David Corzo – 20190432 – Datos 1

Mecanismos de bloqueo y lectura consistente en MySQL, SQLServer, Casandra, DynamoDB.

|  |
| --- |
| MySQL:  "Un bloqueo es un mecanismo para que una computadora coordine múltiples procesos o subprocesos para acceder simultáneamente a un determinado recurso."  Los locks permiten lectura consistente uno causa al otro.   * Tipos de locks en MySQL: Shared and Exclusive Locks. Intention Locks. Record Locks. Gap Locks. Next-Key Locks. Insert Intention Locks. AUTO-INC Locks.Predicate Locks for Spatial Indexes. * Tipos de lectura consistente: Repeatable Read (database version of a git merge), Read commited (each consistent read within a transaction sets and reads its own fresh snapshot.),   Ejemplo:  **SELECT** CONNECTION\_ID();  **INSERT** **INTO** messages(message)**VALUES**('Hello');  **SELECT** \* **FROM** messages;  **LOCK** **TABLE** messages **READ**;  **INSERT** **INTO** messages(message)**VALUES**('Hi');  -- Error Code: 1099. Table 'messages' was locked with a READ lock and can't be updated.  Fuente: <https://programmerclick.com/article/2282898308/> <https://www.mysqltutorial.org/mysql-table-locking/> <https://dev.mysql.com/doc/refman/8.0/en/innodb-locking.html>  <https://dev.mysql.com/doc/refman/8.0/en/innodb-consistent-read.html> |
| SQLServer:  " Lock is a mechanism to ensure data consistency. SQL Server locks objects when the transaction starts. When the transaction is completed, SQL Server releases the locked object. This lock mode can be changed according to the SQL Server process type and isolation level."   * Tipos de locks en SQLServer: Shared (S) Lock, Exclusive (X) Lock, Update (U) Lock, Intent Locks. * Lectura consistente: read uncommited, read commited, repeatable read, serializable, read commited snapshot, snapshot.   Fuente: <https://codingsight.com/main-concept-of-sql-server-locking/#:~:text=Lock%3A%20Lock%20is%20a%20mechanism,Server%20releases%20the%20locked%20object.&text=Exclusive%20(X)%20Locks%3A%20When,or%20access%20a%20locked%20object>.  <https://docs.microsoft.com/en-us/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?view=sql-server-ver15> |
| Cassandra:  " Cassandra does not use RDBMS ACID transactions with rollback or locking mechanisms, but instead offers atomic, isolated, and durable transactions with eventual/tunable consistency that lets the user decide how strong or eventual they want each transaction’s consistency to be."  Cassandra no tiene locks como tal, pero tienen otros mecanismos que sirven como locks.   * Lectura consistente: En cassandra no hay locks, entonces la consistencia de lectura se divide en niveles de consistencia, entre estos está: ONE, TWO, THREE, QUORUM, ALL, LOCAL\_QUORUM, etcétera. Estos actuan como mecanismos de bloqueo aun que técnicamente no implementan suficiente lógica para ser llamados mecanismos de bloqueo.   Fuente: <https://teddyma.gitbooks.io/learncassandra/content/concurrent/concurrency_control.html> <https://blog.yugabyte.com/apache-cassandra-lightweight-transactions-secondary-indexes-tunable-consistency/> |
| DynamoDB:  "DynamoDB supports mechanisms, like conditional writes, that are necessary for distributed locks. However, the AWS SDK doesn’t include the logic needed to actually implement distributed locks. The DynamoDB Lock Client wraps up the necessary client logic for distributed advisory locks in an easy-to-use client interface. The protocol in the lock client Java library is widely applicable, and we encourage you to apply it in other languages."   * Lectura consistente: de nuevo no hay locks, entonces se utilizan niveles de lectura consistente, entre estos está: strong consistent read y eventually consistent read.   Fuente: <https://aws.amazon.com/blogs/database/building-distributed-locks-with-the-dynamodb-lock-client/> <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadConsistency.html> |

Práctica:

