Willkommen zum Informix Newsletter

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

auch diese Ausgabe des Informix Newsletters steht, wie die Ausgabe Q1/2024, zum grossen Teil unter dem Motto "Informix Replikation".

Die aktuellen Einsätze im Consulting hatten Bezug zum Thema Enterprise Replikation, wodurch sich der Inhalt des Newsletters bei den Tests ergab.

Nach längerer Zeit fast ausschliesslich im HomeOffice, kommt es nun wieder vermehrt zu Vorort Einsätzen. Dabei kam die Erkenntnis, dass Reisen mit der DB dann funktionieren, wenn Alle Züge Verspätung haben, und damit der verspätete Anschlusszug noch da ist, wenn man sehr verspätet ankommt.

Wir ziehen uns nun für den heissen Sommer hinter die kühlen Mauern im Schloss zurück und sind pünktlich zur Ausgabe Q3 wieder da.

Ihr TechTeam

TechTipp: HDR zu RSS ändern (und umgekehrt)

Bei der Replikation gibt es sowohl HDR (High availability Data Replication) Server, als auch RSS (Remote Standalone Server). Die Technik HDR wurde mit Version 7.12 eingeführt und basiert auf einer Verbindung, über die der Inhalt der LogBuffer transferiert wird. RSS wurde mit Version 11.10 eingeführt. Hierbei werden die abzuarbeitenden Informationen über mehrere Threads zum Ziel übertragen.

HDR und RSS unterscheiden sich in einigen Eigenschaften.
HDR:

- Kann im Modus "synchron" betrieben werden, so dass ein Commit am Primary Server erst abgeschlossen ist, wenn dieser auch am Secondary Server erfolgt ist.
- Kann die Primary Instanz blockieren, wenn das Nacharbeiten der Transaktionslogs am Secondary Server den Umfang von 12xLogBuffer übersteigen würde.
- Kann zum Primary werden ohne Zwischenschritte.
- Es kann nur einen HDR Secondary Server geben.

RSS:

- Kann mit Delay betrieben werden (kontinuierlicher Abstand zum Zustand des Primary Servers wird gehalten).
- Kann für einem "dbexport" genutzt werden (nach STOP_APPLY).
- Es kann viele RSS geben.

Wer sich für eine der Replikationsarten entschieden hat, der ist jedoch nicht auf Dauer darauf festgelegt.

Mit folgenden Befehlen kann die Replikationsart geändert werden:
HDR -> RSS
1. HDR Secondary mit „onmode -ky“ Offline nehmen.
2. Den Primary Server mittels “onmode -d standard” zum Standard Server ändern.
3. Am Primary Server den neuen RSS bekannt machen:
   „onmode -d add RSS <secname>“
4. Am Secondary Server den Physical Recovery starten: „oninit -PHY“
5. Am Secondary Server den Modus auf RSS ändern: “onmode -d RSS <primaryname>”
6. Mittels "onstat -g rss" oder "onstat -g cluster" die Replikation prüfen.

Die Gegenrichtung ist ebenfalls mögich:
RSS -> HDR
1. Sicherstellen, dass kein DELAY eingestellt ist (ONCONFIG: DELAY_APPLY = 0)
2. Den RSS mittels “onmode -ky” Offline nehmen.
3. Am Primary Server den bisherigen RSS löschen:
   “onmode -d delete RSS <secname>”
3. Am Secondary Server den Physical Recovery starten: “oninit -PHY”
4. Dem Primary mitteilen, dass er Primary ist: “onmode -d primary <secname>”
5. Dem Secondary Server mitteilen, dass er nun Secondary Server ist:
   “onmode -d secondary <primaryname>”
6. Mittels "onstat -g dri" oder "onstat -g cluster" die Replikation prüfen.

TechTipp: CDR - UTF8 mit SQL_LOGICAL_CHAR

Im Informix Newsletter Q2/2022 haben wir bereits beschrieben, wie eine Migration von einem 8-Bit auf einen UFT8 Zeichensatz mittels Enterprise Replikation durchgeführt werden kann. Hierfür kann bei der Definition der Replikate der Schalter "--UTF8 y" gesetzt werden, was für eine implizite Umsetzung der Zeichen bei der Replikation sorgt.

