Informix Enterprise Replication

Nagaraju Inturi
nagaraju.inturi@hcl.com

Why replicate data?

- **Data dissemination**
- **Update anywhere**
- **Workload partitioning**
- **High availability**
- **Disaster recovery**
- **Rolling upgrades with near-zero downtime for business**
- **Multi-node administration**
- **Database codeset migration with zero downtime**

Enterprise Replication basic functionality (1)

- **Log-based replication**
- **Rows identified by:**
  - **Primary key**
  - **Enterprise Replication key**
  - **Unique index**
- **Supports selective replication**
  - **Row selectivity**
  - **Column selectivity**
- **Supports various replication strategies**
  - **Primary/Secondary**
  - **Dissemination**
  - **Consolidation**
  - **Update-Anywhere**

Enterprise Replication basic functionality (2)

- **Supports various rules for conflict resolution**
  - **Timestamp**
  - **Ignore**
  - **Stored procedure**
  - **Always apply**
  - **Deletewins**
- **Supports various topologies**
  - **Fully connected**
  - **Hub and spokes**
  - **Hierarchy**
  - **Forest of trees**

What is new in Enterprise Replication?

- List ER commands from syscdr database

usage: cdr list catalog [-c server] [-s] [-r] [-e] [-t] [-z] [-g] [-q] [-a]

-c server --connect=server connect to server

-s --server print define server commands

-r --replicate print define replicate commands

-e --replicateset print define replicateset commands

-t --template print define template commands

-z --realizetemplate print realize template commands

-g --grid print define grid commands

-a --all print all commands

Example :

cdr list catalog --server

```
#
# cdr define server commands.
#
cdr define server --ats=/work/tmp --ris=/work/tmp --init utm_group_1
cdr define server --connect=utm_group_2 --ats=/work/tmp --ris=/work/tmp --sync=utm_group_1 --nonroot --init utm_group_2
cdr define server --connect=utm_group_3 --ats=/worktmp --ris=/worktmp --sync=utm_group_2 --nonroot --init utm_group_3
cdr define server --connect=utm_group_4 --ats=/worktmp --ris=/work/tmp --sync=utm_group_3 --nonroot --init utm_group_4
```

What is new in Enterprise Replication?

- **Performance improvements**
  - **Optimized Send/Receive queue logic to reduce locking contention**
  - **Huge improvement in performance on a heavily loaded system with 100+ servers**
    - **Internal testing shows that data apply rate increased from 10K rows/minute to 2 million rows/minute!**
  - **Made improvements to memory allocation to reduce locking contention**
    - **Memory allocation pools are associated with specific CPU virtual processors. Enterprise Replication allocates memory to the CPU virtual processors based on which CPU virtual processor the cdr thread is executing on.**
    - **Memory allocation pools are associated with specific block sizes, so that all allocations from a pool are the same size, and the first free block that is found can be used.**
- **Time-series data replication**

RoadMap

12.10 – Timeseries, Sharding, Performance improvements

11.70 – Flexible Grid, Cloning ER node using ifxclone

11.50 – cdr sync/check task enhancements, XML file format support for ATS/RIS files, OAT support for ER

11.10 - rename/truncate table support, online config change support

10.00 - Templates, Schema Evolution, Sync/Resync/Check

9.40 - ER/HDR, Large Transaction, Encryption Support

9.30 - UDT support, Parallel Apply, Queue Rewrite

7.31 - Complex Topology / Routing

7.22 - Initial Release

ER Setup Requirements

- Row must contain a primary key, ER Key or Unique index
- Table must be logged
- Instance must contain a smart blob
  - The smart blob is used for temporary storage of the replicated if the transaction is not able to be applied on all of the targets within a timely manner
- Sqlhost file must be modified to contain a ‘group’ entry
  - Each group contains a ‘cdrid’ as an i=## option

Terms used with Enterprise Replication

- Node
  - An Informix instance within the replication domain
- Replicate
  - An object that defines which table to be replicated and where to replicate.
- Participant
  - The node on which a replicate is being replicated to/from (i.e. source/target)

N.B. A replicate can have multiple participants

Terms used with Enterprise Replication

- Mastered replicate
  - ER maintains a dictionary defining the transport format
  - Provides better support for replicate consistency
  - Required for Schema Evolution Support

- Replicate set
  - Group of replicates. Easy to administer
- Exclusive replicate set
  - To group related replicates to avoid referential constraint failures
- Templates
  - Provides a means for deployment of many tables as a unit.
  - Reduces the Effort to setup Replication.
  - Creates a Replicate Set.

