

The IBM Informix v.14.10.xC4 Technical Deep Dive webcast will start shortly

Carlton Doe

cdoe@us.ibm.com





International
Informix
Users Group

IBM Informix v.14.10.xC4 Technical Deep Dive Webcast series

Session 2: Replication enhancements

Session 3, August 26 - Java and system administration
enhancements

IBM INFORMIX V.14.10.XC4 - REPLICATION ENHANCEMENTS

v.1



Agenda

- Off-line conversion support for HDR and RS secondaries
- Smart trigger survival enhancements
- `cdr migrate server` **RI** constraint enhancements
- `cdr migrate server add_replcheck` **phase**
- Flow control delay statistics for RS and SD secondaries

Off-line conversion support for HDR and RS secondaries

Off-line conversion support

- Over time, the H/A cluster has seen functionality enhancements for supporting version upgrades
- Initially
 - You had to turn the cluster instances off resulting in downtime
 - Upgrade the primary with the new binary and turn it on
 - Upgrade the secondaries with the new binary but re-initialize the cluster using either `ontape` or `ON-Bar` backup and restore operations and `onmode` commands
- With Informix 11.70, a process for executing *inter*-version upgrades was introduced that preserved cluster uptime
 - For example Informix 11.7 to Informix v.12.10
 - This process is still available today if you want to use it
 - It involves converting the H/A cluster to an Grid cluster (the `sec2er` function), upgrading each instance to the new binary, converting the primary back to H/A cluster, re-initializing each instance back into the H/A cluster

Off-line conversion support

- In Informix v.12.10.xC5, the rolling upgrade functionality was introduced for *intra*-version upgrades that preserves uptime
 - This supports fixpak to fixpak upgrades
 - For example, xC4 to xC5 or xC5 to xC6
 - You can NOT go from xC4 to xC6
 - This functionality can NOT be used to roll back a version
 - The lowest level supported is Informix v.12.10.xC4 going to xC5

Off-line conversion support

- But what if you need to make a “major” upgrade?
 - It could be from .xC2 to .xC6
 - It could be from Informix v.12.10.n to Informix v.14.10
- You only option for upgrading the secondaries is to either take an outage and rebuild the secondaries
OR
- Convert the cluster to ER then upgrade and rebuild
 - But that involves unique key constraint problems and so on
- Informix v.14.10.xC4 introduces off-line secondary conversion for HDR and RS secondary instances for major upgrades
 - You’ll still have to take the instances off-line to convert but you won’t have to rebuild the secondary instances

Off-line conversion support

- This functionality supports in-place, offline upgrades from 11.70.xC1 to 14.10.xC4 or later
- Please note — this is a one way operation, reversion to an earlier Informix version (even the original secondary version) is NOT supported
- Enabling `CONVERSION_GUARD` for all instances is **strongly** recommended

Off-line conversion support

- What is the workflow for this process?
 - Disable FOC so a secondary instance is NOT promoted to primary
 - After ensuring `CONVERSION_GUARD` is enabled for all instances, shut down the primary instance
 - Updatable secondary instances will be blocked from attempting any DML operations
 - All secondary read operations will continue without being affected
 - Install the new Informix binary and make any `$ONCONFIG` changes needed for new functionality
 - Restart the primary
 - This instance will go through the upgrade process
 - All behind the scenes instance changes are logged in the logical logs!!!
 - If the upgrade fails, use `onrestorererept` to roll back the conversion changes, fix the issues then try again
 - So you know, during the conversion / upgrade process for the primary or secondary nodes, no end-user connections are allowed until the instance is fully converted

Off-line conversion support

- What is the workflow for this process?
 - With the primary successfully upgraded, the secondary instances will NOT re-connect because of the version mis-match checking
 - The secondary instances will still think the primary is off-line
 - Turn the first secondary off-line
 - Install the new Informix binary on the secondary, make any `$ONCONFIG` changes, and restart the instance
 - This instance will go through the upgrade process
 - Upgrade changes from the primary instance logical log files are sent to the secondary so they are applied
 - If the upgrade fails, use `onrestorer rept` to roll back the conversion changes, fix the issues then try again
 - Once the upgrade process has completed, execute an `onmode -c` command on the primary to force a cluster checkpoint and commit the changes on the secondary
 - The upgraded secondary will re-connect to the cluster primary and be fully functional