Da die Konvertierung auf UTF8 die Länge von Zeichenketten verändern kann, sollte dies auf der Zielseite berücksichtigt werden. So benötigt z.B. ein Umlaut statt einem Byte nun zwei (oder mehr) Byte in UTF8.

Zur Anpassung bestehen folgende Optionen:

- Anpassung der Spaltenlängen auf Quelle und Ziel vor der Definition der Replikate. Dies ist die eleganteste Lösung, da nur die Spalten geändert werden müssen, in denen auch Zeichen vorkommen, die Multibyte Zeichen sein können.
- Der ONCONFIG Parameter SQL_LOGICAL_CHAR zur automatischen Erweiterung aller Spalten vom Typ CHAR und VARCHAR und den Faktor 2, 3 oder 4, je nach gesetztem Wert **beim Erstellen der Datenbank**.

Wird die Erweiterung mittels SQL_LOGICAL_CHAR vorgenommen, so kann der benötigte Platz für die Daten erheblich ansteigen.

Bei der Definition der Replikation wird überprüft, ob die Datentypen und die definierte Länge der Spalten von Quelle und Ziel übereinstimmen.

Der Parameter SQL_LOGICAL_CHAR verändert die interne Speicherung, wodurch in der Tabelle "syscolumns" z.B. für eine Spalte vom Datentyp char(10) die Länge 20 angegeben wird. Die Ausgabe mittels "dbschema" hingegen zeigt korrekterweise die Länge 10.

Das folgende Beispiel zeigt, dass selbst bei der automatischen Anlage der Tabelle im Ziel (Schalter -u) die Struktur im Ziel als Abweichung zum Fehler führt:

(Die Datenbank am Replikations-Server "r_1" ist als "de_de.8859-1" erstellt, die Datenbank am Replikations-Server "r_2" mit "en_us.utf8" und SQL_LOGICAL_CHAR=2)

```
cdr define repl repl1 -c r_1 -A -R -C ignore -i -f y -S trans --UTF8 y -u
     "P kalu@r_1:informix.tab1"  "select * from informix.tab1"
     "R kalu@r_2:informix.tab1"  "select * from informix.tab1"

Interpreting this replicate as a master replicate.
Verification of kalu@r_1:informix.tab1 started
Verification of kalu@r_1:informix.tab1 is successful
Verification of kalu@r_2:informix.tab1 started
Creating table...
create table informix.tab1 (
     f1   integer not null,
     f2   char(10),
     f3   date,
     primary key (f1));
```

```
Master Replicate Error
    Column mismatch

Master:       f2   char(10)
Participant:  f2   char(20)

1 Error(s)
Verification of kalu@r_2:informix.tab1 failed
Master Replicate Verification Total errors:1
```

Wird bei der Definition zuerst die Datenbank auf der UTF8-Seite angegeben, so wird das Replikat ohne Fehler erstellt:

```
cdr define repl repl1 -c r_1 -A -R -C ignore -i -f y -S trans --UTF8 y -u
     "R kalu@r_2:informix.tab1"  "select * from informix.tab1"
     "P kalu@r_1:informix.tab1"  "select * from informix.tab1"

Interpreting this replicate as a master replicate.
Verification of kalu@r_2:informix.tab1 started
Verification of kalu@r_2:informix.tab1 is successful
Verification of kalu@r_1:informix.tab1 started
Verification of kalu@r_1:informix.tab1 is successful
```

Hinweis: Hierbei muss die Tabelle im Ziel existieren, da der Autocreate in dieser Reihenfolge nicht funktionieren kann. Der Schalter -u ist somit leider nicht nutzbar.

Bei der Definition der Replikate ist daher zu beachten, dass als erste Zeile der Definition immer die Zieldatenbank (UTF8-Seite) angegeben werden muss, um bei der Prüfung nicht in einen Fehler zu laufen.

