Firebird White Paper

Firebird 4 Replikation

Fikret Hasovic, Juli 2021

Was ist eine Datenbankreplikation?

Datenbankreplikation bezeichnet das häufige Kopieren von Daten aus einer Datenbank von einem Server zu einer Datenbank auf einem anderen Server, so dass alle Benutzer den gleichen Informationsstand haben. Das Ergebnis ist eine verteilte Datenbank, in der die Nutzer schnell auf die für ihre Aufgaben relevanten Daten zugreifen können.

Eine kurze Einführung

Firebird 4 führt eine eingebaute Unterstützung für unidirektionale ("primary-replica") logische Replikation ein. Logisch bedeutet hier Replikation auf Datensatzebene, im Gegensatz zur physischen Replikation (auf Seitenebene).

Zu den Ereignissen, die bei der Replikation verfolgt werden, gehören eingefügte/geänderte/gelöschte Datensätze, Sequenzänderungen und DDL-Anweisungen.

Die Replikation ist transaktional und die Commit-Reihenfolge wird beibehalten. Jede Tabelle, die repliziert werden soll, muss einen Primärschlüssel oder zumindest einen eindeutigen Schlüssel haben.

Es sind sowohl *synchrone* als auch *asynchrone* Modi verfügbar.

Synchroner Modus

Bei der synchronen Replikation ist die primäre Datenbank (Master) ständig mit der/den replizierten Datenbank(en) (Slave) verbunden, Änderungen werden sofort repliziert. Die Datenbanken werden nach jeder Übertragung synchronisiert, was sich aufgrund des zusätzlichen Netzwerkverkehrs und der Round-trips auf die Leistung auswirken kann.

Sie können bei Bedarf mehr als eine synchrone Replikation haben.

Asynchroner Modus

Bei der asynchronen Replikation werden Änderungen in lokale Journaldateien geschrieben, die über die Leitung übertragen und auf die Replikat-Datenbank angewendet werden. Die Auswirkung auf die Leistung ist viel geringer, aber es kommt zu einer Verzögerung - dem so genannten *Replication Lag* - während die Änderungen darauf warten, auf die Replikat-Datenbank angewendet zu werden; d.h. die Replikat-Datenbank "holt" die Master-Datenbank immer ein.

Zugriffsmodi

Es gibt zwei Zugriffsmodi für Replikat-Datenbanken: *read_only* und *read_write*.

Bei einem read_only-Replikat sind nur Abfragen erlaubt, die die Daten nicht verändern. Änderungen sind nur auf den Replikationsprozess beschränkt.

Eine read_write Replik erlaubt die Ausführung jeder Abfrage, so dass potenzielle Konflikte bestehen können, die von Benutzern oder Datenbankadministratoren gelöst werden müssen.

Eine ausführliche Erklärung und vollständige Dokumentation finden Sie in den Release Notes.

Lassen Sie uns jetzt ein paar Übungen machen...

Einrichten der Replikation

Die Einrichtung umfasst Aufgaben sowohl auf der primären als auch auf der replizierenden Seite.

Einrichten der Primärseite

Die Replikation wird mit einer einzigen Konfigurationsdatei, `replication.conf`, auf dem Host konfiguriert, der die primäre Datenbank bedient. In derselben Datei sind sowohl globale als auch datenbank-bezogene Einstellungen möglich. Die verfügbaren Optionen werden in der Datei `replication.conf` zusammen mit kommentierten Beschreibungen aufgeführt.

Innerhalb der Datenbank sollte die Replikation mit der folgenden DDL-Anweisung aktiviert werden:

```
ALTER DATABASE ENABLE PUBLICATION
```

Definieren eines benutzerdefinierten Replikationssatzes

Optional sollte das Replikationsset (auch als Publikation bezeichnet) definiert werden. Es enthält Tabellen, die repliziert werden sollen. Dies geschieht mit den folgenden DDL-Anweisungen:

```
-- to replicate all tables (including the ones created later)
ALTER DATABASE INCLUDE ALL TO PUBLICATION

-- to replicate specific tables
ALTER DATABASE INCLUDE TABLE T1, T2, T3 TO PUBLICATION
```

Tabellen können später aus dem Replikationsset ausgeschlossen werden:

```
-- to disable replication of all tables (including the ones created later)
ALTER DATABASE EXCLUDE ALL FROM PUBLICATION

-- to disable replication of specific tables
ALTER DATABASE EXCLUDE TABLE T1, T2, T3 FROM PUBLICATION
```

Tabellen, die für die Replikation in der Datenbank aktiviert sind, können zusätzlich durch zwei Einstellungen in der Datenbank gefiltert werden: replication.conf: `include_filter` und `exclude_filter`.