Off-line conversion support

- What is the workflow for this process?
 - Continue this process for each HDR and RS secondary in the cluster
 - Turn FOC back on
 - Congratulations, you've upgraded the cluster without having to rebuild the secondary instances!
- But what if I have a SD secondary instance?
 - Turn it off before making any changes to the cluster or other secondary instances
 - Once the other instances are upgraded and online
 - Install the new binary on the SD secondary node(s), make any `$ONCONFIG` changes, and turn them on
 - Since SD secondary instances use the same disks as the primary, no conversion is required



Smart trigger survival enhancements

Smart trigger survival enhancements

- As mentioned in the CSDK presentation, smart triggers were introduced in Informix v.12.10.xC9
 - They were created to work around conditions where an application must constantly ping the instance to see if specific data conditions exist so the application can do some work
 - For example, was data inserted into a table that must be picked up and sent to another application or target?
 - Using Java and JDBC, you can create a “smart trigger” that monitor changes to data and when a “triggering condition” occurs, send an alert and the data to the application to work on
- In Informix 12.10.xC10 additional enhancements were made including support for receiving triggering events after the application reconnects to the instance
- Earlier you saw that smart triggers can now be supported with the Python programming language
 - It's now publicly released

Smart trigger survival enhancements

- The original design for smart triggers was for single instance functionality and single session
 - The original design did not take into account instance failover in an H/A cluster for example nor to survive the disconnect and reconnect of a session
 - If the session disconnected, you started over
 - Smart trigger related objects were stored in memory and immediately cleared when the session or the instance died
- But smart triggers use has expanded significantly and are being deployed everywhere including production workloads protected by H/A and other cluster technologies
- It's no longer acceptable for smart trigger functionality to be lost if the session is interrupted
- In xC4, smart trigger session functionality has been enhanced to support both cases
 - Part of the solution involves writing session and event information into the `syscdr` database

Smart trigger survival enhancements

- Smart trigger session or server survivability requires the session have a new session designator — detachable
- To designate a session as detachable, the application must execute this type of operation
 - FYI - it doesn't work in dbaccess! 😊

```
execute function sysadmin:informix.task('pushdata setdetach') into :retvalstr;
```

- This returns a pseudo-session ID value into the *retvalstr* variable
 - It is NOT the instance session ID, lets call it a *cdr_session_id*
 - All smart trigger functionality for this session is tied to this *cdr_session_id*
- The application needs to capture this *cdr_session_id* for use later as shown in a moment

Smart trigger survival enhancements

- Once the session is qualified as detachable, client and session specific event information is stored in the `pushdata_client` and `pushdata_event` tables / views
 - The `pushdata_client` table contains a list of all the detachable sessions
 - The `pushdata_event` table contains general information for events
 - Either all events

```
onstat -g pd event
```

- Or generalized event information for the detached session invoking the smart trigger

```
onstat -g pd session_id event
```

- If a client dies then re-attaches, or there is a server failover, the event information stored in these tables can be processed when the client re-attaches

Smart trigger survival enhancements

- If a application session is detached and reconnects, regardless of cause (e.g. simple reconnect or failover condition), the new session can adopt its earlier `cdr_session_id` if it knows what it was

```
execute function sysadmin:informix.task('pushdata join',  
    "{session_id:\"cdr_session_id\"}") into :retvalstr;
```

- This function does NOT create a new `cdr_session_id`
 - It returns a smartBLOB file descriptor to read the event data