TechTipp: CDR - Where-Bedingung mit 'DATE'

Im Informix Newsletter Q1/2024 wurde beschrieben, wie mittels der Angabe einer WHERE-Bedingung bei der Definition eines Replikats die zu übertragenden Daten eingeschränkt werden können.
Dabei bietet es sich an, während einer Migration die "aktuellen" Daten in die neue Produktion zu replizieren, die älteren Daten in eine Archivdatenbank.
In der aktuellen Version bekommt man beim Vergleich auf ein Datumsfeld allerdings eine Fehlermeldung "command failed -- Source and Target do not have the same data type (202)".
Hierzu gibt es einen Workaround, der einfach zu implementieren ist: Ein Cast auf DATE()

Beispiel:

```
cdr define repl repl2 -c r_1 -A -R -C ignore -i -f y -S trans --UTF8 y -u
     "P kalu@r_1:informix.tab1"
          "select * from informix.tab1 where date(f3) > date('20.05.2024')"
     "R kalu@r_2:informix.tab1"
          "select * from informix.tab1 where date(f3) > date('20.05.2024')"
```

TechTipp: CDR ALTER - Änderungen erlauben

Tabellen, auf denen Replikate definiert sind, können nicht einfach mittels "alter table" geändert werden, da die Struktur im Replikat genutzt wird.

Um Änderungen zu erlauben, wie z.B. das Hinzufügen von Spalten an eine Tabelle, kann die Tabelle mittels "cdr alter" vorbereitet werden.

Beispiel:

```
cdr alter --on kalu:informix.tab1
dbaccess ... Struktur der Tabelle anpassen wie z.B. alter table tab1 add f4 int;
cdr alter --off kalu:informix.tab1
```

Im online.log werden diese Anweisungen protokolliert:

```
05/22/2024 10:14:37   ER: Alter mode set for table.
          Table: kalu:'informix'.tab1.

WARNING:  While the table is in alter mode Enterprise Replication cannot
          apply transactions involving this table or in a referential
          relationship with this table.  Enterprise Replication also
          cannot apply transactions from the same originating site
          that follow the first transaction it cannot apply.
```

Nachdem die Änderungen auf beiden Seiten vorgenommen wurden, muss das Replikat mittels "remaster" angepasst werden.

TechTipp: CDR Remaster - Strukturanpassung

Nachdem die Tabelle auf allen beteiligten Servern angepasst wurde, müssen die Replikate auf die neue Struktur angepasst werden.
(Im Beispiel betrifft dies die Übertragung der Inhalte der neuen Spalte f4).

```
cdr remaster -c r_1 -M r_1 repl1 "select * from tab1"
```

Handelt es sich beim Ziel um eine Datenbank, die mit SQL_LOGICAL_CHAR erstellt wurde, so muss dieser Aufruf von Remaster mit Bezug auf diese Datenbank aufgerufen werden. Bezogen auf unser Beispiel bekäme man den Fehler:

```
Master Replicate Error
    Column mismatch
Master:       f2   char(10)
Participant:  f2   char(20)
1 Error(s)
Failure
command failed -- An error occurred concerning a mastered replicate (125)
```

Die Lösung ist auch hier, die UTF8 Seite als Master anzugeben:

```
cdr remaster -c r_2 -M r_2 repl1 "select * from tab1"
```

TechTipp: CDR - Initiale Datenübertragung

Beim Start einer Replikation stellt sich die Frage
"Wie bekomme ich die Daten auf die Zielseite ?"

Die beste Option ist, mittels "dbschema" die Definition der Tabellen in der Quelle zu sichern und diese im Ziel (ohne Indexe und Constraints) einzuspielen.
Um einen Konsistenzcheck vorzubereiten, sollte hierbei gleich, wie auf der Quelle, die Schattenspalte "ifx_replcheck" mittels "WITH REPLCHECK" angefügt werden.

Die Spalte ifx_replcheck kann explizit ausgegeben werden:

```
select *, ifx_replcheck
from tab1

         f1 f2                   f3                   f4         ifx_replcheck
         10 syssyntabl           20.04.2024                         2382310389
         11 sysconstra           20.04.2024                         3148577182
         12 sysreferen           20.04.2024                         2892792801
         13 syschecks            20.04.2024                          389932816
```

Für die grossen Tabellen werden die Daten mittels "unload" und "load" übertragen, wobei das Entladen in externe Tabellen parallelisiert werden kann, indem mehr als eine Datei als Basis der Externen Tabelle definiert wird. Optimal wären partitionierte Tabellen.
Beim Laden können diese Teiltabellen parallel geladen werden, wobei in der Enterprise Edition der Parameter PDQPRIORITY gesetzt werden sollte.
Um eine bessere Performance zu erzielen, können die Tabelle auf "raw" gesetzt werden, da ansonsten das Protokollieren in der Transaktionslogs viel Zeit kostet.