Server Definitions

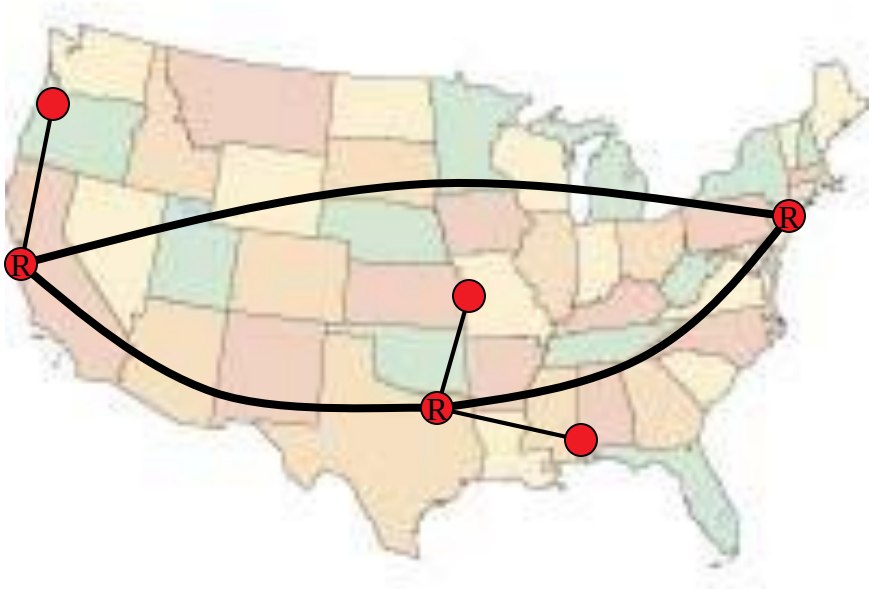
Roots

```
cdr define serv –c Oakland_g  –I  Oakland_g
cdr define serv –c NewYork_g  –I  -S Oakland_g NewYork_g
cdr define serv –c Dallas_g  –I  -S NewYork_g Dallas_g
```

