Firebird : Open Source Relational Database

Replication in Firebird 4: concepts and usage

Dmitry Yemanov
dimitr@firebirdsql.org

Firebird Project
www.firebirdsql.org

Firebird Conference 2019
Berlin, 17-19 October

IBPhoenix
YOUR PREMIER SOURCE OF FIREBIRD SUPPORT

IBSurgeon

MOSCOW EXCHANGE

Fast Reports
Reporting must be fast!

IB Expert

REDSOFT

Concepts

Initial goals

- Built-in replacement for (3rd party) trigger-based solutions
- Logical (aka record level) replication (*)
- No need for journal table(s) and triggers (intact schema)
- Better performance (small overhead, no GC problems)
- Native support for sequences and DDL operations

Concepts

Initial goals

- Built-in replacement for (3rd party) trigger-based solutions
- Logical (aka record level) replication (*)
- No need for journal table(s) and triggers (intact schema)
- Better performance (small overhead, no GC problems)
- Native support for sequences and DDL operations

(*) Statement-level for sequence and DDL operations

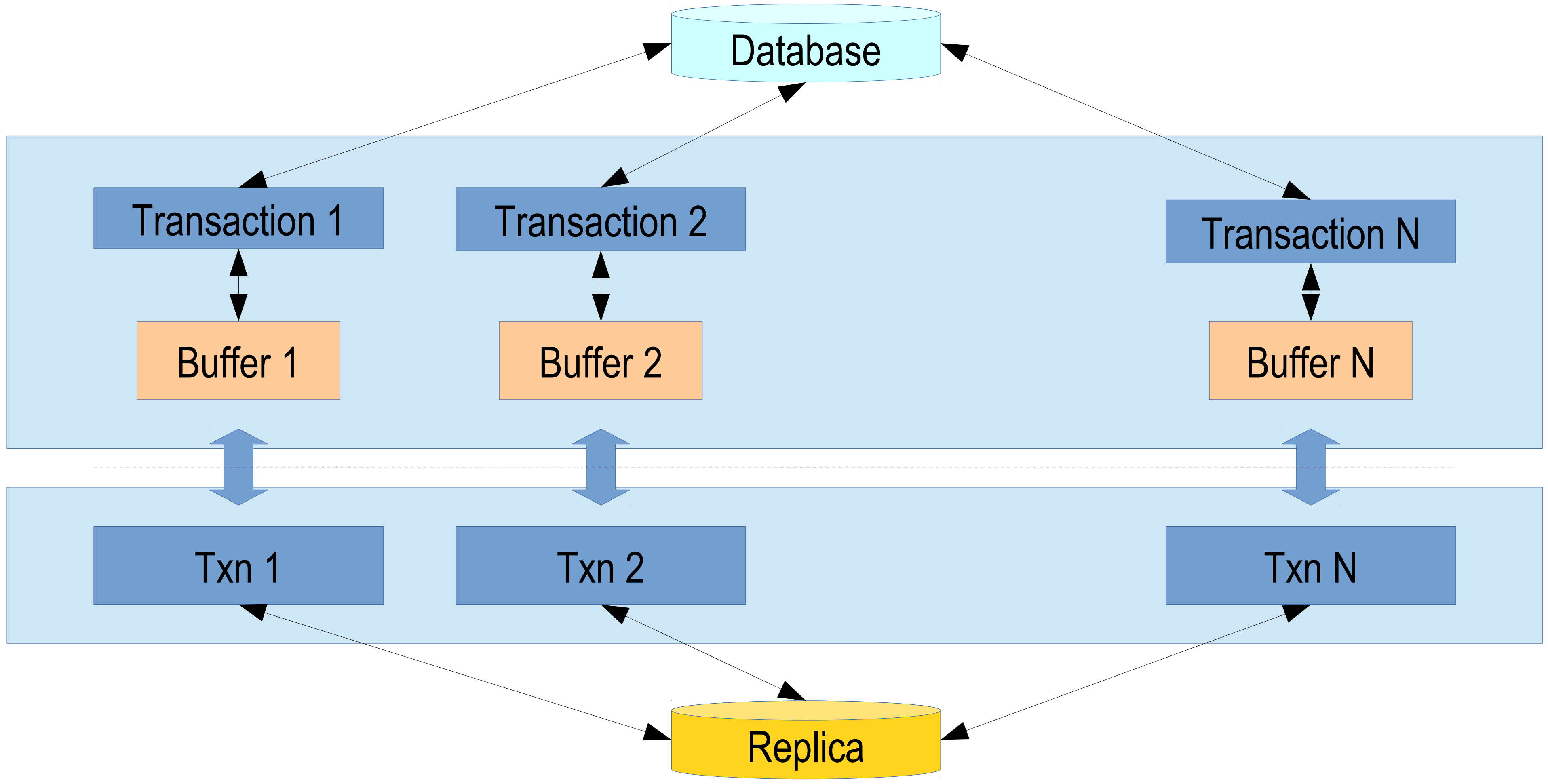
Concepts

Key features

- Logical uni-directional replication
- «PUSH» approach, different transport options
- Synchronous and asynchronous
- Simple configuration
- Customizable replication set
- Conflict detection, reporting and correction
- Load balancing (read-only)