Synchrone/Asynchrone Modi

Synchroner Modus

Die synchrone Replikation kann durch die Angabe von `sync_replica` aktiviert werden, wobei ein Verbindungsstring zur *Replikat*-Datenbank mit vorangestelltem Benutzernamen und Passwort angegeben wird. Mehrere Einträge sind zulässig, so dass Sie viele Replikate definieren können.

Asynchroner Modus

Für die asynchrone Replikation muss der Journaling-Mechanismus eingerichtet werden. Der primäre Parameter ist `journal_directory`, der den Speicherort des Replikationsjournals definiert. Die Angabe dieses Ortes schaltet die asynchrone Replikation ein und weist den Firebird Server an, mit der Produktion der Journalsegmente zu beginnen.

Eine minimale Konfiguration

Eine kleine primärseitige Konfiguration würde wie folgt aussehen:

```
database = c:\db\MASTER.FDB
{
    journal_directory = c:\db\journal_directory\
    journal_archive_directory = c:\db\journal_archive_directory\
}
```

Die Archivierung erfolgt durch den Firebird Server, der die Segmente von `c:\db\journal_directory\` zu `c:\db\journal_archive_directory\`.

Sie können jedoch die Einrichtung mit benutzerdefinierter Archivierung erstellen. Die benutzerdefinierte Archivierung mit der Einstellung `journal_archive_command` ermöglicht die Verwendung eines beliebigen System-Shell-Befehls, einschließlich Skripten oder Batch-Dateien, um Segmente an die Replikat-Seite zu liefern. Dabei kann die Komprimierung, FTP oder eine andere auf dem Server verfügbare Methode verwendet werden.

Das gleiche Setup wie oben, wobei die Archivierung alle 10 Sekunden erfolgt:

```
database = c:\db\MASTER.FDB
{
    journal_directory = c:\db\journal_directory\
    journal_archive_directory = c:\db\journal_archive_directory\
    journal_archive_timeout = 10
}
```

Sie können in der Datei `replication.conf` (im Firebird Root-Verzeichnis) viele weitere mögliche Einstellungen finden.

Anwendung der Einstellungen der Primärseite

Damit die Änderungen an den primärseitigen Einstellungen wirksam werden, müssen alle mit einer Datenbank verbundenen Benutzer getrennt werden (oder die Datenbank muss heruntergefahren werden). Wenn sich danach alle Benutzer erneut verbinden, wird die aktualisierte Konfiguration verwendet.

Einrichten der Replikat-Seite

Die Datei `replication.conf` wird auch zum Einrichten der Replikat-Seite verwendet. Das Setzen des Parameters `journal_source_directory` legt den Ort fest, den der Firebird Server nach den übertragenen Segmenten durchsucht. Darüber hinaus kann der DBA explizit angeben, welche Quelldatenbank für die Replikation akzeptiert wird, indem er den Parameter `source_guid` setzt.

Ein Beispiel für eine Replikationseinrichtung

Eine Konfiguration für eine Replik könnte wie folgt aussehen:

```
database = c:\db\REPLICA.FDB
{
    journal_source_directory = c:\db\journal_archive_directory\
    source_guid = {6F9619FF-8B86-D011-B42D-00CF4FC964FF}
}
```

Lesen Sie die `replication.conf` für andere mögliche Einstellungen.

Anwenden der Einstellungen auf der Replikat-Seite

Um Änderungen an den Einstellungen auf der Replikat-Seite zu übernehmen, muss der Firebird Server neu gestartet werden.

Erstellen einer Replikat-Datenbank

Aufgabe 1 - Anfertigung des ersten Replikats

Jede physische Kopiermethode kann verwendet werden, um eine erste Replik der primären Datenbank zu erstellen, aber der einfachste Weg ist eine Kopie auf Dateiebene, während der Firebird Server heruntergefahren ist.