Server Definitions

Roots

Non-Roots



cdr define serv –c Portland_g –I **-N** -S Oakland_g Portland_g

cdr define serv –c Lenexa_g –I **-N** -S Dallas_g Lenexa_g

cdr define serv –c Mobile_g –I **-N** -S Dallas_g Mobile_g

Server Definitions

Roots

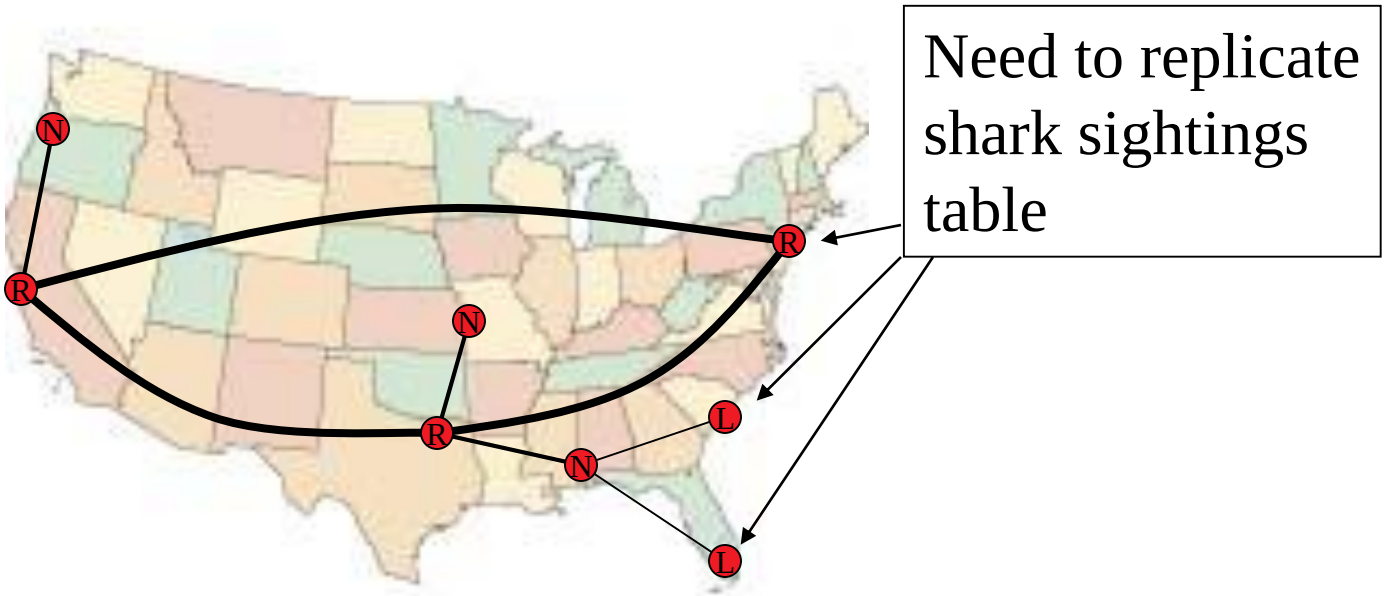
Non-Roots

Leafs

cdr define serv –c Charleston_g –I **-L** -S Mobile_g Charleston_g

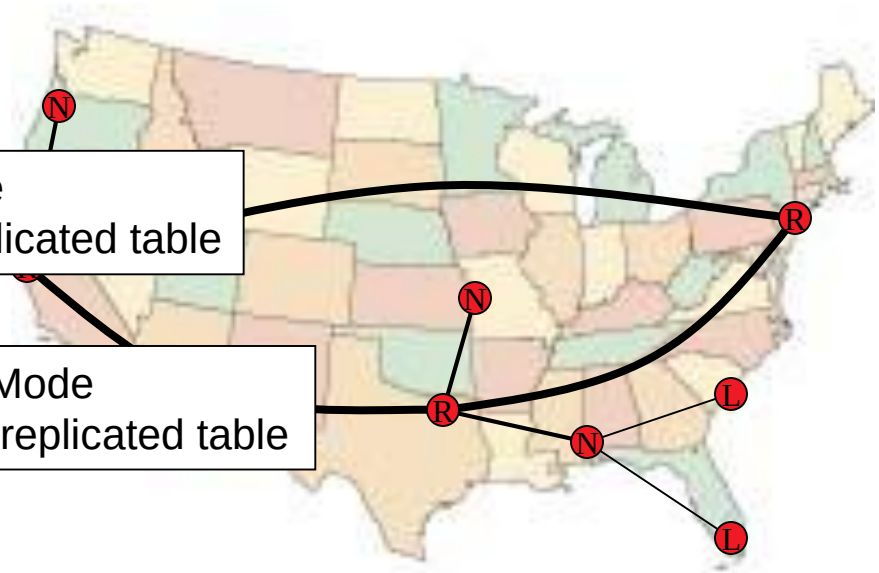
cdr define serv –c Miami_g –I **-L** -S Mobile_g Miami_g

Replicate Definitions (Update Anywhere)




```
cdr define replicate –c Dallas_g –S row –C timestamp SharkRepl \
“DB@Miami_g:userid.Sharks” “select * from Sharks” \
“DB@Charleston_g:userid.Sharks” “select * from Sharks” \
“DB@NewYork_g:userid.Sharks” “select * from Sharks”
cdr start replicate –c Dallas_g SharkRepl
```

Replicate Definitions (Primary/Target)



Primary Mode
Source of replicated table

Receive Mode
Target of replicated table

```
cdr define replicate –c Dallas_g –S row –C ignore StockQuotes \
“P DB@NewYork:userid.StockQ” “select * from StockQ” \
“R DB@Lenexa_g:userid.StockQ” “select * from StockQ” \
“R DB@Oakland_g:userid.StockQ” “select * from StockQ”
cdr start replicate –c Dallas_g StockQ
```

Conflict Resolution

- The rules used to determine if the local row should remain or if the replicated row should be applied
- Required for update anywhere

Ignore	Row must be applied using original operation	No upsets
Always	Replicated row will be applied	Upsert support
Timestamp	Row with most recent activity wins	Upsert support
Deletewins	Delete operation always wins the conflict. For other operations most recent activity wins the conflict.	No upserts
Stored Procedure	Stored Procedure invoked to determine winner	Upsert support
Timestamp with Stored Procedure	Stored Procedure invoked to break a timestamp tie	Upsert support