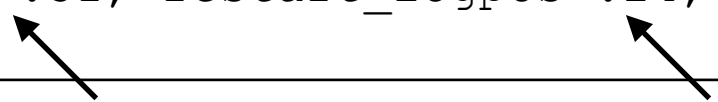
Smart trigger survival enhancements

- In the event of a failover, smart trigger event processing can restart on its own with the next event OR you can go back in time to make sure any events in flight around the failover are processed
- This is a manual process, so to speak
 - You are responsible for handling any duplicate events that might be processed again
- How do you know which moment / transaction to go back to?
 - With detachable sessions, new information about the transaction is returned to the session in a `json` document

Smart trigger survival enhancements

- With detachable sessions, additional information about the transaction is returned to the session in a `json` document
 - This information includes the active logical log ID as well as the log position
 - For example, from the documentation, the transaction report from an `insert` operation

```
{ "operation": "insert", "table": "t1", "owner": "nagaraju", "database": "testdb", "txnid": 133151498608, "operation_owner_id": 37188, "operation_session_id": 67, "commit_time": 1568823415, "op_num": 4, "restart_logid": 31, "restart_logpos": 24, "rowdata": { "c1": 4001, "c2": 4001 } }
```



- *The application is responsible* for capturing and storing the `restart_logid` and `restart_logpos` values from this return statement
 - They reflect the moment for that transaction

Smart trigger survival enhancements

- With this information, after an instance failover or restart, an administrator must review the captured transaction values and determine which moment to re-start from
- The following command resets the ER snoopy process to the given moment
 - Smart trigger events are replayed from that moment forward

```
execute function admin("pushdata reset_capture", '{"logid':"id_val",  
"logpos":"'pos_val' }');
```

- This operation should only be executed by the instance administrator and outside of a regular smart trigger session

Smart trigger survival enhancements

- To delete detachable session information from the instance so their session and events are no longer captured, there are several options

- To delete your current session

```
execute function informix:task('pushdata delete') into :retvalstr;
```

- To delete another session which has been disconnected from the instance requires knowing its `cdr_session_id`

```
execute function task('pushdata delete', '{session_id:"cdr_session_id"}') ;
```

- To delete all disconnected sessions

```
execute function task('pushdata delete', '{delete_all:"1"}');
```

Smart trigger survival enhancements


- Let's see some of this in action
- Using the smart trigger demo in `$INFORMIXDIR/demo/cdc . . .` and lots of help from Naragaju
 - Compile the smart trigger program (`pushdata.ec`)

```
Inst_1: pwd
Inst_1: cd $INFORMIXDIR/demo/cdc
Inst_1:
Inst_1: ls -l
total 2756
-rw-r--r--. 1 informix informix 51196 May 14 20:06 cdcapi.ec
-rw-rw-r--. 1 informix informix 54176 Jun 26 13:44 pushdata.c
-rw-r--r--. 1 informix informix 21051 May 14 20:06 pushdata.ec
Inst_1:
Inst_1: esql -static -o pushdata pushdata.ec
Inst_1:
Inst_1: ls -l
total 2756
-rw-r--r--. 1 informix informix 51196 May 14 20:06 cdcapi.ec
-rwxrwxr-x. 1 informix informix 1339440 Jun 26 13:44 pushdata
-rw-rw-r--. 1 informix informix 54176 Jun 26 13:44 pushdata.c
-rw-r--r--. 1 informix informix 21051 May 14 20:06 pushdata.ec
Inst_1:
```

Smart trigger survival enhancements

- The program is invoked with a 180 second timeout / close value on the `my_test:customer_full` table
 - The `cdr_session_id` is 2

```
Inst_1: pushdata -D my_test -T customer_full -o informix -s "select * from customer_full" -t 180 -p
Connected to sysadmin@inst_1
pushdata open session for server inst_1 with Timeout 180 Max recs per read 1
COMMAND: execute function informix.task('pushdata open')
COMMAND: execute function informix.task('pushdata setdetach')
Push-data client session unique id 2
COMMAND: execute function informix.task('pushdata register', {txnid:"0",timeout:"180",max_pending_ops:"0",maxrecs:"1"})
pushdata register return value OK
pushdata register of inst_1 on session 39
COMMAND: execute function informix.task('pushdata register', {table:"customer_full",owner:"informix",database:"my_test",query:"select * from customer_full"})
pushdata register return value OK
Start Reading the records...
```



