

A set of rows that transaction reads once might be different from a set of rows when the transaction attempts to read it again

Transaction 1	Transaction 2
	`SELECT count(*)` `FROM DEPARTMENT`
	`20`
`DELETE DEPARTMENT` `WHERE NAME = 'SALES';`	
`COMMIT;`	
	`SELECT count(*)` `FROM DDEPARTMENT;`
	`19`

Outline

- Phenomena
- Isolation levels

Isolation levels

ANSI standard of SQL provides four levels of isolation for processing of database transactions

Isolation levels are equivalent to correctness levels

Isolation levels are defined in terms of several possible phenomena, or weird hard-to-explain occurrences of operations

Isolation levels

READ UNCOMMITTED

READ COMMITTED

REPEATABLE READ

SERIALIZABLE

READ UNCOMMITTED level

At READ UNCOMMITED isolation level a transaction may exhibit:

dirty read phenomenon,

non-repeatable read phenomenon,

phantom phenomenon

READ COMMITTED level

At READ COMMITTED isolation level a transaction may exhibit:

non-repeatable read phenomenon,

phantom phenomenon

REPEATABLE READ level

At REPEATABLE READ isolation level a transaction may exhibit:

phantom phenomenon

SERIALIZABLE level

At SERIALIZABLE isolation level a transaction may exhibit:

none of the phenomena

Isolation levels

Level	Dirty Read	Nonrepeatable Read	Phantom
READ UNCOMMITTED	Possible	Possible	Possible
READ COMMITTED	Not possible	Possible	Possible
REPEATABLE READ	Not possible	Not possible	Possible
SERIALIZABLE	Not possible	Not possible	Not possible

References

Elmasri R., Navathe S., Fundamentals of Database Systems, 6th edition, chapter 21.6 Transaction Support in SQL, pp. 774-776

U O W

Transaction Processing in ANSI SQL

(Setting isolation levels)

CSIT882: Data Management Systems

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Setting isolation levels in ANSI SQL

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

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

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

READ UNCOMMITTED isolation level

Transaction T1	Transaction T2
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL READ UNCOMMITTED; START TRANSACTION;
SELECT budget FROM DEPARTMENT WHERE name='School of Arts';	
700.0	
	UPDATE DEPARTMENT SET budget=budget+100 WHERE name='School of Arts';
SELECT budget FROM DEPARTMENT WHERE name='School of Arts';	
800.0	
	ROLLBACK;

READ COMMITTED isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL READ COMMITTED; START TRANSACTION;
	SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';
	700.0
UPDATE DEPARTMENT SET budget = budget + 100 WHERE name = 'School of Arts';	
	SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';
	700.0

READ COMMITTED isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL READ COMMITTED; START TRANSACTION;
	SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';
	700.0
UPDATE DEPARTMENT SET budget = budget + 100 WHERE name = 'School of Arts';	
COMMIT;	
	SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';
	800.0

READ COMMITTED isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL READ COMMITTED; START TRANSACTION;
	SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';
	700.0
UPDATE DEPARTMENT SET budget = budget + 100 WHERE name = 'School of Arts';	
	UPDATE DEPARTMENT SET budget = budget + 100 WHERE name = 'School of Arts';
	Wait …

READ COMMITTED isolation level

Transaction 1	Transaction 2
...	...
SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';	
800.0	
COMMIT;	
	SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';
	900.0
	COMMIT;

READ COMMITTED isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL READ COMMITTED; START TRANSACTION;
SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';	
700.0	
	UPDATE DEPARTMENT SET budget = budget + 50 WHERE name = 'School of Arts';
UPDATE DEPARTMENT SET budget = budget + 100 WHERE name = 'School of Arts';	
Wait …	

READ COMMITTED isolation level

Transaction 1	Transaction 2
...	...
	ROLLBACK;
SELECT budget FROM DEPARTMENT WHERE name = 'School of Arts';	
800.0	
COMMIT;	

READ COMMITTED isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL READ COMMITTED; START TRANSACTION;
	SELECT budget FROM DEPARTMENT WHERE code = 'FIN';
	700.0
SELECT budget FROM DEPARTMENT WHERE code = 'SCIT';	
100.0	
...	...