ATS and RIS files

- ATS
  - Complete transaction is spooled
  - One ATS file per transaction
  - ATS file created when complete transaction is aborted
  - Only replicated rows are spooled in this file. Local rows are not spooled.
  - Shows RIS file name if RIS file created for this transaction
  - XML file format supported
  - ATS file can be repaired using ‘cdr repair ats’ command
    - XML format not supported with ‘cdr repair ats’
- RIS
  - Only aborted rows are spooled
  - One RIS file per transaction
  - Both local and replicated rows are spooled for the aborted rows.
  - Shows ATS file name if ATS file was also created for this transaction
  - XML file format supported
  - RIS file can be repaired using ‘cdr repair ris’ command
    - XML format not supported with ‘cdr repair ris’ command

Templates

- Provides a means for deployment of many tables as a unit
  - Simplifies Administration
  - Reduces the Effort to setup Replication
- Number of commands required to setup replication between two ER nodes for 300 tables:
  - Without templates: 600 commands
  - With templates: 2 commands!
- Two step process
  - Define the template
    - Sets the tables within the template – can be entire database
    - Generates mastered replicates for the members of the template
  - Realize the template
    - Creates all of the participants on the node

Few important ER configuration parameters

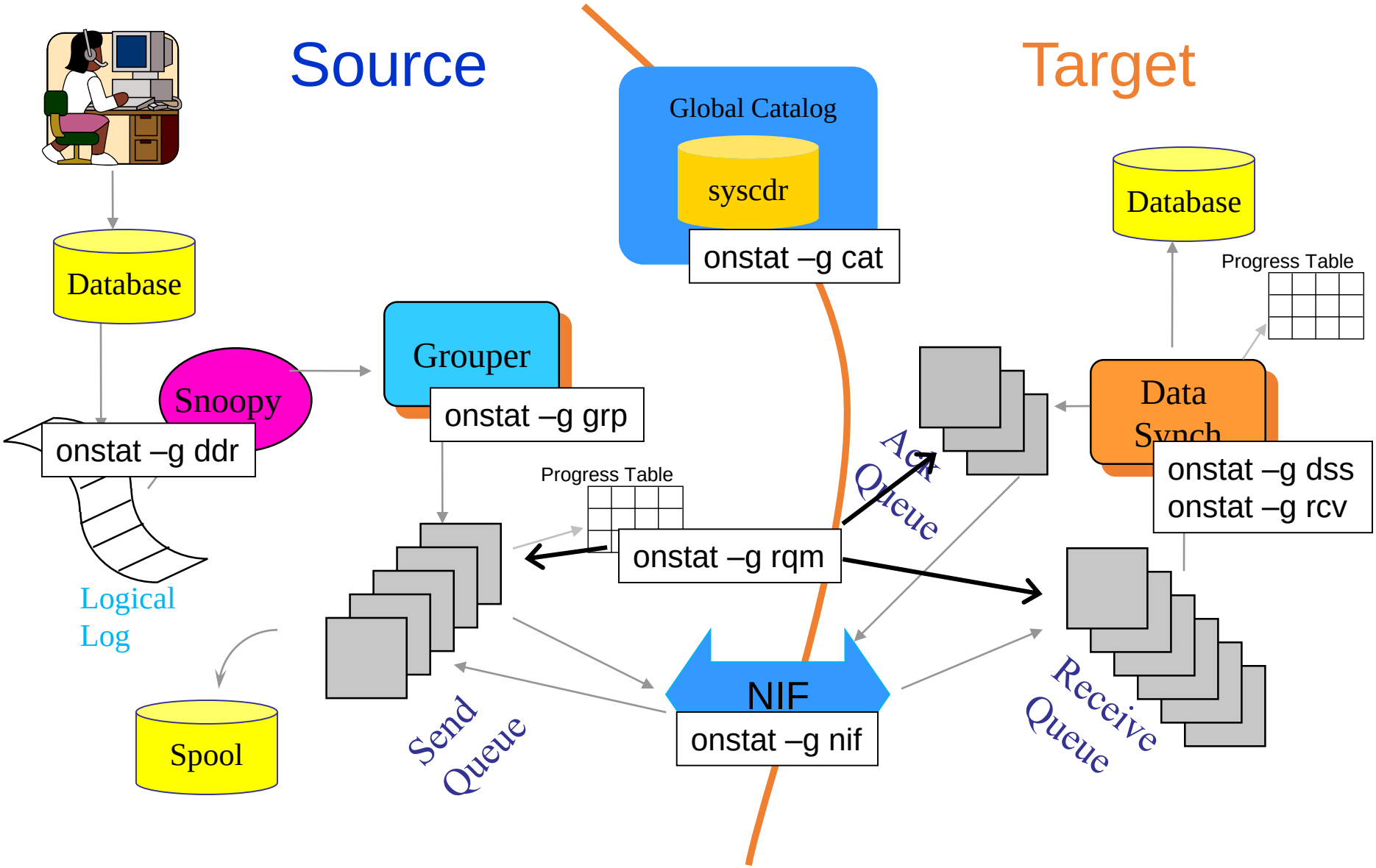
CDR_QUEUEMEM	Queue memory size. Default value=4MB. Recommended value= 30% of logical log size or 10 percent of SHMVIRTSIZE
CDR_QDATA_SBSPACE	List of queue smartblob space names separated with comma.Recommended to configure one logging and one non-logging sbspace.
CDR_DBSPACE	Syscdr database location. Defaults to rootdbs.
CDR_QHDR_DBSPACE	Dbspace name for spooling transaction headers in ER queues.
CDR_SERIAL	Needed only if serial column being used as primary key.

