Oracle DML Locks

Harveer Choudhary
Department of Computer Science Engineering
Research Scholars, Jagannath University, Jaipur, Rajasthan, India

Deepak Goyal
Department of Computer Science Engineering
Poornima Group of Institutions, Jaipur, Rajasthan, India

Abstract - The purpose of a DML lock, also called a data lock, is to guarantee the integrity of data being accessed concurrently by multiple users. Locks are mechanisms that prevent destructive interaction between transactions accessing the same resource. Locks can occur on any Oracle user objects such as tables and rows. Also System objects that are not visible to users, such as shared data structures in the memory and data dictionary rows. These locks will stay only till the duration of the session or the transaction. . DML locks prevent destructive interference of simultaneous conflicting DML or DDL operations.

Keyword- DML Lock, TM Lock, Table Lock, Row Lock, Non-blocking Queries.

I. INTRODUCTION

The purpose of a DML lock, also called a data lock, is to guarantee the integrity of data being accessed concurrently by multiple users. For example, a DML lock can prevent multiple 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 locks at both the table level and the row level. Row Locks (TX) Arrow lock, also called a TX lock, is a lock on a single row of a table. A transaction acquires a row lock for each row modified by one of the following statements: INSERT, UPDATE, DELETE, MERGE, and SELECTFORUPDATE. The row lock exists until the transaction commits or rolls back. A **DML lock** is a lock obtained on a table that is undergoing a DML operation (insert, update, delete). DML_LOCKS specifies the maximum number of DML locks—one for each table modified in a transaction. The value should equal the grand total of locks on tables currently referenced by all users. For example, if three users are modifying data in one table, then three entries would be required. If three users are modifying data in two tables, then six entries would be required.

II. TYPES OF LOCKS

2.1 Row Locks

These are the simplest part of the Oracle locking scheme to understand. Every row is either locked or it is unlocked. That's it (no lock types to consider). When you execute regular DML, row locks are obtained on the affected rows. The fact that a row is locked by a transaction is NOT stored in a centralized list of locks. When your transaction needs to lock a row, Oracle inspects if the transaction specified on the row is still active. If yes, the row is locked by that transaction and your transaction is blocked. If not, the row is locked by your transaction and your transaction id is now stored on the row.. Row locking provides the lowest level of locking possible provides the best possible transaction concurrency. Readers of data do not wait for writers of the same data rows.

2.2 Table Locks

When a DML Table lock is in place, another session cannot execute DDL on this table, thus preventing alteration of the table structure while you still hold a table lock. DML table locks are also referred to as TM locks. A transaction automatically acquires a table lock (TM lock) when a table is modified with the following statements: INSERT, UPDATE, DELETE, MERGE, and SELECT ... FORUPDATE. These DML operations require table locks to reserve DML access to the table on behalf of a transaction and to prevent DDL operations that would conflict with the transaction. Table locks can be taken in 5 different modes:

RS: row share
RX: row exclusive
S: share

SRX: share row exclusive
X: exclusive

A table lock can be held in any of the following modes:
A row share lock (RS), also called a sub share table lock (SS), indicates that the transaction holding the lock on the table has locked rows in the table and intends to update them. An SS lock is the least restrictive mode of table lock, offering the highest degree of concurrency for a table.
A row exclusive lock (RX), also called a sub exclusive table lock (SX), indicates that the transaction holding the lock has updated table rows or issued SELECT ... FORUPDATE. An SX lock allows other transactions to query, insert, update, delete, or lock rows concurrently in the same table. Therefore, SX locks allow multiple transactions to obtain simultaneous SX and SS locks for the same table.
A share table lock (S) held by one transaction allows other transactions to query the table (without using SELECT ... FORUPDATE) but allows updates only if a single transaction holds the share table lock. Multiple transactions may hold a share table locks concurrently, so holding this lock is not sufficient to ensure that a transaction can modify the table.
A share row exclusive table locks (SRX), also called a share-sub exclusive table lock (SSX), is more restrictive than a share table lock. Only one transaction at a time can acquire an SSX lock on a given table. An SSX lock held by a transaction allows other transactions to query the table (except for SELECT ... FORUPDATE) but not to update the table.
An exclusive table lock (X) is the most restrictive mode of table lock, allowing the transaction that holds the lock exclusive write access to the table. Only one transaction can obtain an X lock for a table.

III. ORACLE LOCKING MODE

3.1 Share Lock Mode

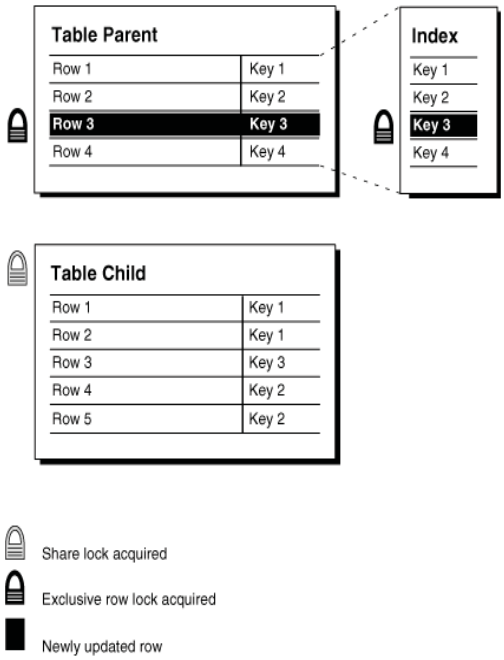
This mode allows the associated resource to be shared, depending on the operations involved. Multiple users reading data can share the data, holding share locks to prevent concurrent access by a writer (who needs an exclusive lock). Several transactions can acquire share locks on the same resource.

