Database Migration and Replication: On Consistency

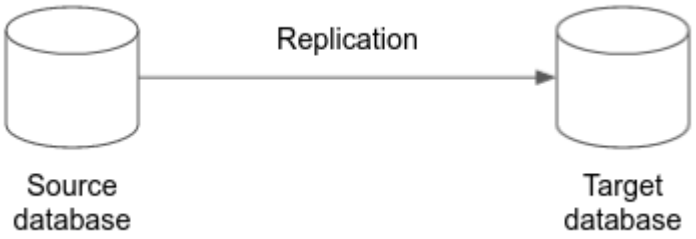
Mind the key replication axioms

Christoph Bussler
Jun 11 · 11 min read

tl;dr Data consistency is in many cases the key requirement for database migration or replication: if consistency cannot be guaranteed, the (contents of the) target database might be inconsistent compared to the (contents of the) source database. This blog discusses data migration and replication conceptually from the viewpoint of data consistency independent of any specific database migration or replication system or technology and independent of any non-functional aspects like throughput or latency.

Definition of database replication consistency

Database replication in its simplest form is continuously replicating all data from a source database to a target database:



Basic replication

For clarity of the discussion this blog focuses on a single source and target database, in one direction of replication. Additional situations, like reverse replication for fallback, data split, data combination, data filtering or n:m replications are not discussed; the basic principles apply to those cases as well.

The difference between database replication and database migration is discussed in a separate section “Database replication vs. database migration” below.

Database consistency

Database consistency can be defined based on transactions. The source database is consistent in its initial empty state, and then after each successfully committed transaction. The following diagram denotes source transactions with “st”, and the number is the order of the commit 1, 2, and 3. There might be additional transactions in the future, currently there are 3 committed transactions.

Source database transactions

The source transactions are linearized according to the transaction serialization and commit protocol implemented in the source database (ACID properties).

Even if during the source transaction execution the statements of different source transactions are interleaved, the source database establishes a linear commit order (and in some cases a commit tree to account for concurrent and independent committing transactions, which is abstracted from here for simplicity as it does not change the discussion in the following). Given any two transactions, their order of execution in the source database is known once they committed even though their execution might be concurrent and interleaved.

The linear order of the source transactions in the source database is important as each committed transaction establishes a consistent state. This order of source transactions will later be replicated in the target database to establish the same consistent states.

Database replication consistency

Database replication consistency can be defined on the equivalence of the linear source database transaction order. The target database is consistent with the source database if

1. The target database is empty
2. The target database has some, but not all equivalent source transactions executed each exactly once in the same order starting from the first source transaction
3. The target database has all equivalent transactions of the source database each exactly once executed in the same order starting from the first source transaction

“Exactly once” asserts that no source transaction is left out, or executed twice (or more times) in the target database.

The following diagram shows case 2. Each source transaction is represented as a target database transaction, labelled “tt” with numbers establishing the correspondence and order as above.

Application of source database transactions as independent transactions to the target database

The target database has the first two source transactions executed as two target transactions. The third source transaction is not yet executed in the target database. The target database is consistent with the source at one of its previous points in time, meaning, it lags behind — in this example by one transaction — but has a past consistent state.

Database replication is consistent if at any point the source database and the target database are consistent based on the source transaction execution order.

Note: above consistency definition allows for a delay (step 2). Such a delay is possible if the source transactions can be committed, and those are later on executed as target database transactions on the target database. “Later on” is basically referring to an

asynchronous replication protocol that replicates the source transactions after they have been committed. A synchronous replication protocol would ensure that the target transaction commits together with the corresponding source transactions. In a synchronous replication protocol step 2 from above does not apply.

In the following the discussion focuses on asynchronous replication as this is the predominant approach when database replication technology is involved that is not implemented as native database technology.

Database replication technology implementing asynchronous replication sometimes applies individual statements to the target database, not transactions. This approach is discussed below in section “System architecture aspects”.