Smart trigger survival enhancements

- A repl is started to support the smart trigger

```
Inst_1: cdr list repl

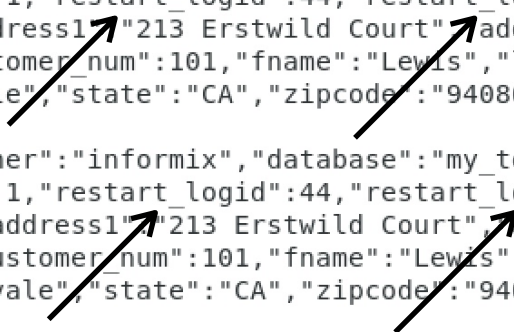
CURRENTLY DEFINED REPLICATES
-----
REPLICATE:      pushrepl_192_1593462178_903846598
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65537 / 0x10001
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

Inst_1: _
```


Smart trigger survival enhancements

- An end-user session is opened in `dbaccess` and performs two separate operations on `customer_num 101`
 - Change the first name from “Ludwig” to “Lewis”
 - Change the last name from “Pauli” to “Patrick”
- The transaction result values are captured by the application — including the new log ID and log position values

```
Start Reading the records...
bytesread is 686 loreaderr is 0 SQLCODE 0
RECORD: {"operation":"update","table":"customer_full","owner":"informix","database":"my_test","txnid":188984218240,"operation_owner_id":1000,"operation_session_id":310,"commit_time":1593462822,"op_num":1,"restart_logid":44,"restart_logpos":5554816,"rowdata":{"customer_num":101,"fname":"Lewis","lname":"Pauli","company":"All Sports Supplies","address1":"213 Erstwild Court","address2":"null","city":"Sunnyvale","state":"CA","zipcode":"94086","phone":"408-789-8075"},"before_rowdata":{"customer_num":101,"fname":"Lewis","lname":"Pauli","company":"All Sports Supplies","address1":"213 Erstwild Court","address2":"null","city":"Sunnyvale","state":"CA","zipcode":"94086","phone":"408-789-8075"}}
bytesread is 688 loreaderr is 0 SQLCODE 0
RECORD: {"operation":"update","table":"customer_full","owner":"informix","database":"my_test","txnid":188984226432,"operation_owner_id":1000,"operation_session_id":310,"commit_time":1593462837,"op_num":1,"restart_logid":44,"restart_logpos":5657216,"rowdata":{"customer_num":101,"fname":"Lewis","lname":"Patrick","company":"All Sports Supplies","address1":"213 Erstwild Court","address2":"null","city":"Sunnyvale","state":"CA","zipcode":"94086","phone":"408-789-8075"},"before_rowdata":{"customer_num":101,"fname":"Lewis","lname":"Pauli","company":"All Sports Supplies","address1":"213 Erstwild Court","address2":"null","city":"Sunnyvale","state":"CA","zipcode":"94086","phone":"408-789-8075"}}
bytesread is 690 loreaderr is 0 SQLCODE 0
```



Smart trigger survival enhancements

- Generalized information about the smart trigger session

```
Inst_1: onstat -g pd event

IBM Informix Dynamic Server Version 14.10.FC4DE -- On-Line (Prim)
-- 234176 Kbytes
push-data subsystem structure at 0x48076028
  push-data session structure at 0x47d1a028
    push-data sql session id: 317 0x13d
    Marked as detachable session, session unique id: 2
    Number of event conditions: 1

    Push-data event structure at 0x47d43028
      Full Table Name: my_test:informix.customer_full
User data:
  Replicate name: pushrepl_317_1593462760_970551784

Inst_1:
```