3.2 Exclusive Lock Mode

This mode prevents the associated resource from being shared. This lock mode is obtained to modify data. The first transaction to lock a resource exclusively is the only transaction that can alter the resource until the exclusive lock is released.

TM Lock Example
The Major contributor for the TM Locks or the Table Locks is the Missing Indexes on the Foreign Keys.

Locking with No Index on a Foreign Key
Non-indexed foreign keys cause DML on the primary key to get a share row exclusive table lock (also sometimes called a **share-sub exclusive table lock, SSX**) on the foreign key table. This prevents DML on the table by other transactions. The SSX lock is released immediately after it is obtained. If multiple primary keys are updated or deleted, the lock is obtained and released once for each row. The below diagram illustrates the locking mechanism on table with no index on the foreign key column.
Rows 1 through 4 of the parent table are indexed on keys 1 through 4, respectively. The child table is not foreign-key indexed to the parent table. Row 3 in the parent table is updated and acquires an exclusive row lock. At the same time, the child table acquires a share lock on the whole table.

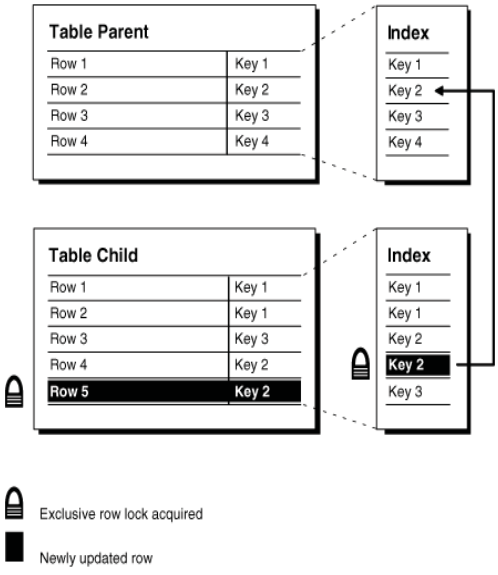


Locking with Index on a Foreign Key

Indexed foreign keys cause a row share table lock (also sometimes called a **sub share table lock, SS**). This prevents other transactions from exclusive locking the whole table, but it does not block DML on the parent or the child table. This situation is preferable if there is any update or delete activity on the parent table while update activity is taking place on the child table. Inserts, updates, and deletes on the parent table do not require any locks on the child table, although updates and deletes will wait for row-level locks on the indexes of the child table to clear.

The below diagram illustrates the locking mechanism on table with an index on the foreign key column.

Rows 1 through 4 of the parent table are indexed on keys 1 through 4, respectively. Row 5 in the child table is updated and acquires an exclusive row lock (key 2).



Locks When Rows Are Queried

A query can be explicit, as in the SELECT statement, or implicit, as in most INSERT, MERGE, UPDATE, and DELETE statements. The only DML statement that does not necessarily include a query component is an INSERT statement with a VALUES clause. Because queries only read data, they are the SQL statements least likely to interfere with other SQL statements.

The following characteristics apply to a query *without* the FOR UPDATE clause:

- The query acquires no data locks. Therefore, other transactions can query and update a table being queried, including the specific rows being queried. Because queries without the FOR UPDATE clause do not acquire any data locks to block other operations, such queries are often referred to as **non-blocking queries**.
- The query does not have to wait for any data locks to be released. Therefore, the query can always proceed. An exception to this rule is that queries may have to wait for data locks in some very specific cases of pending distributed transactions.

IV. EFFECT OF LOCKS

When the table gets locked out and there are other transactions that are waiting to acquire lock, turns out to a dead lock and all the waiting transactions will be indefinitely waiting on this table/resource. This would cause the DB job that is waiting for this table or the application that is waiting to Hang or wait until the deadlock is released.

4.1 Table Dead Lock Recovery

The Only way of recovering from the Table deadlock is to identify the transactions/sessions and kill them. If the table is used by any Applications and there are active sessions trying to lock this table, would keep the deadlocks growing on this particular table. So this table has to be taken out of access (Revoking the privileges to Application or users other than the owner) and clearing off the existing deadlocks and granting them the privilege after creating the required foreign key index.

4.2 Preventing the Deadlocks

The TM Locks or the Deadlocks can be prevented only by following set of design or coding standards and vigorous code/design reviews. During the design phase, if there is any new table designed with foreign key columns. The index creation on the foreign key column must be defined and the same has to be followed by the developers while developing the Table creation scripts. Also this can be caught during any of the lower environment testing or validation.

4.3 Identifying and fixing the cause for locks

One way of fixing the TM lock is to identify all the foreign key columns that are not having an appropriate index created on them and create the index. Also there is a possibility of having multiple update/delete queries in the same transaction. This can only be identified or fixed with the help of a code walk through. The below query will help in finding all the foreign key columns that are missing indexes. This query can be tuned to check on a specific schema level as well.

REFERENCES

[1] http://www.orafaq.com/wiki/Category:Frequently_Asked_Questions
[2] http://www.dba-oracle.com/m_tm.htm
[3] http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/data_int.htm#i8605
[4] http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/data_int.htm#i8589