Managing Data Concurrency

Objectives

After completing this lesson, you should be able to:

- Describe the locking mechanism and how Oracle manages data concurrency
- Monitor and resolve locking conflicts

Locks

- Prevent multiple sessions from changing the same data at the same time
- Are automatically obtained at the lowest possible level for a given statement
- Do not escalate



SQL> UPDATE employees

2 SET salary=salary+100

3 WHERE employee_id=100;

Transaction 2

SQL> UPDATE employees

2 SET salary=salary*1.1

3 WHERE employee_id=100;

Locking Mechanism

- High level of data concurrency:
 - Row-level locks for inserts, updates, and deletes
 - No locks required for queries
- Automatic queue management
- Locks held until the transaction ends (with the COMMIT or ROLLBACK operation)

Example

Assume that the rows for employee_id 100 and 101 reside in the same block:

Transaction 1

SQL> UPDATE employees

- 2 SET salary=salary+100
- 3 WHERE employee_id=100;

Transaction 2

SQL> UPDATE employees

- 2 SET salary=salary*1.1
- 3 WHERE employee id=101;

Data Concurrency

Time: Transaction 1 UPDATE hr.employees
SET salary=salary+100

WHERE employee id=100;

Transaction 2 UPDATE hr.employees

SET salary=salary+100

WHERE employee id=101;

09:00:00 Transaction 3 UPDATE hr.employees

SET salary=salary+100

WHERE employee id=102;

...

Transaction x UPDATE hr.employees

SET salary=salary+100

WHERE employee id=xxx;

DML Locks

Transaction 1

SQL> UPDATE employees 2 SET salary=salary*1.1 3 WHERE employee_id= 107; 1 row updated.

Transaction 2

```
SQL> UPDATE employees

2 SET salary=salary*1.1

3 WHERE employee_id= 106;
1 row updated.
```

Each DML transaction must acquire *two* locks:

- EXCLUSIVE row lock on the row or rows being updated
- Table lock (TM) in ROW EXCLUSIVE (RX) mode on the table containing the rows

Enqueue Mechanism

The enqueue mechanism keeps track of:

- Sessions waiting for locks
- Requested lock mode
- Order in which sessions requested the lock



Lock Conflicts

Transaction 1	Time	Transaction 2
UPDATE employees SET salary=salary+100 WHERE employee_id=100; 1 row updated.	9:00:00	UPDATE employees SET salary=salary+100 WHERE employee_id=101; 1 row updated.
UPDATE employees SET COMMISION_PCT=2 WHERE employee_id=101; Session waits enqueued due to lock conflict.	9:00:05	SELECT sum(salary) FROM employees; SUM(SALARY)692634
Session still waiting!	16:30:00	Many selects, inserts, updates, and deletes during the last 7.5 hours, but no commits or rollbacks!
<pre>1 row updated. Session continues.</pre>	16:30:01	commit;

Possible Causes of Lock Conflicts

- Uncommitted changes
- Long-running transactions
- Unnecessarily high locking levels



Detecting Lock Conflicts

Select Blocking Sessions on the Performance page.

Bloc	king Sessions										
(Vio	w Session)(Kill Session)						Page Refreshe	d Aug 18, 20	08 11:0	4:23 PM M	DT (Refresh
	nd All Collapse All										
200	t U sername	Sessions Blocked	Session	Serial Number		Wait Class	Wait Event		P2 Value	P3 Value	Seconds in
0	▼ Blocking Sessions										
æ	▼ BERNST	1	114	33091			SQL*Net message from client	1650815232	1	0	89
C	SMAVRIS	0	124	46897	Otqktcvhr5fcf	Application	enq: TX - row lock	1415053318	65545	3085	69

Click the Session ID link to view information about the locking session, including the actual SQL statement.

Resolving Lock Conflicts

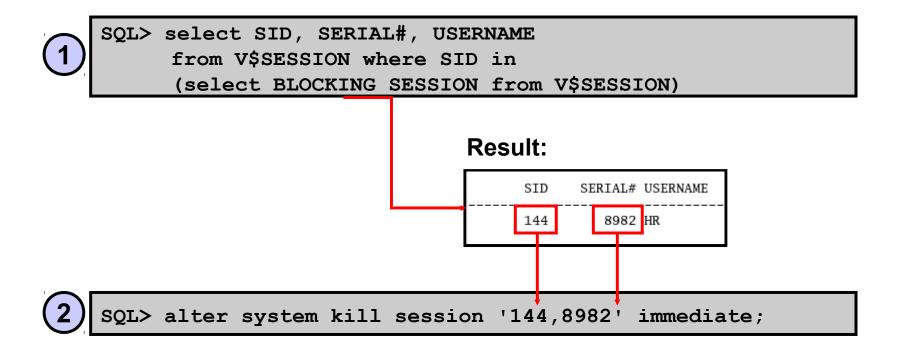
To resolve a lock conflict:

- Have the session holding the lock commit or roll back
- Terminate the session holding the lock (in an emergency)



Resolving Lock Conflicts with SQL

SQL statements can be used to determine the blocking session and kill it.



Deadlocks

Transaction 1		Transaction 2
<pre>UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 1000;</pre>	9:00	UPDATE employees SET manager = 1342 WHERE employee_id = 2000;
<pre>UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 2000;</pre>	9:15	<pre>UPDATE employees SET manager = 1342 WHERE employee_id = 1000;</pre>
ORA-00060: Deadlock detected while waiting for resource	9:16	

Quiz

The lock mechanism defaults to a fine-grained, row-level locking mode.

- 1. True
- 2. False

Quiz

When the deadlock occurs, Oracle database automatically:

- 1. Waits 300 seconds before terminating both sessions
- 2. Terminates one statement with an error in one session
- 3. Terminates the statements with an error in both sessions
- 4. Takes no action by default and leaves it to DBA

Summary

In this lesson, you should have learned how to:

- Describe the locking mechanism and how Oracle manages data concurrency
- Monitor and resolve locking conflicts

Practice 9 Overview: Managing Data and Concurrency

This practice covers the following topics:

- Identifying locking conflicts
- Resolving locking conflicts