Few important ER configuration parameters

CDR_APPLY	Specify minimum and maximum number of datasync threads. Default value: A range of one to four datasync threads for each CPU VP
CDR_LOG_LAG_ACTION (11.70)	Specifies how Enterprise Replication responds to a potential log wrap situation. •logstage •dlog •ddrblock •ignore •shutdown

Note: You can change most of the ER configuration parameters while server is on-line using ‘cdr change config’ command.

ER – how it works



ER alarms to monitor

Class id	Class message	Specific message	Explanation
31	1. ER stable storage queue dbspace is full 2. ER stable storage queue sbspace is full 3. ER stable storage pagersbspace is full	1. CDR QUEUER: Send Queue space is FULL - waiting for space in CDR_QHDR_DBSPACE. 2. CDR QUEUER: Send Queue space is FULL - waiting for space in *sbspace_name*. 3. CDR QUEUER: Send Queue space is FULL - waiting for space in CDR_QDATA_SBSPACE 4. CDR Pager: Paging File full: Waiting for additional space in *sbspace_name*	This event runs in the foreground.The storage space of a queue has filled. **ER state:** Active and waiting for space to be added to the dbspace/sbspace **User action:** Add a new chunk to the dbspace/sbspace specified in specific message

ER alarms to monitor

Class id	Class message	Specific message	Explanation
47	• CDR is shutting down due to internal error: failure	1. CDR is shutting down due to an internal error. 2. CDR is shutting down due to internal error: Memory allocation failed	•Enterprise Replication has stopped. •**ER state:** Inactive. •**User Action:** To resume replication, solve the issue described in specific message, and run the **cdr start** command or shut down and restart the database server.

Note: 47 can be triggered if snoopy log position was overwritten too. In this situation if you try restarting ER then restart fails and server raises alarm 75.

ER alarms to monitor

Class id	Class message	Specific message	Explanation
75	• ER: the logical log replay position is not valid. Restart ER with the **cdr cleanstart** command, and then synchronize the data with the **cdr check –repair** command.	• The replay position (logical log ID *log_number* and log position *log_position*) has been overwritten. • The replay position (logical log ID *log_number* and log position *log_position*) is later than the current position.	**ER State:** Replication is stopped on this server. **User Action:** Clear the receive queue and restart replication by running the **cdr cleanstart** command and then synchronize the data by running the **cdr check replicateset** command with the **--repair** option.

ER alarms to monitor

Class id	Class message	Specific message	Explanation
48	• ATS and/or RIS files spooled to disk	*file name|file name*	One or more failed transactions caused the generation of one or more ATS or RIS files. The generated file names are listed in the specific message, separated with a pipe (|) character. **ER State:** Replication is continuing normally. **User Action:** To process the failed transactions, run the **cdr repair ats/ris** command for each file, or run the **cdr check replicateset/repl** command with the **--repair** option.

