IBExpert White Paper

Firebird Database Replication: Reliable, Near-Real-Time Synchronization for Mission-Critical Systems

Holger Klemt, August 2026

Modern applications increasingly require databases to remain available across multiple locations, servers, and environments. Whether the goal is high availability, migration from a desktop application to a web-based system, distributed databases, or continuous synchronization between data centers, reliable database replication is a critical component.

Our Firebird replication solution has been developed for precisely these scenarios. It combines transaction-safe replication, near-real-time synchronization, one-to-many replication, and support for large Firebird databases — while allowing replication to be introduced into an existing production environment without taking the database offline.

Designed for Firebird

The replication framework supports Firebird 2.5 and newer versions, including Firebird 4, Firebird 5 and current Firebird 6 databases.

A key advantage is that replication is implemented directly at the database level using active triggers and a transaction log. Changes made to the source database are recorded as transactions and can subsequently be transferred to one or more target databases.

This architecture ensures that changes are not lost simply because the network connection or replication process is temporarily unavailable.

Replication without production downtime

One of the most important requirements in a migration or high-availability project is the ability to activate replication without interrupting the running application.

Our replication process is designed for exactly this purpose.

Once replication is initialized, the required triggers begin recording INSERT, UPDATE and DELETE operations in a dedicated transaction log. The production database continues to operate normally.

The initial synchronization can then be performed by creating a backup of the master database and restoring it on the target server. This process can also be performed without taking the production database offline.

After the target database has been initialized, the replication process transfers all transactions that occurred while the backup was being created and subsequently continues with new transactions.

Transaction-safe replication

Reliability is at the heart of the architecture.

Imagine that the master database has accumulated 100 transactions waiting to be replicated. If the network connection fails after only part of the data has been transferred, the remaining transactions are not lost.

They remain in the transaction log and are transferred during the next replication operation.

The replication process therefore does not depend on a permanently available network connection. A target server can even be unavailable for days or weeks while the master database continues to collect the transactions required for synchronization.

When the target becomes available again, replication can continue from where it stopped.

One-to-many replication

The architecture also supports one-to-many replication.

A single master database can replicate its data to multiple target servers. Each target is tracked independently.

For example:

Master Server → Slave 1
Master Server → Slave 2
Master Server → Slave 3

If Slave 2 is temporarily offline, replication to Slave 1 and Slave 3 can continue normally. The transactions required for Slave 2 remain available on the master until that target reconnects.

This makes the architecture suitable for distributed systems where individual locations may have unreliable network connections or temporarily unavailable servers.

Multi-master replication

Depending on the database structure, the technology can also support multi-master replication.

In this configuration, several servers can act as master databases and exchange changes between one another.

However, multi-master environments require careful consideration of the database design, particularly primary-key generation. For example, automatically generated keys must be designed so that records created independently on different master servers cannot produce conflicts.

Where necessary, database metadata and application logic can be adapted to support this architecture.

Near-real-time synchronization

The replication process is designed for near-real-time operation.

In a typical environment, transactions are transferred continuously through a push/pull process. The actual latency depends on server performance, transaction volume and network conditions.

In one large distributed deployment, changes were typically visible at remote locations in less than 10 seconds. Even under periods of very high system load, latency generally remained within approximately one minute.

Locations with slower or unreliable Internet connections could continue operating locally. Once connectivity was restored, accumulated transactions were synchronized automatically.

Proven with large and distributed Firebird installations

The technology has been used in demanding real-world environments involving large databases and geographically distributed systems.

One particularly extensive project involved:

- 9 central servers
- 6 different data centers and cities
- Approximately 225 Firebird databases at individual locations
- Approximately 500 GB in the largest central database
- Around 2 billion records in the largest database
- Approximately 5–6 million INSERT, UPDATE and DELETE operations per day
- Near-real-time synchronization between the central systems and remote locations

The individual remote databases contained subsets of the central data and synchronized local changes back to the central infrastructure.