Transaction replication and execution

There are two aspects of transactions that I did not discuss so far, but are important as well:

1. **Statements within transactions**. Transactions are a series of statements like query, insert, update and delete. The order of their execution in the source database must be preserved in the target database as well since several statements can modify the same data. A different statement order within a target transaction compared to its corresponding source transactions could result in different data values, and hence violate consistency.
2. **Queries**. Only the modifying statements in a transaction are relevant: insert, update, and delete (and their order). Queries (read) can be left out. This means that if source transactions contain query statements, those do not have to be executed in the target database. If the source transactions are obtained from a source database transaction log then those might not contain the queries anymore.

Each source database transaction must be replicated as a corresponding target database transaction:

- Same order of in-transaction statements
- Optionally with the queries left out (if not already removed in the source transactions)

Each replicated target database transaction must be executed exactly once in the same order as its corresponding source transaction.

Key replication axioms

In order to ensure that the target database is always consistent with the source database the following replication axioms have to hold at any point during the replication:

- **Completeness**. The complete set of source database transactions starting at the first transaction must be executed as corresponding target transactions in the target database. Complete also means no transactions are left out, and that both the source and target database start from an empty state.
- **Exactly once**. Each source transaction is executed exactly once as a corresponding target transaction in the target database. Exactly once means that there are no duplicate transaction executions.
- **Ordered**. The sequence of target transactions is executed in the exact same order as their corresponding source transactions. This means that there is no change of the target transaction commit order in relation to the corresponding source transaction commit order.
- **Isolated**. The target database must be isolated from any other possible changes except those corresponding to the source transactions. If the target database has schemas or sets of tables that are completely independent of those tables that are replicas of the source database then their concurrent modifications are possible. Those independent schemas and tables are isolated by definition.

In practical terms this means that in order to have a target database that is always consistent (in the above definition) with the source database any migration or replication technology has to comply with the processing axioms.

Error semantics and behavior

Database replication is in general implemented by an independent replication system that can obtain the source database transactions from the source database and execute the equivalent target database transactions in the target database. A replication system therefore interfaces with both databases, and has to comply with the replication axioms just outlined.

The same applies to replication technology that is built-in to the database engine technology itself. In the following this is subsumed by the notion of replication technology.

A database replication system is a third independent system and there are three principle failure situations:

- Access of the source database system fails (might be API invocation error, or also a source database file system corruption, among other reasons)
- Access of the target database system fails
- The replication system itself fails

If a failure occurs, the replication system has two options

- Recover from the failure and continue processing, ensuring that none of the replication axioms are violated. This is only possible if the replication system keeps precisely track of the replication progress to ensure that the replication axioms are upheld in all possible failure circumstances.
- Restart from the beginning, with an empty target database and execute target transactions corresponding to the source transactions starting at the first source transaction. This is a safe option if the replication system cannot ensure that the replication axioms are upheld and not violated.

For example, it is possible that a target database transaction is aborted. In this case the replication system must retry this transaction until it succeeds before executing the next target transaction in the appropriate order. It is possible that the target transaction never succeeds. To prevent endless retry attempts, an upper limit is usually set and if met the replication process stops completely. No further target transactions are executed until the error situation is resolved.

Another example is that the replication system itself might fail and restart. In this case, after restart, the migration system has to establish its last consistent state and that of the overall replication. It has to establish which source transactions it has already fetched, which target transactions it executed and continue processing without violating the replication axioms. If the replication system cannot establish a consistent state, it has to

restart from the very beginning from the first source transaction and with an empty target database.

Database replication vs. database migration

Database replication and migration differentiate each other by intent. The intent of database replication is an ongoing, in principle never ending process — "forever". One started, database replication replicates a source database to a target database without a predefined end of the replication process.

Database migration, in contrast, has a different intent. Its intent is to have a temporary process that replicates a source database to a target database with the goal to eventually turn down the source database. An additional aspect is that before the turndown of the source database the target database contains the exact same state as the source database so that no data is lost.

In principle, database migration is the controlled conclusion of database replication. Based on this differentiation, database migration can be defined in terms of database replication as follows:

- Replicate the source database to the target database continuously following the replication axioms
- Stop any changes to the source database
- Wait until all source transactions are applied as target transactions in the target database following the replication axioms
- Turn down the source database

System architecture aspects

There are many database replication and migration systems implemented, products as well as home-grown mechanisms. In the following some select aspects are briefly discussed.