Aufgabe 2 - Aktivieren Sie den Zugriffsmodus für Replikate

Zum Aktivieren des Zugriffsmodus - für die kopierte Datenbank - wird das Kommandozeilenprogramm *gfix* mit dem neuen `-Replica`-Switch und entweder `read_only` oder `read_write` als Argument:

- **Die Datenbankkopie als schreibgeschütztes Replikat festlegen**

  Starten Sie *gfix* aus dem Firebird 4 Stammverzeichnis in der Kommandozeile:

  ```
  gfix -replica read_only <database>
  ```

Wenn das Replikat schreibgeschützt ist, kann nur die Replikator-Verbindung die Datenbank ändern. Dies ist vor allem für Hochverfügbarkeitslösungen gedacht, da die Replikat-Datenbank garantiert mit der primären Datenbank übereinstimmt und für eine schnelle Wiederherstellung verwendet werden kann. Normale Benutzerverbindungen können alle Operationen durchführen, die für schreibgeschützte Transaktionen erlaubt sind: aus Tabellen auswählen, Nur-Lese-Prozeduren ausführen, in globale temporäre Tabellen schreiben usw. Datenbankwartung wie Sweeping, Shutdown, Monitoring ist ebenfalls erlaubt.

Eine schreibgeschützte Replik kann nützlich sein, um schreibgeschützte z. B. Analysen, von der Master-Datenbank weg zu verteilen. Schreibgeschützte Verbindungen können zu Konflikten mit der Replikation führen, wenn DDL-Anweisungen, die auf der Master-Datenbank ausgeführt werden, eine exklusive Sperre für Metadaten erfordern.

- **So legen Sie die Datenbankkopie als Lese- und Schreib-Replikat fest**

  Wiederum mit *gfix* in der Kommandozeile:

  ```
  gfix -replica read_write <database>
  ```

  Bei Replikaten mit Lese- und Schreibzugriff können sowohl die Replikator-Verbindung als auch normale Benutzerverbindungen die Datenbank gleichzeitig ändern. In diesem Modus gibt es keine Garantie, dass die Replikat-Datenbank mit der Master-Datenbank synchronisiert ist. Daher wird die Verwendung eines Read-Write-Replikats für Hochverfügbarkeitsbedingungen nicht empfohlen, es sei denn, die Benutzerverbindungen auf der Replikat-Seite sind darauf beschränkt, nur Tabellen zu ändern, die von der Replikation ausgeschlossen sind.

Aufgabe 3 — Umwandlung der Replikation in eine reguläre Datenbank

Ein drittes `gfix -replica`-Argument steht zur Verfügung, um die Replikation auf ein Read-Write-Replikat "auszuschalten", wenn die Bedingungen es erfordern, dass der Replikationsfluss aus irgendeinem Grund unterbrochen wird. In der Regel wird es verwendet, um das Replikat nach einem Ausfall zur primären Datenbank zu machen oder um physische Sicherungskopien vom Replikat zu erstellen.

```
gfix -replica none <database>
```

Beispiel und Übung aus der Praxis

Laden Sie die 64-Bit-Version manuell herunter und führen sie als Anwendung aus (vielleicht haben Sie bereits Firebird 2.5 und/oder Firebird 3 als Dienst lokal laufen), aber Sie können Ihre bevorzugte Installations-methode verwenden.

Entpacken Sie unser Firebird-4.0.0.2496-1-x64.zip-Archiv in ein benutzerdefiniertes Verzeichnis, zum Beispiel `C:\Firebird\Firebird-4.0.0.2496-1-x64\`. Wie oben erwähnt, wird es hier als Anwendung ausgeführt, indem die `firebird.exe -a` in der Befehlszeile im obigen Verzeichnis ausgeführt wird. Stellen Sie sicher, dass Sie den Standard Firebird 4 Port auf einen benutzerdefinierten Wert ändern, wenn Sie bereits ältere (oder dieselben) Firebird Instanzen laufen haben, indem Sie folgendes zur `firebird.conf` hinzufügen:

```
RemoteServicePort = 3054
IpcName = FIREBIRD4
```

Da wir hier aus einer Zip-Datei installieren, muss außerdem die *Sicherheitsdatenbank* manuell initialisiert werden