Achtung: Beim Laden muss ggf. die Codeset Conversion berücksichtigt werden, indem die CLIENT_LOCALE auf den Wert der bisherigen Datenbank eingestellt wird, die DB_LOCALE auf den neuen Codeset UTF8.

Sind die grossen Tabellen geladen, werden diese wieder auf Mode "standard" gesetzt. Nun werden die Primary Keys und so weit notwendig weitere Indexe erstellt (was in der Enterprise Edition mit gesetzter PDQPRIORITY deutlich schneller geht).

Anschliessend werden die Replikate definiert.
Für die kleineren Tabellen werden die Daten mittels "cdr sync" übertragen.
(siehe Artikel zu CDR SYNC)

Tabellen, die sich referenzieren (Foreign Keys), müssen in ReplicateSets zusammengefasst werden.

Erst nachdem alle Daten geladen wurden, sollten die Foreign Keys (falls notwendig) auf dem Ziel erstellt werden.

TechTipp: CDR CHECK - Performance

Der "cdr check" nutzt die interne Funktion ifx_checksum() um die Werte zwischen Quelle und Ziel zu überprüfen. Auf grossen Tabellen kann dies sehr lange dauern.
Werden die Tabellen mit der Schattenspalte "ifx_replcheck" vorbereitet, indem ein "alter table <tabname> ADD REPLCHECK" vor dem Start der Replikation vorgenommen wird, ist die Zeit für Check und Repair der Tabellen deutlich kürzer.
Auf der Schattenspalte ifx_replcheck sollte ein Index erstellt werden mit dem Primary Key und der Spalte ifx_replckeck:

```
create index i_ifx_replcheck on tab1(id, ifx_replcheck);
```

Achtung: Der "alter table ADD REPLCHECK" ist ein "Slow Alter", der die Tabelle komplett umspeichert und dafür zwischenzeitlich den doppelten Platz der Tabelle benötigt. Zudem ist die Tabelle in dieser Zeit gegen Zugriffe gesperrt.

Der Aufruf für den Check mit Repair (-R) lautet:

```
cdr check repl -c r_1 -m r_1 -r repl1 r_2 -R
```

Sind die Daten bis zu einem gewissen Zeitraum sicher konsistent, und es sollen nur die neuen Daten geprüft werden, so kann dies mittels einer weiteren Option "--since" bzw. "-S" erfolgen.

Als Argumente für "since" sind unterstützt:

- H z.B. 23H für 23 Stunden
- M z.B. 13M für 13 Minuten
- D z.B. 42D für 42 Tage
- W z.B. 7W für 7 Wochen
- 'timestamp' '2024-02-23 00:42:23' Änderungen nach 23.02.2024 0:42:23 Uhr

Dieser Filter "since" ist sowohl für den Check, als auch den Repair gültig.
Mit gesetzten "since" werden nur die neueren Daten überprüft was sich auch an der Anzahl der Werte in der Ausgabe zeigt.

```
cdr check repl -c r_1 -m r_1 -r repl1 r_2 -R
May 22 2024 16:33:50 ------   Table scan for repl1 start  --------
Node                  Rows     Extra   Missing  Mismatch Processed
--------------- --------- --------- --------- --------- ---------
r_1                     72         0         0         0         0
r_2                     72         0         0         0         0
May 22 2024 16:33:50 ------   Table scan for repl1 end   ---------

cdr check repl -c r_1 -m r_1 -r repl1 r_2 -R -S '2024-05-20 00:00:00'
May 22 2024 16:33:52 ------   Table scan for repl1 start  --------
Node                  Rows     Extra   Missing  Mismatch Processed
--------------- --------- --------- --------- --------- ---------
r_1                      0         0         0         0         0
r_2                      0         0         0         0         0
May 22 2024 16:33:52 ------   Table scan for repl1 end   ---------
```

TechTipp: CDR CHECK REPAIR - Options

Sind die Daten der Tabellen in der Replikation nicht synchron, so können diese mittels "Repair" synchronisiert werden.
Dabei stehen folgende Optionen zur Verfügung:

- -e delete # Zusätzliche Daten werden im Ziel gelöscht, unterschiedliche Daten # werden entsprechend dem Master eingefügt, bzw. upgedatet
- -e keep # Daten des Masters werden an das Ziel übertragen und ggf. # angeglichen, zusätzliche Daten bleiben erhalten.
- -e merge # Die Daten werden entsprechend dem Master angeglichen, # Zusätzliche Daten werden vom Ziel an den Master übertragen.