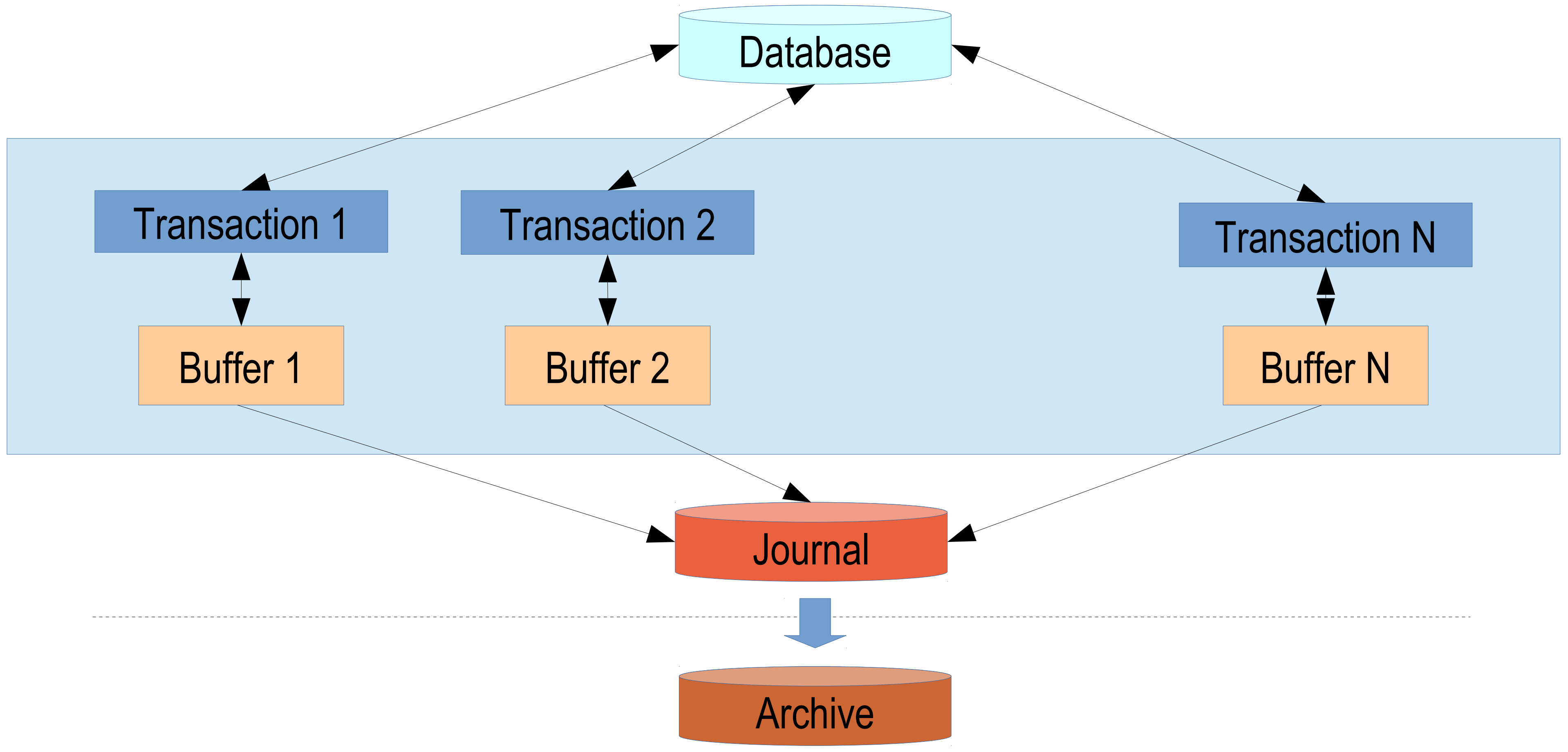
Concepts

Synchronous replication



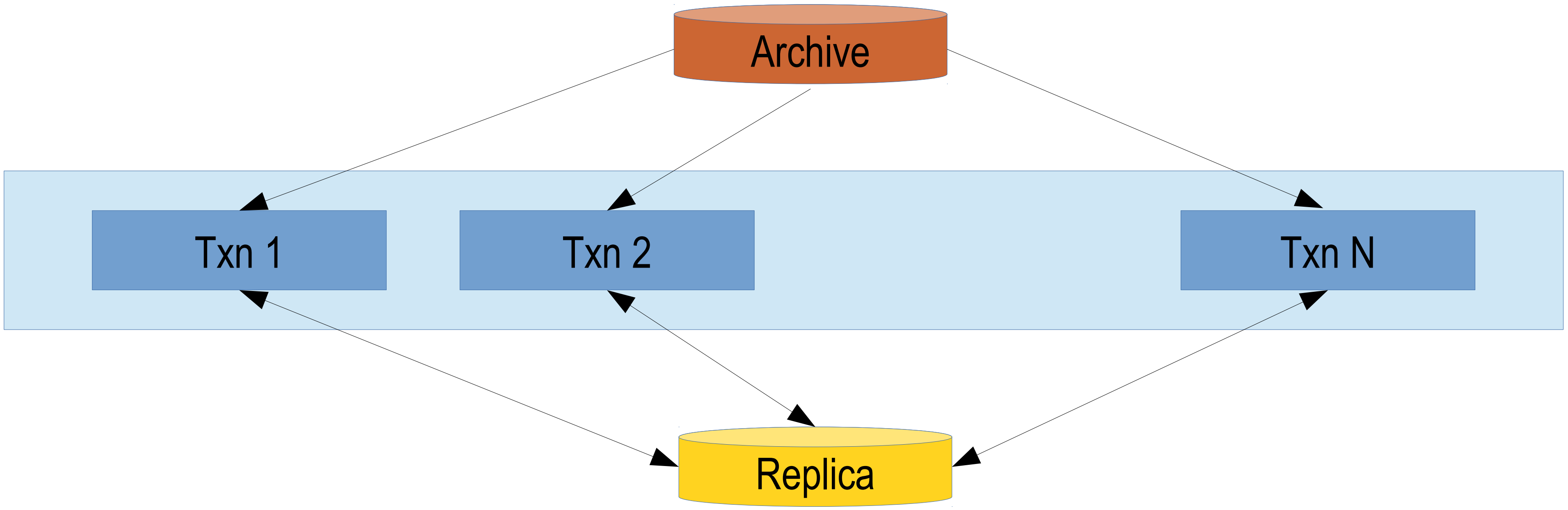
Concepts

Asynchronous replication (primary side)



Concepts

Asynchronous replication (replica side)



Architecture

Under the hood

- Replication is transaction-aware
- Every transaction has internal replication buffer
  (size is configurable, should be balanced)
- Buffer is flushed upon either:
  - Size overflow
  - Transaction commit / rollback
  - Savepoint rollback
- Every «flush» produces a replication packet (aka «change block»)
- Blocks are transferred to replica database(s)
  or written to the journal

Architecture

Under the hood

- Buffers are not always flushed synchronously
- Replication background thread per database
- Queue of «overflow» blocks to be flushed
- Lagging is limited to keep the throughput stable

Architecture

Specifics

- Both «changes» and «undos» are replicated
- Savepoint stack is preserved
- «Undos» are frame-based (using savepoints)

Optimization

- Small rolled back transactions are not flushed, just discarded

Architecture

Never replicated

- «De facto» read-only transactions
- External tables
- Virtual tables
- Temporary tables
- Any garbage collection activity
- System sequences, except *RDB$BACKUP_HISTORY*
- Some DDL commands:

```
ALTER DATABASE, DROP DATABASE
CREATE SHADOW, DROP SHADOW
CREATE USER, ALTER USER, DROP USER
```

Architecture

Error handling

- Every error is written to replication.log,
  prefixed with (primary | replica) side and database pathname
- For synchronous replication, errors may be duplicated
  on the both sides
- If error is persistent and affects user operations,
  replication is automatically disabled (at least partially)
- replication.log may also contain warnings,
  they do not affect the replication flow

Architecture

Synchronous replication

- Every primary database keeps active connections to all the synchronous replica databases
- Replication packets are transferred via native remote protocol and Firebird API
- Failed synchronous replica is excluded from replication, others remain working