Smart trigger survival enhancements

- Generalized information about the detachable session executing the smart trigger

```
Inst_1: onstat -g ses

IBM Informix Dynamic Server Version 14.10.FC4DE -- On-Line (Prim) -- Up 00:39:55 -- 2

session
id      user      tty      pid      hostname  #RSAM   total    used      dynamic
334     informix -        0        -        0       16384    12504    off
317     informix 0        4741    localhost 1       278528   247824   off
310     testuser 1        4588    localhost 1       90112    72344    off
54      informix -        0        -        1       438272   363584   off
53      informix -        0        -        1       569344   470296   off
43      informix -        0        -        1       630784   486080   off
42      informix -        0        -        1       102400   87288    off
3       informix -        0        -        0       16384    14120    off
2       informix -        0        -        0       16384    12504    off

Inst_1:
Inst_1:
Inst_1: onstat -g pd 317 event

IBM Informix Dynamic Server Version 14.10.FC4DE -- On-Line (Prim) -- Up 00:40:13 -- 2
push-data subsystem structure at 0x48076028
  push-data session structure at 0x47d1a028
    push-data sql session id: 317 0x13d
    Marked as detachable session, session unique id: 2
    Number of event conditions: 1

    Push-data event structure at 0x47d43028
      Full Table Name: my_test:informix.customer_full
User data:
  Replicate name: pushrepl_317_1593462760_970551784
```

Smart trigger survival enhancements

- After testing with several sessions that closed due to an application timeout, there are several repls supporting the triggers
 - One for each detached session

```

Inst_1: cdr list repl

CURRENTLY DEFINED REPLICATES
-----
REPLICATE:      pushrepl_194_1593202508_642567685
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65537 / 0x10001
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_195_1593202723_1146843052
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65538 / 0x10002
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_206_1593202911_392619377
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65539 / 0x10003
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_212_1593203519_1507913588
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65540 / 0x10004
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

Inst_1:
  
```

Smart trigger survival enhancements

- You no longer want to use the smart trigger, or the sessions, so remove them from the system
 - You can remove them one session at a time, by `cdr_session_id`

```
Inst_1: cdr list repl

CURRENTLY DEFINED REPLICATES
-----
REPLICATE:      pushrepl_238_1593204140_26881050
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65541 / 0x10005
REPLMODE:       PRIMARY  ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_246_1593204183_1947453964
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65542 / 0x10006
REPLMODE:       PRIMARY  ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1
```

```
----- sysadmin@inst_1 ----- Press CTRL-W fo
execute function task('pushdata delete', '{session_id:"6"}') ;
```

```
Inst_1: cdr list repl

CURRENTLY DEFINED REPLICATES
-----
REPLICATE:      pushrepl_246_1593204183_1947453964
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65542 / 0x10006
REPLMODE:       PRIMARY  ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1
```


Smart trigger survival enhancements

- You no longer want to use the smart trigger, or the sessions, so remove them from the system
 - You can remove them all the sessions at once

```
Inst_1: cdr list repl

CURRENTLY DEFINED REPLICATES
-----
REPLICATE:      pushrepl_194_1593202508_642567685
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65537 / 0x10001
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_195_1593202723_1146843052
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65538 / 0x10002
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_206_1593202911_392619377
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65539 / 0x10003
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

REPLICATE:      pushrepl_212_1593203519_1507913588
STATE:          Active ON:g_inst_1
CONFLICT:       Always Apply
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    my_test:informix.customer_full
OPTIONS:        row,fullrow,cascade,pushdata
REPLID:         65540 / 0x10004
REPLMODE:       PRIMARY ON:g_inst_1
APPLY-AS:       INFORMIX ON:g_inst_1

Inst_1:
```

```
----- sysadmin@inst_1 ----- Press CTRL-W f
execute function task('pushdata delete', '{delete_all:"1"}');
```

```
Inst_1: dbaccess sysadmin
Inst_1:
Inst_1:
Inst_1: cdr list repl

CURRENTLY DEFINED REPLICATES
-----
Inst_1:
Inst_1:
```




cdr migrate server RI constraint enhancements

`cdr migrate server` enhancements

- Informix v.14.10.xC1 introduced the `cdr migrate server` command
 - It configures and starts ER to support heterogeneous instance migration
- Since it's ER, there is additional granularity available to what gets replicated
 - It is at the database level
 - Not all instance databases must be copied as `ifxclone` does
 - A new option in .xC3 allows you to exclude a named list of tables from the database
- This utility supports operations from Informix v.11.7.xC1 and later
 - You can migrate from older versions to the latest more easily with less downtime
 - It does NOT support code-set conversion during data transfer (eg. `en_us.819` to `utf8`)