ER alarms to monitor

Class id	Class message	Specific message	Explanation
30	• DDR subsystem failure	• WARNING: The replay position was overrun, data may not be replicated. • DDR Log Snooping - Catchup phase started, userthreads blocked • DDR Log Snooping - Catchup phase completed, userthreads unblocked	Server ran into DDRBLOCK situation. **ER state:** Active and replicating data. Enterprise Replication shuts down if the log read position also gets overwritten. If Enterprise Replication shuts down, event alarm 47 is raised. **User action:** Consider expanding logical log and CDR_QUEMEM configuration if you frequently see this alarm.

CDR GC errors in the message log file

05:35:42 CDR GC peer processing failed: command: define repl, error 40, CDR server 2

05:35:43 CDR GC peer processing failed: command: start repl, error 31, CDR server 2

cdr error

SERVER:SEQNO	REVIEW	TIME	ERROR
site2:6	N	2001-04-26 03:54:46	40

GC operation define replicate 'rep1' failed: unsupported SQL select clause syntax

cdr finderr 40

40 unsupported SQL syntax (join, etc..)

cdr finderr 31

31 undefined replicate

Are any servers suspended or dropped?

```
onstat -g nif
 Id    Name              State                  Sent   Received
-----------------------------------------------------------------
   9   site4             RUN                     9     15440      6031

cdr list server
SERVER              ID STATE    STATUS     CONNECTION CHANGED
-----------------------------------------------------------
site1                6 Active   Local            0
site2                7 Suspend  Dropped          0 Jun 11 14:38:40
site3                8 Suspend  Dropped          0 Jun 11 14:38:37
site4                9 Active   Connected        0 Jun 11 14:36:50
```

Check the replicate state

```
cdr list repl

REPLICATE:      Repl1
STATE:          Inactive ON:g_cdr_ol_nag_1
CONFLICT:       Ignore
FREQUENCY:      immediate
QUEUE SIZE:     0
PARTICIPANT:    bank:nagaraju.teller
OPTIONS:        transaction,ris,ats,fullrow
REPLID:         655361 / 0xa0001
REPLMODE:       PRIMARY  ON:g_cdr_ol_nag_1
APPLY-AS:       INFORMIX ON:g_cdr_ol_nag_1
REPLTYPE:       Master
```

‘onstat –g cat repl’

```
REPLICATES
-------------------
  Parsed statements:
        Id 655361 table teller

  Inuse databases: bank(1)
    Name: Repl1a_1, Id: 655361 State: ACTIVE Flags: 0   use 0 lastexec Wed Dec 31 18:00:00 1969

      Local Participant: bank:nagaraju.teller
      Attributes: TXN scope, Enable ATS, Enable RIS, all columns sent in updates
      Conflict resolution[Prim::Sec]: [IGNORE]
      Column Mapping: ON, columns INORDER, offset 8, uncomp_len 327
      No Replicated UDT Columns
```

Make sure replicate state is active in cache as well as in disk

Log Position (onstat –g ddr)

```
DDR -- Running --

# Event   Snoopy    Snoopy    Replay    Replay    Current   Current
Buffers    ID       Position   ID       Position    ID      Position
528       5         134080    5         129018    5         135000

Log Pages Snooped:
      From       From       Tossed
     Cache       Disk   (LBC full)
         7        302            0

Total dynamic log requests: 0

DDR events queue

Type    TX id     Partnum   Row id
```

Want to see fairly close together.

Onstat –g grp T (Transaction summary in grouper)

IBM Informix Dynamic Server Version 11.50.FC4 -- On-Line -- Up 03:37:57 -- 260096 Kbytes

```
Last Tx to Queuer began : (1236041566) 2009/03/02 18:52:46
Last Tx to Queuer ended : (1236041566) 2009/03/02 18:52:46
Last Tx to Queuer log ID, position: 125,594396
Open   Tx: 0
Serial Tx: 0
Tx not sent: 61
Tx sent to Queuer: 526
Tx returned from Queuer: 526
Events sent to Queuer: 1
Events returned from Queuer: 1
Total rows sent to Queuer: 526
Open Tx array size: 1024
```

Begin and commit time for last txn queued

Commit work log position for last txn queued

Total rows queued so far

Could be due to txn rollbacks

onstat –g grp pager

IBM Informix Dynamic Server Version 10.00.UC6 -- On-Line -- Up 05:23:46 -- 36864 Kbytes

