



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

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=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: 09:00:00 | 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; |
| | 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.
```

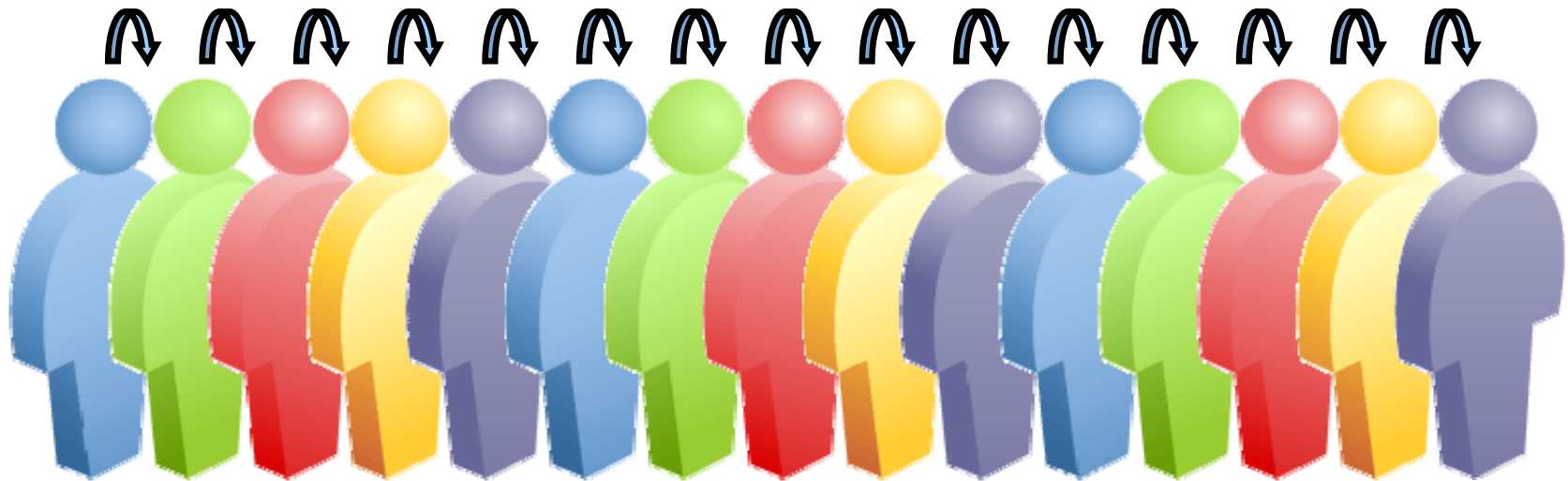
Each DML transaction must acquire *two* locks:

- 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 COMMISSION_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! |
| 1 row updated. Session continues. | 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.

| Blocking Sessions | | | | | | | | | | | |
|---|-------------------|------------------|---------------------|---------------|-------------------------------|-------------|---|------------|----------|----------|-----------------|
| Page Refreshed Aug 18, 2008 11:04:23 PM MDT Refresh | | | | | | | | | | | |
| View Session Kill Session | | | | | | | | | | | |
| Expand All Collapse All | | | | | | | | | | | |
| Select | Username | Sessions Blocked | Session ID | Serial Number | SQL ID | Wait Class | Wait Event | P1 Value | P2 Value | P3 Value | Seconds in Wait |
| | Blocking Sessions | | | | | | | | | | |
| | BERNST | 1 | 114 | 33091 | | Idle | SQL*Net message from client | 1650815232 | 1 | 0 | 89 |
| | SMAVRIS | 0 | 124 | 46897 | 0tqktcvhr5fcf | Application | eng: TX - row lock contention | 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.

1

```
SQL> select SID, SERIAL#, USERNAME  
       from V$SESSION where SID in  
       (select BLOCKING_SESSION from V$SESSION)
```


Result:

| SID | SERIAL# | USERNAME |
|-----|---------|----------|
| 144 | 8982 | HR |

2

```
SQL> alter system kill session '144,8982' immediate;
```

Deadlocks

| Transaction 1 | |  | Transaction 2 | |
|--|--|--|---|--|
| UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 1000; | | 9:00 | UPDATE employees SET manager = 1342 WHERE employee_id = 2000; | |
| UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 2000; | | 9:15 | UPDATE employees SET manager = 1342 WHERE employee_id = 1000; | |
| 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