1. **Stoppen Sie den Firebird Server.** Firebird 4 speichert Verbindungen zur Sicherheitsdatenbank im Cache aggressiv. Das Vorhandensein von Serververbindungen kann *isql* daran hindern, eine eingebettete Verbindung herzustellen.
2. Starten Sie in einer geeigneten Shell eine interaktive *isql*-Sitzung mit der Befehlszeile `isql.exe` aus dem Firebird Root-Verzeichnis und öffnen Sie die *Employee*-Datenbank über ihren Alias:

```
isql -user sysdba employee
```

3. Erstellen Sie den Benutzer SYSDBA:

   Sie können dies auf der Kommandozeile oder in IBExpert: *Nützliches / SQL Editor* tun:

```
SQL> create user SYSDBA password 'masterkey';
SQL> commit;
SQL> quit;
```

4. Um die Initialisierung abzuschließen, starten Sie den Firebird Server erneut. Jetzt können Sie sich über das Netzwerk mit dem SYSDBA Passwort an den Datenbanken anmelden, auch an der Sicherheitsdatenbank.

Gemeinsam für alle Szenarien:

Bevor Sie mit der Replikation beginnen, müssen Sie eine physische Kopie der Master-Datenbank erstellen, z. B. durch einfaches Kopieren/Einfügen, während der Server nicht läuft. Kopieren Sie also `MASTER.FDB` und fügen Sie sie unter einem neuen Namen ein, z. B. `REPLICA.FDB` oder einen anderen Namen Ihrer Wahl.

Der nächste Schritt ist die Ausführung von:

```
gfix -replica read_only c:\db\REPLICA.FDB -user sysdba -pass masterkey
```

Verbinden Sie sich mit der Master-Datenbank und führen Sie die folgende DDL aus:

```
ALTER DATABASE INCLUDE ALL TO PUBLICATION;
ALTER DATABASE ENABLE PUBLICATION;
```

Szenario 1:

Versuchen Sie, die Replikation auf dieser einzelnen Firebird 4 Instanz zu testen, und verwenden Sie den *synchronen* Modus. Nun fügen Sie folgende Ergänzung in die Datei `replication.conf` (die sich im Firebird Stammverzeichnis befindet, zusammen mit der Datei `firebird.conf` Datei) ein:

```
database = c:\db\MASTER.FDB
{
    sync_replica = SYSDBA:masterkey@localhost/3054:c:\db\REPLICA_SYNC.FDB
}
```

Firebird 4 hier die einzige Instanz ist, ist das alles was benötigt wird.

Führen Sie nun die Common-Prozedur mit dem Namen `REPLICA_SYNC.FDB` aus.

Es ist jetzt an der Zeit, Firebird zu starten, wie es weiter oben beschrieben wurde:

```
firebird.exe -a
```

Jetzt sollte die Replikation Ihrer Datenbank beginnen. Sie können versuchen, Daten hinzuzufügen oder zu löschen, und alle Operationen sollten sofort in Ihrer Replikat-Datenbank dupliziert werden.

Scenario 2:

Lassen Sie uns den *asynchronen* Modus verwenden:

Diesmal muss die Datei `replication.conf` geändert werden, indem Folgendes hinzugefügt wird:

```
database = c:\db\MASTER.FDB
{
 journal_directory = c:\db\journal_directory\
 journal_archive_directory = c:\db\journal_archive_directory\
 journal_archive_timeout = 10
 verbose_logging = true
}
database = c:\db\REPLICA_ASYNC.FDB
{
 journal_source_directory = c:\db\journal_archive_directory\
 verbose_logging = true
}
```

Beachten Sie, dass es dieses Mal zwei Datenbankeinträge gibt. Der Grund dafür ist, dass diese einzelne Firebird 4 Instanz für den Umgang mit Journaldateien verantwortlich ist.

Führen Sie nun die Common-Prozedur mit dem Namen `REPLICA_SYNC.FDB` aus.

In diesem Fall können Sie feststellen, dass die Protokollierung aktiviert ist, so dass Sie leicht den Status des Replikationsprozesses in `replication.log` überprüfen können, die sich in Ihrem Firebird 4 Root-verzeichnis befindet.