`cdr migrate server` enhancements

- `cdr migrate server` operates in a “phased” approach
 - There are multiple phases, each executing a specific unit of functionality
 - For example defining and starting ER, adding ER key columns if needed to tables, creating the ER related spaces if they don’t already exist, creating the target database and migrating the data
 - Each “phase” can be executed individually either in non-execute mode or in execute mode
 - This way, you can look at the commands that phase will execute before it happens

`cdr migrate server` enhancements

- You can execute the command in two modes — offline or online
 - `static` mode, or offline means the source instance is blocked and the data is migrated as though it was a backup and recovery operation
 - There is no need to worry about referential constraint violations
 - `dynamic` mode or online migration means the source instance continues to process transactions during the migration process
 - There is a potential for RI violations to occur during the data migration phase

cdr migrate server enhancements

- The actual data loading process *for xC3 and earlier* has several steps to it
 - Step 1:
 - Migrate and create the database schema on the target
 - Indexes, PKs and constraints (unique or referential [FK]) are NOT created
 - The tables are created in `raw` mode
 - Step 2:
 - Start multiple, parallel jobs for data loading and index creation
 - Uses the Informix v.14.10 `insert into .. select * from ..` distributed SQL functionality
 - When the table is loaded, indexes, PKs and unique constraints are built
 - Step 3:
 - Referential constraints (FKs) are created and verified
 - Once the data is loaded, a data sync phase is executed to verify the data sets
- Because of the parallel load process, it's possible for RI constraints to fail causing the migration process to fail

`cdr migrate server` enhancements

- With `.xC4`, the `create_schema_loaddata_nori` clause has been added to stop RI building in `dynamic` migrations
- The `add_ri` phase has been added to rebuild RI after the `data_sync` phase of `dynamic` migrations
- Together, these clauses help resolve data sync and RI problems that arise from multiple parallel loads occurring from an active instance



`cdr migrate server` enhancements

- With .xC4, there is a new phase to `cdr migrate server` — `add_replcheck`
- An optional phase, it is not automatically executed in a migration
 - If it is executed manually, it should be executed first on the source instance before any other phases
 - By the way — if the target is empty / new, there's no point to executing this since all data must be copied to the target
 - That said, the `replcheck` column can be useful to speed up the data verification task if you decide to keep ER active after the data migration process
- This phase creates a shadow `ifx_replcheck` column (`bigint`) and a composite unique index on all source, and, by extension, target tables
 - WARNING!!! Adding the column is a slow alter requiring an outage

`cdr migrate server` **enhancements**

- The value in the `ifx_replcheck` column varies depending on whether the row is a new insert / pre-existing at command execution or modified after command execution
 - New inserts or pre-existing — the column contains a checksum of the row's values
 - Modified rows — the column contains a version related value
 - Both values incorporate the value for the `cdr` group ID
 - As defined in `SQLHOSTS`
- The `add_replcheck` index incorporates the `ifx_replcheck` shadow column and the `PK / UI / ERKey` columns
 - There are no options to control how this index is built (dbspaces, fragmentation, etc.)
 - However, with the “no execute” and print options, you can get the `add_replchck` phase commands to be executed, modify the index creation operations and execute the script manually
- Why do this? It dramatically increases the performance of the `sync_data` phase