READ COMMITTED isolation level

Transaction 1	Transaction 2
...	...
UPDATE DEPARTMENT SET budget = (SELECT budget FROM DEPARTMENT WHERE code = 'FIN') WHERE code = 'SCIT';	
SELECT budget FROM DEPARTMENT WHERE code = 'SCIT';	
700.0	
	UPDATE DEPARTMENT SET budget = budget + 50 WHERE code = 'FIN';

READ COMMITTED isolation level

Transaction 1	Transaction 2
...	...
	SELECT budget FROM DEPARTMENT WHERE code = 'FIN';
	750.0
	COMMIT;
UPDATE DEPARTMENT SET budget = budget + (SELECT budget FROM DEPARMENT WHERE code = 'FIN') WHERE code = 'SCIT'; COMMIT;	

Database is corrupted

Deadlock

Transaction 1	Transaction 2
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;` `START TRANSACTION;`	`SET TRANSACTION ISOLATIONAL LEVEL READ COMMITTED;` `START TRANSACTION;`

	`UPDATE DEPARTMENT`
	`SET budget = budget+500`
	`WHERE code = 'FIN';`

`UPDATE DEPARTMENT`
`SET budget = budget + 100`
`WHERE code = 'SCIT';`	

	`UPDATE DEPARTMENT`
	`SET total_staff = 45`
	`WHERE code = 'SCIT';`

	`Wait …`
...	...

Deadlock

Transaction 1	Transaction 2
...	...

`UPDATE DEPARTMENT`
`SET chair = 'Alice'`
`WHERE code = 'FIN';`	

`Wait …`	

`ERROR 1213 (40001): Deadlock`
`found when trying to get lock;`
`try restarting transaction`	

`ROLLBACK;`	
	`COMMIT;`

REPEATABLE READ isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL REPEATABLE READ; START TRANSACTION;
SELECT budget FROM DEPARTMENT WHERE code = 'SCIT';	
100.0	
	UPDATE DEPARTMENT SET budget = budget + 50 WHERE code = 'SCIT';
	COMMIT;
SELECT budget FROM DEPARTMENT WHERE code = 'SCIT';	
100.0	

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 processed so far,
(2) Process the additional (but different) statements,
(3) rollback the entire transaction

ERROR at line 2: ORA-08177: can't serialize access for this transaction

SERIALIZABLE isolation level

Transaction 1	Transaction 2
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; START TRANSACTION;	SET TRANSACTION ISOLATIONAL LEVEL SERIALIZABLE; START TRANSACTION;
	SELECT budget FROM DEPARTMENT WHERE code = 'Sales';
	700
	UPDATE DEPARTMENT SET budget = budget + 10 WHERE code = 'Sales';
UPDATE DEPARTMENT SET budget = budget + 10 WHERE code = 'Sales';	
Wait …	
...	...

ERROR at line 2: ORA-08177: can't serialize access for this transaction

SERIALIZABLE isolation level

Transaction 1	Transaction 2
...	...
	ROLLBACK;
SELECT budget FROM DEPARTMENT WHERE code = 'SCIT';	
710	

SERIALIZABLE isolation level

Transaction 1	Transaction 2
	SELECT budget FROM DEPARTMENT WHERE name = 'Sales';
	700
UPDATE DEPARTMENT SET budget = budget + 50 WHERE name = 'Sales';	
COMMIT;	
	SELECT budget FROM DEPARTMENT WHERE name = 'Sales';
	700.0

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

References

Elmasri R., Navathe S., Fundamentals of Database Systems, 6th edition, chapter 21.6 Transaction Support in SQL, pp. 774-776