Caveats

In the following I discuss a few caveats that you might want to be aware of if consistency is your main concern:

- **Statement replication**. Some replication systems do not execute target transactions in the target database that correspond to source transactions. Instead, they insert a target transaction for each source statement (!) of a source transaction. So if a source transaction has 5 DML statements, that results in 5 transactions in the target database. This leads to continuous inconsistency since the target database is not consistent unless and only when the last of the statements of a transaction is committed.

- **No ordering**. Some systems do not guarantee that the order of the target transactions corresponds to the order of the equivalent source transitions. Since a replication axiom is violated the target database cannot be guaranteed to be consistent.

- **Not exactly once or at least once**. Some systems state that duplicate target transactions are possible. Since a replication axiom is violated, the target database might reject a transaction or it might lead to inconsistent data.

- **Completeness not guaranteed**. Some systems do not guarantee completeness and some source transactions might not get executed in the target database in the form of an equivalent target database transaction. Since this violates the replication axioms an inconsistent target database is possible.

- **Error does not stop the system**. Some systems do not stop when they encounter an error but write a log entry and continue with the next target transaction. Since completeness is violated, the target database might become inconsistent.

- **Dead letter queue**. Some systems implement the replication functionality based on queuing systems. It is possible that a dead letter queue is present that contains all failed transactions. The system puts a transaction into the dead letter queue and proceeds with the next transaction. Since this violates completeness, the target database might be inconsistent.

For some of the above aspects possible workarounds might be possible as follows.

- **Statement replication => statement collection**. If it is important that the target database is always consistent in the above sense, then the replication system could

collect all statements of a transaction in the correct statement order and execute the whole transaction instead of individual statements in the target database.

- **No ordering => ordering**. Ordering can be established within the replication system by preserving the order the source database established. If that is not possible, the transaction order could be recreated from metadata if it is available.
- **Not exactly once => duplicate detection**. A replication system in general has state and its state could establish an exactly once execution of target database transactions by keeping track of the execution progress.
- **Completeness not guaranteed => completeness verification**. The source database is assumed to be complete and provides a complete sequence of source database transactions. This completeness can be preserved by the replication system by keeping track of target database transaction execution in its internal state.
- **Error does not stop the system => immediate halt on error**. A non-recoverable error can immediately stop the execution of target transactions so that the replication axioms are complied with.
- **Dead letter queue => remove**. A dead letter queue cannot have an entry while target transactions are being executed. As soon as a transaction fails, the system has to stop for error resolution.

Initial load

Above consistency is defined on source transactions starting from an empty source database. In real database migration deployments the need for replication might come long after source transactions have been executed. In these cases the concept of an initial load applies. When starting the replication, the whole source database is read at a snapshot (consistent), and after it being applied to the target database, the source transactions after the snapshot are executed as target database transactions in the target database. A source snapshot is the combination of all transactions up to this point in time and a snapshot is consistent. This approach results in a consistent target database when the replication axioms are implemented after the snapshot.

Optimization

It is possible that a system implements optimizations, for example, executing target transactions concurrently if they are fully independent of each other. However, first, an optimization must not violate the replication axioms. Second, while the target database is consistent from the viewpoint of its data set, other aspects are different, mainly the transaction execution order. If any client relies on a specific transaction order in the target database (by e.g. reading log files) then this might lead to inconsistencies on the client side.

Restart from the beginning

In a non-recoverable error situation it might be necessary to restart the migration process with an empty target database. While this is perfectly possible and will result in a consistent target database if the replication axioms are complied with, other problems might arise if clients depend on investigating or reacting to individual target database transactions. A restart will in general not replay each source transaction individually as those might not be available and instead an initial dump is executed.

Summary

This blog discussed database replication and migration solely from the viewpoint of data consistency. It established a definition and inferred from those replication axioms that replication technology has to comply with. It looked at selected system architecture aspects from the viewpoint of data consistency as well.

Acknowledgements

I'd like to thank Pritesh Jani and John Darrah for the thorough review and several comments to improve the accuracy of this content.

Disclaimer

Christoph Bussler is a Solutions Architect at Google, Inc. (Google Cloud). The opinions stated here are my own, not those of Google, Inc.

Google Cloud Database Replication Database Migration Consistency