Despite unreliable network connections at some locations, the replication architecture ensured that transactions remained available until they could be transferred successfully.

Handling very large databases and BLOB data

Large Firebird databases can contain substantial amounts of BLOB data, particularly in applications such as healthcare, document management, imaging and archive systems.

In one implementation, the overall database environment reached approximately 7 TB, including transaction logs and annual BLOB databases.

For very large datasets, separating historical data into dedicated databases can provide significant advantages.

For example, annual transaction or BLOB databases can be closed and made read-only at the end of a financial year. They remain accessible via the main database, but no longer increase the size of the active production database.

This approach can significantly simplify and accelerate future backup and restore operations.

No single point of failure in the replication process

A temporary failure of a target server or network connection does not stop the master database from recording changes.

This is particularly important for distributed installations.

A remote server may be restarted, disconnected, or unavailable for an extended period. Once it becomes available again, the replication process can continue using the transactions that accumulated while it was offline.

The result is a replication architecture designed around reliable, post-event synchronization without data loss, which does not require all servers and network connections to be permanently available.

Database metadata plays a crucial role

Reliable replication starts with a suitable database design.

Ideally, every table that needs to be replicated should have a primary key or at least a suitable unique index.

Primary keys should also be reasonably compact. Extremely complex primary keys spanning a large number of columns can make replication and application development unnecessarily complicated.

Tables without primary keys may require an alternative identification mechanism. Depending on the database structure, a different technology based on Firebird's RDB$DB_KEY can be considered.

For complex environments, the database metadata can be analyzed before implementation to determine the most appropriate replication strategy.

Continuous replication with minimal maintenance

Once implemented, the replication system is designed to operate largely in the background.

In many installations, no regular manual maintenance is required.

Nevertheless, database administrators may occasionally perform extensive metadata changes or structural modifications that require replication to be reinitialized or adjusted.

For such situations, remote technical assistance can be provided through a prepaid support and hotline system. This allows administrators to request assistance when required rather than maintaining a permanent support contract solely for occasional replication operations.

Implementation and migration

Every replication project is different. Database size alone is not sufficient to determine the implementation effort.

Before a project begins, we therefore recommend analyzing:

- The database metadata
- Primary keys and indexes
- Database statistics
- Number and size of databases
- Transaction volume
- BLOB usage
- Network topology
- Number of replication targets
- Required replication direction
- Expected availability and recovery requirements

A metadata-only backup is often sufficient for the initial technical assessment. This allows the database structure to be reviewed without providing access to sensitive customer data.

Implementation and training can be carried out entirely remotely – even for customers in distant time zones.

Source code and customization

The replication framework can be delivered together with its source code, allowing customers to understand, adapt and integrate the solution into their own environment.

This is particularly valuable when replication forms part of a larger migration project, such as moving an existing desktop application to a web-based architecture while keeping the existing production system operational during the transition.

A practical foundation for system migration

Database replication is especially valuable when an organization needs to modernize an existing application without accepting prolonged downtime.

For example, a healthcare information system (HIS) and its associated image database may need to remain continuously available while the underlying application is migrated from a desktop architecture to a modern web application.

A transaction-based replication layer can provide a bridge between the existing and new environments, allowing both systems to operate while data is continuously synchronized.

This can substantially reduce migration risks and enable a controlled transition rather than a disruptive "big bang" migration.

A solution built on real-world experience

The architecture described here is not limited to laboratory scenarios. It has been used for more than a decade in demanding production environments with geographically distributed Firebird databases, large transaction volumes, unreliable network connections and substantial amounts of data.

The combination of transaction logging, trigger-based change tracking, push/pull replication and independent target tracking provides a robust foundation for high-availability and distributed Firebird environments.

For organizations planning a Firebird migration, high-availability project, distributed database architecture or near-real-time synchronization solution, the first step is a technical review of the existing database environment.

With the database metadata and current database statistics available, a realistic implementation plan, replication architecture and project-specific quotation can be prepared.