Szenario 3:

Wenn Sie möchten, können Sie sowohl *synchrone* als auch *asynchrone* Replikationsmodi aktivieren, indem sie diese Replikationskonfiguration eingeben:

```
database = c:\db\MASTER.FDB
{
 sync_replica = SYSDBA:masterkey@localhost/3054:c:\db\REPLICA_SYNC.FDB
 journal_directory = c:\db\journal_directory\
 journal_archive_directory = c:\db\journal_archive_directory\
 journal_archive_timeout = 10
 verbose_logging = true
}
```

```
database = c:\db\REPLICA_ASYNC.FDB
{
 journal_source_directory = c:\db\journal_archive_directory\
 verbose_logging = true
}
```

Vergessen Sie nicht, die Common-Prozedur mit dem Namen `REPLICA_SYNC.FDB` auszuführen.

Szenario 4:

Lassen Sie uns eine weitere Firebird 4 Instanz einführen, für diese Übung auf dem `localhost`.

Die Firebird Installation wird nun so repliziert und eingerichtet, dass sie einen anderen Port überwacht:

```
RemoteServicePort = 3055
IpcName = FIREBIRD4_2
```

Sie können die *asynchrone* Replikation wie folgt einrichten:

1. Ändern Sie auf der Instanz 3054 die Datei `replication.conf`:

```
database = c:\db\MASTER.FDB
{
 journal_directory = c:\db\journal_directory\
 journal_archive_directory = c:\db\journal_archive_directory\
 journal_archive_timeout = 10
 verbose_logging = true
}
```

2. Ändern Sie auf der Instanz 3055 die Datei `replication.conf`:

```
database = c:\db\REPLICA_ASYNC.FDB
{
 journal_source_directory = c:\db\journal_archive_directory\
 verbose_logging = true
}
```

Vergessen Sie wie üblich nicht, die Common-Prozedur mit dem Namen `REPLICA_ASYNC.FDB` auszuführen.

Nach dem Neustart beider Instanzen kann nun eine Instanz den Replikationsprozess einleiten und Journaldateien erstellen, während die zweite Instanz Journaldateien verarbeitet und auf die Replikat-Datenbank anwendet.

Beachten Sie, dass wenn Sie zwei verschiedene Server verwenden, `journal_archive_directory` und `journal_source_directory` (jeweils pro Instanz) auf ein Netzlaufwerk verweisen müssen, auf das beide Instanzen zugreifen können und rechnen Sie mit einer gewissen Verzögerung bei der Verarbeitung der Journaldateien.

Szenario 5:

Sync-Replikation auf der sekundären Instanz, definieren Sie die asynchrone Replikationseinrichtung wie folgt:

```
database = c:\db\MASTER.FDB
{
    sync_replica = SYSDBA:masterkey@localhost/3055:c:\db\REPLICA_SYNC.FDB
}
```

Auf der sekundären Instanz gibt es nichts zu konfigurieren..

Vergessen Sie wie üblich nicht, die Common-Prozedur mit dem Namen `REPLICA_ASYNC.FDB` auszuführen.

Szenario 6:

Es besteht die Möglichkeit, mehrere synchrone Replikate zu aktivieren, indem man sie in `replication.conf` auflistet:

```
sync_replica = SYSDBA:pwd1@myserver1:c:\db1\REPLICA_SYNC.FDB
sync_replica = SYSDBA:pwd2@myserver2:c:\db2\REPLICA_SYNC.FDB
sync_replica = SYSDBA:pwd3@myserver3:c:\db3\REPLICA_SYNC.FDB
```

Vergessen Sie wie üblich nicht, die Common-Prozedur mit dem Namen `REPLICA_SYNC.FDB` auszuführen.

Verwendung der Replik als Haupt-Datenbank

Wenn Sie das Replikat zur primären Datenbank machen wollen (zum Beispiel nach einem Ausfall), sollten Sie diesen Befehl ausführen:

```
gfix -replica none <database> -user sysdba -pass masterkey
```

Wie bereits erwähnt, kann das Nur-Lese-Replikat als Hot-Standby-Server und zur Verteilung der Nur-Lese-Last von der Master-Datenbank verwendet werden.

Wie oben schon angesprochen, erlauben Read-Write-Replikate jedoch, dass sowohl die Replikator-Verbindung als auch reguläre Benutzerverbindungen die Datenbank gleichzeitig ändern können, so dass es keine Garantie dafür gibt, dass die Replikat-Datenbank mit der Master-Datenbank synchronisiert ist.

Die bidirektionale Replikation wird also leider nicht von Haus aus unterstützt.