DBMSs – seminar 4 - extra doc

Monitoring locks

SQL Server Profiler

- tracing tool that can be used to monitor events on the server;

- e.g., the user can create a *trace* for *Lock:Acquired* and *Lock:Released* events (indicating a lock has been acquired / released) and run it, then analyze its output;

sp_lock (feature in maintenance mode)

- returns info about all the locks held by sessions that are currently active / one or two sessions specified through input parameters (i.e., 0, 1 or 2 parameters);

- example:

```
SELECT @@SPID --53 (ID of current user process – see seminar 3, Databases course)

BEGIN TRAN
UPDATE Movie
SET Director = 'Vladimir Menshov'
WHERE MovieID = 1

ROLLBACK TRAN
       -- execute BEGIN.. UPDATE (don't rollback);

sp_lock 53    --sp_lock with one parameter
       --there's one X lock that has been granted:
       --SPID: 53    DBID: 23      Type: KEY     Mode: X              Status: GRANT
```

- Type = KEY, since there's an index on MovieID; remove the index => Type = RID (Row ID).

sys.dm_tran_locks

- 1 row per currently active request issued to the lock manager, i.e., the corresponding lock has been granted or is waiting to be granted;

- example:

```
BEGIN TRAN
UPDATE Movie
SET Director = 'Vladimir Menshov'
WHERE MovieID = 1

ROLLBACK TRAN
       -- execute BEGIN.. UPDATE (don't rollback);

SELECT resource_type, resource_database_id, request_mode, request_type, request_status,
request_session_id, request_owner_type, request_owner_id
FROM sys.dm_tran_locks
WHERE resource_database_id = DB_ID('MyImdb')
```

```
       -- resource_type: KEY       resource_database_id: 23    request_mode: X
          request_type: LOCK       request_status: GRANT       request_session_id: 53
          request_owner_type: TRANSACTION       request_owner_id: 106940
```

sys.dm_tran_active_transactions

- info about transactions;
Ex. join with sys.dm_tran_locks on *transaction_id*

Query Governor Cost Limit

- example – AdventureWorks

```
SELECT *
FROM Sales.SalesOrderDetail d INNER JOIN Sales.SalesOrderHeader h
  ON d.SalesOrderID = h.SalesOrderID
--query allowed to run; result set of about 120.000 rows
```

```
SET QUERY_GOVERNOR_COST_LIMIT 1
--query is not allowed to run, since its estimated cost exceeds the specified value:
The query has been canceled because the estimated cost of this query (2) exceeds the
configured threshold of 1.
```

Row Level Versioning (RLV) in SQL Server

Read Committed Snapshot & *Full Snapshot* – illustrated on examples:

- example – dirty read:

* row with MovieID = 1, Nominations = 28

* dirty read scenario:

T1	T2

UPDATE Movie
SET Nominations = Nominations + 2
WHERE MovieID = 1	

	SELECT * FROM Movie

* if T2 runs under READ COMMITTED, it's suspended until T1 releases its X lock;

* if T2 runs under READ UNCOMMITTED, it reads dirty data (value *30* for Nominations);

* if T2 runs under READ COMMITTED SNAPSHOT, it doesn't block, it doesn't read dirty data; it instead reads a value that has been previously committed: the most recent committed data as of the beginning of the SELECT statement, which is 28 in this particular case;

* if T2 runs under FULL SNAPSHOT, it doesn't block, it doesn't read dirty data; it instead reads a value that has been previously committed: the most recent committed data as of the beginning of the transaction, which is again 28;

* set the isolation level to READ COMMITTED SNAPSHOT:

```
ALTER DATABASE MyImdb
SET READ_COMMITTED_SNAPSHOT ON
```

- make sure T2 runs under READ COMMITTED;

* set the isolation level to FULL SNAPSHOT:

```
ALTER DATABASE MyImdb
SET ALLOW_SNAPSHOT_ISOLATION ON

SET TRANSACTION ISOLATION LEVEL SNAPSHOT -- for T2
```

* check if RCS / FS has been enabled on the DB:

```
SELECT database_id, is_read_committed_snapshot_on, snapshot_isolation_state
FROM sys.databases
WHERE database_id = DB_ID('MyImdb')
```

- example – non-repeatable read:

* row with MovieID = 1, Nominations = 28

* non-repeatable read scenario:

T1	T2	Read Nominations value under RCS	Read Nominations value under FS

	SELECT *
	FROM Movie
	WHERE MovieID = 1
		28	28

UPDATE Movie
SET Nominations = Nominations + 2
WHERE MovieID = 1
 	
COMMIT TRAN			

	SELECT *
	FROM Movie
	WHERE MovieID = 1
		30
		- since T2 doesn't acquire an S lock for its read operation, T1 is able to change the row and commit;
			28
			- T2 doesn't acquire an S lock for its read operation, so T1 is able to change the row and commit;
			- when the 2nd SELECT is executed, the most recent

		- when the 2nd SELECT is executed, the most recent committed value as of the beginning of the operation (i.e., the SELECT statement) is 30, i.e., the read is not repeatable under RCS!	committed value as of the beginning of the transaction is 28, i.e., the read is repeatable under FS!

- example – update conflict:

* row with MovieID = 1, Nominations = 28

* update conflict scenario:

T1	T2 – under Full Snapshot

	BEGIN TRAN
	SELECT Nominations
	FROM Movie
	WHERE MovieID = 1
	--returned value: 28

BEGIN TRAN
UPDATE Movie
SET Nominations = Nominations + 2
WHERE MovieID = 1
--nominations is now 30	

	UPDATE Movie
	SET Nominations = Nominations + 5
	WHERE MovieID = 1
	--T2 is suspended, since it's trying to acquire an X lock that conflicts with the X lock held by T1

COMMIT TRAN	=> error: Snapshot isolation transaction aborted due to update conflict... - T2 cannot proceed with its update; - T2 started when the value for Nominations was 28; - in the meantime, T1 changed that value to 30: another row version was created for the row and added to tempdb; - if T2 is allowed to proceed, it adds 5 to the most recent committed value as of the beginning of the transaction, which is 28: we would have a lost update (the update of T1 would be lost in this case);

PIVOT

- example – table ProductSales:

```
ProductType   DOW          Sales
Chocolate     Monday       1000
Chocolate     Monday       500
Chocolate     Tuesday      1000
IceCream      Monday       2000
IceCream      Tuesday      600
IceCream      Wednesday    300
IceCream      Wednesday     600
IceCream      Wednesday     300
```

```
SELECT *
FROM ProductSales
PIVOT (SUM(Sales)
       FOR DOW in ([Monday], [Tuesday], [Wednesday])
          ) t
```

```
ProductType   Monday        Tuesday       Wednesday
Chocolate     1500          1000          NULL
IceCream      2000          600           1200
```