cdr migrate server enhancements

- Let's look at more detail
 - In the `my_test` database are two tables from the `stores` database, `customer_empty` and `customer_full`
 - As the names suggest, one has data loaded in it, the other doesn't
 - There is a PK on each table

```
create table if not exists "informix".customer_empty
(
    customer_num serial not null ,
    fname char(15),
    lname char(15),
    company char(20),
    address1 char(20),
    address2 char(20),
    city char(15),
    state char(2),
    zipcode char(5),
    phone char(18)
) extent size 16 next size 16 lock mode row;

set no collation;

alter table "informix".customer_empty add constraint primary key (customer_num) ;
revoke all on "informix".customer_empty from "public" as "informix";

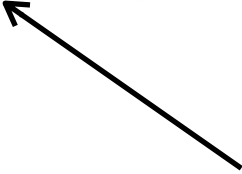
create index "informix".zip_ix on "informix".customer_empty (zipcode)
    using btree in data_space_1;
```

cdr migrate server **enhancements**

- I want to migrate this database from inst_1 to inst_6
 - But want to apply the add_replcheck phase so that is the first command executed

```
Inst_1: cdr migrate server -s inst_1 -t inst_6 -p add_replcheck -d my_test --exec
The version of the server you are using has full ERKEY support.
#--
#-- Creating replcheck column and index
#--
dbaccess - - <<EOF
database my_test@inst_1;
alter table 'informix'.customer_empty add REPLCHECK;
create unique index ifx_mig_replcheck_customer_empty_1 on informix.customer_empty(customer_num, ifx_replcheck);
alter table 'informix'.customer_full add REPLCHECK;
create unique index ifx_mig_replcheck_customer_full_2 on informix.customer_full(customer_num, ifx_replcheck);
EOF

Inst 1:
```



cdr migrate server

- After the command, check the table schema
 - There are two new objects including the replcheck index and the shadow column in the table

```
create table "informix".customer_empty
(
    customer_num serial not null ,
    fname char(15),
    lname char(15),
    company char(20),
    address1 char(20),
    address2 char(20),
    city char(15),
    state char(2),
    zipcode char(5),
    phone char(18)
) with replcheck extent size 16 next size 16 lock mode row;
set no collation;

alter table "informix".customer_empty add constraint primary
    key (customer_num) ;
revoke all on "informix".customer_empty from "public" as "informix";

create unique index "informix".ifx_mig_replcheck_customer_empty_1
    on "informix".customer_empty (customer_num,ifx_replcheck)
    using btree in data_space_1;

create index "informix".zip_ix on "informix".customer_empty (zipcode)
    using btree in data_space_1;
```

cdr migrate server enhance

- Obviously the index command displays what the new column name is but if you're interested, you can get the column and data type information from the new schema manager IHQ functionality

Select Database: my_test

Logged in as: ihqadmin [Change User](#)

Info SQL Editor

my_test > customer_empty

Table Details: [Information](#) [Columns](#) [Indexes](#) [References](#) [Privileges](#) [Statistics](#) [Constraints](#) [Fragments](#) [Triggers](#)

my_test > customer_empty

Columns

Search for name or data type...

Name	Data Type	Constraints	Visible
customer_num	SERIAL	n106_5: NOT NULL u106_6: PRIMARY KEY	✓
fname	CHAR(15)	-	✓
lname	CHAR(15)	-	✓
company	CHAR(20)	-	✓
address1	CHAR(20)	-	✓
address2	CHAR(20)	-	✓
city	CHAR(15)	-	✓
state	CHAR(2)	-	✓
zipcode	CHAR(5)	-	✓
phone	CHAR(18)	-	✓
ifx_replcheck	BIGINT	-	✗

Previous 1 Next

Rows per page: 20

cdr migrate server **enhancements**

- Looking at the contents of the shadow column
 - Obviously there is nothing in the “empty” table
 - The “full” table has checksum values
 - They are not user friendly 😊

```
----- my_test@inst_1 ----- Press CTRL-W for Help
```

customer_num	fname	lname	ifx_replcheck
101	Ludwig	Pauli	2513719298
102	Carole	Sadler	461185234
103	Philip	Currie	2674091415
104	Anthony	Higgins	321225238
105	Raymond	Vector	2884984374
106	George	Watson	3182389198

cdr migrate server enhancements

- A single row is inserted into the “empty” table
 - Its replcheck value contains versioning information
 - It's not user friendly either 😊

```
----- my_test@inst_1 ----- Press CTRL-W for Help
customer_num fname          lname          ifx_replcheck
          101 Ludwig          Pauli          6842431481533956097
```