Beispiel:

```
cdr check repl -c r_1 -m r_1 -r repl1 r_2 -R -e merge

May 22 2024 16:43:37 ------   Table scan for repl1 start  --------

Node                  Rows     Extra   Missing  Mismatch Processed
--------------- --------- --------- --------- --------- ---------
r_1                     72         0         0         0        10
r_2                     71         4         5         5         4

The repair operation completed. Validating the repaired rows ...
Validation of repaired rows failed.
WARNING: replicate is not in sync
May 22 2024 16:43:42 ------   Table scan for repl1 end   ---------

command failed -- WARNING: replicate is not in sync (178)


cdr check repl -c r_2 -m r_1 -r repl1 r_2

May 22 2024 16:47:57 ------   Table scan for repl1 start  --------

Node                  Rows     Extra   Missing  Mismatch Processed
--------------- --------- --------- --------- --------- ---------
r_1                     72         0         0         0         0
r_2                     76         4         0         0         0

WARNING: replicate is not in sync
```

Da hier das Ziel eine UTF8 Datenbank ist, können die dort eingefügten Datensätze nicht an die Quelle zurück übertragen werden. Daher konnte der Merge nur die "Missing" Daten ergänzen, und den "Mismatch" anhand des Master angleichen.

Mit der Option "-e delete" könnten die Daten wieder synchronisiert werden.

TechTipp: CDR SYNC - Übertragen initialer Daten

Die Inhalte kleinerer Tabellen können nach der Definition der Replikate mittels "cdr sync" übertragen werden. Hierbei werden die Daten der Quelle sequentiell gelesen und mit der Option "Always Apply" im Ziel eingetragen.
Dabei werden die Daten im Ziel eingefügt.

Die Syntax hierzu ist:

```
cdr sync repl -c r_1 -m r_1 -r repl1 --memadjust=500M --background r_2
```

Für eine schnellere Übertragung kann der Speicher, der zur Übertragung genutzt wird, angepasst werden. Der Befehl kann sowohl im Vordergrund, als auch mit der Option "--background" im Hintergrund laufen. Der Fortschritt kann über den Aufruf "cdr stats sync" betrachtet werden.
Als weitere Optionen steht die Behandlung der "Extra Rows" im Ziel mit "keep, delete und merge", sowie die Wahl, ob Trigger im Ziel ausgelöst werden sollen, oder nicht, zur Wahl.

Hinweis: Für grosse Tabellen ist diese sequentielle Übertragung nicht geeignet, da der sequentielle Scan auf der Quelle zu lange dauert.

Nutzung des INFORMIX Newsletters

Die hier veröffentlichten Tipps&Tricks erheben keinen Anspruch auf Vollständigkeit.
Die IUG hat sich dankenswerterweise dazu bereit erklärt, den INFORMIX Newsletter auf ihren Web Seiten zu veröffentlichen.
Da uns weder Tippfehler noch Irrtümer fremd sind, bitten wir hier um Nachsicht, falls sich bei der Recherche einmal etwas eingeschlichen hat, was nicht wie beschrieben funktioniert.
Rückmeldungen hierzu sind herzlich Willkommen !
Die gefundenen Tippfehler dürfen zudem behalten und nach Belieben weiterverwendet werden.
Eine Weiterverbreitung in eigenem Namen (mit Nennung der Quelle) oder eine Bereitstellung auf der eigenen HomePage ist ausdrücklich erlaubt. Alle hier veröffentlichen Scripts stehen uneingeschränkt zur weiteren Verwendung zur Verfügung.

Die Autoren dieser Ausgabe

Andreas Legner **INFORMIX Advanced Support**
HCL Software

Martin Fuerderer **Database Development**
HCL Software

Gerd Kaluzinski **IBM Expert Lab DACH Consultant**
Data & AI Software
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Dank auch an die vielen Helfer im Hintergrund.

Nicht zu vergessen der Dank an die Informix User Group, ohne die es den INFORMIX Newsletters heute nicht mehr geben würde, und die dankenswerterweise die Verteilung übernimmt.

Foto Nachweis:
Schloss Wolfurt (Gerd Kaluzinski)