# Overview of the Oracle Database Locking Mechanism

1.14.5

A **lock** is a mechanism that prevents destructive interactions.

Interactions are destructive when they incorrectly update data or incorrectly alter underlying data structures, between transactions accessing shared data. Locks play a crucial role in maintaining database concurrency and consistency.

## Summary of Locking Behavior

The database maintains several different types of locks, depending on the operation that acquired the lock.

In general, the database uses two types of locks: exclusive locks and share locks. Only one exclusive lock can be obtained on a resource such as a row or a table, but many share locks can be obtained on a single resource.

Locks affect the interaction of readers and writers. A reader is a query of a resource, whereas a writer is a statement modifying a resource. The following rules summarize the locking behavior of Oracle Database for readers and writers:

- A row is locked only when modified by a writer.

  When a statement updates one row, the transaction acquires a lock for this row only. By locking table data at the row level, the database minimizes contention for the same data. Under normal circumstances[1] the database does not escalate a row lock to the block or table level.

- A writer of a row blocks a concurrent writer of the same row.

  If one transaction is modifying a row, then a row lock prevents a different transaction from modifying the same row simultaneously.

- A reader never blocks a writer.

  Because a reader of a row does not lock it, a writer can modify this row. The only exception is a `SELECT ... FOR UPDATE` statement, which is a special type of `SELECT` statement that *does* lock the row that it is reading.

- A writer never blocks a reader.

  When a row is being changed by a writer, the database uses undo data to provide readers with a consistent view of the row.

> **Note:**
>
> Readers of data may have to wait for writers of the same data blocks in very special cases of pending distributed transactions.

[1] When processing a distributed two-phase commit, the database may briefly prevent read access in special circumstances. Specifically, if a query starts between the prepare and commit phases and attempts to read the data before the commit, then the database may escalate a lock from row-level to block-level to guarantee read consistency.

The database automatically acquires different types of locks at different levels of restrictiveness depending on the resource and the operation being performed.

> **Note:**
>
> The database never locks rows when performing simple reads.

Oracle Database locks are divided into the categories show in the following table.

**Table 9-6 Lock Categories**

| Lock | Description | To Learn More |
|---|---|---|
| DML Locks | Protect data. For example, table locks lock entire tables, while row locks lock selected rows. | "DML Locks" |
| DDL Locks | Protect the structure of schema objects—for example, the dictionary definitions of tables and views. | "DDL Locks" |
| System Locks | Protect internal database structures such as data files. Latches, mutexes, and internal locks are entirely automatic. | "System Locks" |

## DML Locks 1.14.5

A **DML lock**, also called a *data lock*, guarantees the integrity of data accessed concurrently by multiple users.

For example, a DML lock prevents two customers from buying the last copy of a book available from an online bookseller. DML locks prevent destructive interference of simultaneous conflicting DML or DDL operations.

DML statements automatically acquire the following types of locks:

- Row Locks (TX)
- Table Locks (TM)

In the following sections, the acronym in parentheses after each type of lock or lock mode is the abbreviation used in the Locks Monitor of Oracle Enterprise Manager (Enterprise Manager). Enterprise Manager might display TM for any table lock, rather than indicate the mode of table lock (such as RS or SRX).

> **See Also:**
>
> "Oracle Enterprise Manager"

### Row Locks (TX)

A **row lock**, also called a *TX lock*, is a lock on a single row of table. A transaction acquires a row lock for each row modified by an `INSERT`, `UPDATE`, `DELETE`, `MERGE`, or

`SELECT ... FOR UPDATE` statement. The row lock exists until the transaction commits or rolls back.

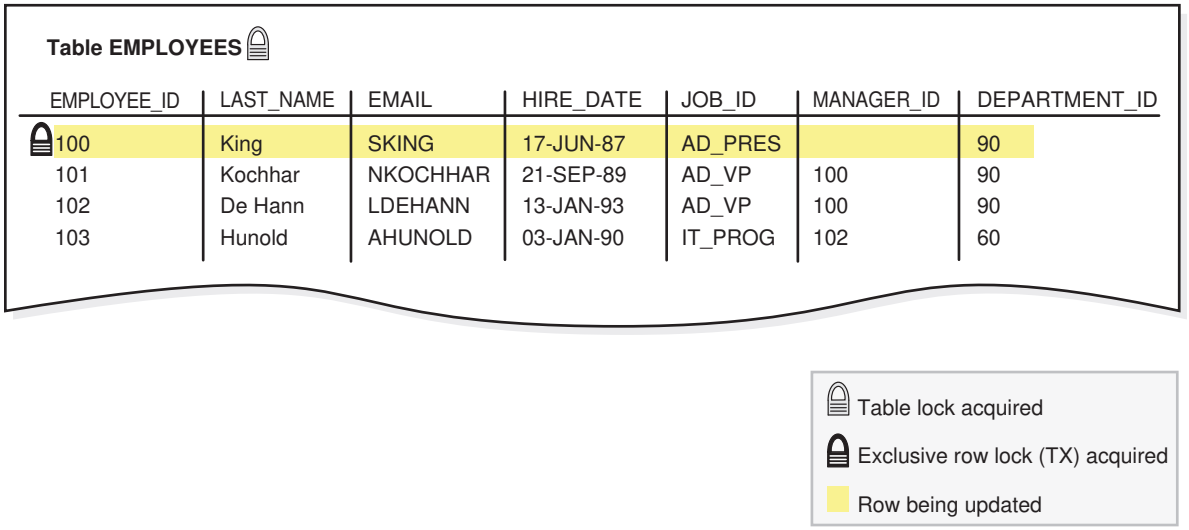
1.14.5

Row locks primarily serve as a queuing mechanism to prevent two transactions from modifying the same row. The database always locks a modified row in exclusive mode so that other transactions cannot modify the row until the transaction holding the lock commits or rolls back. Row locking provides the finest grain locking possible and so provides the best possible concurrency and throughput.

> **Note:**
>
> If a transaction terminates because of database instance failure, then block-level recovery makes a row available before the entire transaction is recovered.

If a transaction obtains a lock for a row, then the transaction also acquires a lock for the table containing the row. The table lock prevents conflicting DDL operations that would override data changes in a current transaction. Figure 9-2 illustrates an update of the third row in a table. Oracle Database automatically places an exclusive lock on the updated row and a subexclusive lock on the table.

**Figure 9-2 Row and Table Locks**




## Row Locks and Concurrency

This scenario illustrates how Oracle Database uses row locks for concurrency.

Three sessions query the same rows simultaneously. Session 1 and 2 proceed to make uncommitted updates to different rows, while session 3 makes no updates. Each session sees its own uncommitted updates but not the uncommitted updates of any other session.