- The row is updated with new column values
 - The version numbering changes as expected

```
----- my_test@inst_1 ----- Press CTRL-W for Help
customer_num fname          lname          ifx_replcheck
          101 Lewis          Paulson        6842432276102905858
```




Flow control delay statistics for RS and SD secondaries

Flow control statistics for SD and RS secondary instances

- In Informix v.11.50.xC6 (for `RSS_FLOW_CONTROL`) and Informix v.11.70 (for `SDS_FLOW_CONTROL`), flow control was introduced to help these instances stay current with the cluster primary
- It was possible for the RS / SD secondaries to fall behind in applying / recognizing committed transactions
 - Either because of slow network transmission, insufficient power on the secondary to handle the workload, or the transaction volume
 - The further behind these nodes became, the less useful they were for supporting connected sessions or being a potential failover target
 - If the secondaries fell too far behind, it could cause a logical log rollover condition on the primary resulting in suspension of instance activities until the secondary instance(s) caught back up
- The flow control parameter provided a way to throttle primary processing to help the secondary nodes to keep up

Flow control statistics for SD and RS secondary instances


- With the improved roll forward technology introduced at the end of Informix v.12.10 (for HDR) and Informix v.14.10.xC1 (for RS secondaries), you probably won't need flow control any more
 - The apply rate is insanely quick!
- In Informix 14.10.xC4 more flow control statistical information is available in the `onstat -g [rss | sds] verbose` output
 - If applications are experiencing processing delays, this information can help diagnose whether or not flow control is the issue

Flow control statistics for SD and RS

- For example, in a quiet test environment with one SD and RS secondary instance
 - From the primary looking at the SD secondary

```
Number of SDS servers:1
Updater node alias name :inst_1

SDS server control block: 0x468ead50
server name: inst_2
server type: SDS
server status: Active
connection status: Connected
Last log page sent(log id,page):41,120
Last log page flushed(log id,page):41,120
Last log page acked (log id, page):41,120
Last LSN acked (log id,pos):41,491544
Last log page applied(log id,page): 41,120
Approximate Log Page Backlog:0
Current SDS Cycle:26
Acked SDS Cycle:26
Sequence number of next buffer to send: 3391
Sequence number of last buffer acked: 3390
Time of last ack:2020/06/26 14:28:24
Supports Proxy Writes: N
Time of last received message: 2020/06/26 14:28:57
Time of last alternate write: N/A
Time of last alternate read : N/A
Total number of delay(s): 14
Time of last delay: 2020/06/26 13:42:12
```



Flow control statistics for SD and RS secondary


- For example, in a quiet test environment with one SD and RS secondary instance
 - From the primary looking at the RS secondary

```
Local server type: Primary
Index page logging status: Enabled
Index page logging was enabled at: 2020/06/24 10:28:08
Number of RSS servers: 2

RSS Server information:

RSS Server control block: (nil)
RSS server name: inst_5
RSS server status: Defined
RSS connection status: Disconnected

RSS Server control block: 0x46f78d20
RSS server name: inst_4
RSS server status: Active
RSS connection status: Connected
RSS flow control:384/352
Log transmission status: Active
Next log page to send(log id,page): 41,126
Last log page acked(log id,page): 41,125
Last log page applied(log id,page): 41,125
Time of Last Acknowledgement: 2020-06-26.14:33:01
Pending Log Pages to be ACKed: 0
Approximate Log Page Backlog:0
Sequence number of next buffer to send: 443
Sequence number of last buffer acked: 442
Supports Proxy Writes: Y
Total number of delay(s): 14
Time of last delay: 2020-06-26.13:42:12
```



Questions

Your registration confirmation had information
on where the replay and slides are located