Architecture

Asynchronous replication

- Journal — contiguous sequence of segments
- Linked to its corresponding database via UUID
- Segments are uniquely (and sequentially) numbered
- Change blocks are written one after another, every block has an associated flush timestamp
- Operational and archive journals

Architecture

Operational journal

- One or more segments on the primary side that are being written to
- Segments are rotated (with renaming)
- Segments may have multiple states:
  - FREE — empty segment ready for reuse
  - USED — segment being currently written
  - FULL — segment ready for archiving
  - ARCHIVE — segment being archived
- Archieving is a process of copying full segments elsewhere (to apply them to the replica database later)
- Archive segments are persistent, read-only and not rotatable

Architecture

How segments are applied to the replica

- Firebird process creates an embedded connection
- Journal directory is periodically scanned for new files
- Found segments are read and processed
  one after another in the sequence order
- Segments are removed automatically after applying
- Replica may be disconnected and reconnected after timeout

Architecture

How segments are applied to the replica

- Replica may be disconnected and reconnected after timeout
- Segments containing changes from not yet committed transactions are preseved until those transactions are finished
- Markers: Oldest Sequence and Next Sequence : Offset
- Current state is stored in the {UUID} file
- After reconnection changes from «unfinished» transactions are re-applied, other changes up no Next:Offset are skipped
- Then replication continues in the usual mode

Architecture

Load balancing on the replica side

- Replica may be read-only or read-write
- In read-only replica, all transactions started by regular users are forced to be read-only
- Thanks to MGA, readers do not conflict with writers — concurrent reads by users (e.g. reporting) are possible
- But conflicts are still possible (DDL changes)
- Read-write replica allows concurrent writes by users
- Conflicts must be avoided by users

Configuration

Concepts

- replication.conf —
  all settings for both primary and replica sides
- Primary side: parsed and cached when the first connection attaches to the database
- Replica side: parsed and cached when Firebird is started

Core settings for the primary side

- include_filter, exclude_filter —
  regular expressions for table customization
- buffer_size — per transaction caching threshold

Configuration

Synchronous mode

- sync_replica — connection string to the replica database
- Multiple entries are allowed:

```
sync_replica = john:smith@backup1:/my/replica/db1.fdb
sync_replica = john:smith@backup2:/my/replica/db2.fdb
```

Example

```
database = /your/db.fdb
{
  sync_replica = sysdba:masterkey@otherhost:/db.fdb
}
```

Configuration

Asynchronous mode (master side)

- Many options — read replication.conf for details
  log_directory, log_file_prefix, log_segment_size, log_segment_count, log_group_flush_delay, log_archive_directory, log_archive_command, log_archive_timeout
- *log_directory* is required
- Either *log_archive_directory* or *log_archive_command* is required
- Other options are used for tuning

Example

```
database = /your/db.fdb
{
    log_directory = /your/db/operlog/
    log_archive_directory = /your/db/archlog/
}
```

Configuration

Asynchronous mode (replica side)

- Only *log_source_directory* is required
- Other options are used for tuning

Example

```
database = /your/db.fdb
{
    log_source_directory = /your/db/incominglog/
}
```

Usage patterns

How to start synchronous replication

1) Set up replication.conf for your database
2) Restart Firebird or reconnect all users
3) Check replication.log

Usage patterns

How to start asynchronous replication

1) Create directory for operational and archive journals (better on a different storage)
2) Set up replication.conf for your database
3) Restart Firebird or reconnect all users
4) Check replication.log
5) Ensure journal files are being created and archived properly

Usage patterns

How to set up replica

1) Make a file-level copy of the primary database
2) Gfix <database> -replica read_only

If asynchronous replication is used

3) Set up replication.conf for the replica database
4) Restart Firebird service
5) Ensure journal files are received and processed
6) Try *verbose_logging = true* for better understanding

Usage patterns

How to fix broken replication

1) Make a file-level copy of the primary database
2) Gfix <database> -replica read_only
3) Shutdown the broken replica
4) Copy the broken replica elsewhere (or remove it)
5) Rename the new replica to the old name
6) Do not touch anything else ;-)

Usage patterns

How to recover from failure

1) For asynchronous replication decide whether to recover right now (but lose some recent changes) or wait for replication to catch up
2) Stop replication on the replica side:
   - Shutdown replica database
   - Shutdown Firebird service
3) Disable replica side settings in replication.conf
4) Gfix <database> -replica none
5) Copy replica to become the new primary

Questions?