Grouper Pager statistics:

Number of active big transactions: 1

Total number of big transactions processed: 2

Spool size of the biggest transaction processed: 314572800 Bytes

TIP: ‘onstat -g cdr config long’ shows grouper paging threshold for large transaction (in bytes)

Send Queue (onstat –g rqm sendq)

```
CDR Reliable Queue Manager (RQM) Statistics:

RQM Statistics for Queue (0x0x4579f018) trg_send
 Transaction Spool Name: trg_send_stxn
 Insert Stamp: 31/0
 Flags: SEND_Q, SPOOLED, PROGRESS_TABLE, NEED_ACK
 Txns in queue:              13
 Log Events in queue:        0
 Txns in memory:             13
 Txns in spool only:         0
 Txns spooled:               0
 Unspooled bytes:            10920
 Size of Data in queue:      10920 Bytes
 Real memory in use:         10920 Bytes
 Pending Txn Buffers:        0
 Pending Txn Data:           0 Bytes
 Max Real memory data used: 10920 (4194304) Bytes
 Max Real memory hdrs used  10556 (4194304) Bytes
 Total data queued:          18480 Bytes
 Total Txns queued:          31
 Total Txns spooled:         0
 Total Txns restored:        0
 Total Txns recovered:       0
 Spool Rows read:            0
 Total Txns deleted:         18
 Total Txns duplicated:      0
 Total Txn Lookups:          38
```

Current Statistics

Summary

onstat –g rqm sendq

Progress Table Portion

Peer Node

ReplicateID

What's in the Queue

What has been ACKED

How far have we put into the queue

```
Server     Group Bytes Queued      Acked                          Sent
-----------------------------------------------------------------------------
     6  0x10009                  0 1/9/138ad8/0              -            1/9/138ad8/0
     5  0x10009                  0 1/9/138ad8/0              -            1/9/138ad8/0
     4  0x10009                  0 1/9/138ad8/0              -            1/9/138ad8/0
     3  0x10009                  0 1/9/138ad8/0              -            1/9/138ad8/0
     2  0x10009               4154 effffffff/effffffff/effffffff/effffffff - 1/9/138ad8/0
     6  0x10006                  0 1/9/12d8f8/0              -            1/9/12d8f8/0
     5  0x10006                  0 1/9/12d8f8/0              -            1/9/12d8f8/0
     4  0x10006                  0 1/9/12d8f8/0              -            1/9/12d8f8/0
     3  0x10006                  0 1/9/12d8f8/0              -            1/9/12d8f8/0
     2  0x10006              31625 effffffff/effffffff/effffffff/effffffff - 1/9/12d8f8/0
```

Transaction key = {CDRID, Txn commit log id, log pos., Sequence number}

Onstat –g rqm sendq

Is first transaction changing ?

```
First Txn (0x0D60C018) Key: 1/9/0x000d4bb0/0x00000000
 Txn Stamp: 1/0, Reference Count: 0.
 Txn Flags: Notify
 Txn Commit Time: (1094670993) 2004/09/08 14:16:33
 Txn Size in Queue: 5908
 First Buf's (0x0D31C9E8) Queue Flags: Resident
 First Buf's Buffer Flags: TRG, Stream
 NeedAck: Waiting for Acks from <[0004]>
 No open handles on txn.
```

BIT-MAP indicating who needs to send an ACK

From onstat –g cat

```
Id: 02, Nm: cdr2, Or: 0x0004, off: 0, idle: 0, state Suspended

   root Id: 00, forward Id: 02, ishub: FALSE, isleaf: FALSE
```

!!?

Apply (onstat –g rcv full)

```
Receive Manager global  block 0D452018
        cdrRM_inst_ct:                          5
        cdrRM_State:                    00000000
        cdrRM_numSleepers:              3
        cdrRM_DsCreated:                3
        cdrRM_MinDSThreads:             1
        cdrRM_MaxDSThreads:             4
        cdrRM_DSBlock                   0
        cdrRM_DSParallelPL              0
        cdrRM_DSrateRate                0.000000
        cdrRM_DSNumRun:                 35
        cdrRM_DSNumLockTimeout          0
        cdrRM_DSNumLockRB               0
        cdrRM_DSNumDeadLocks            0
        cdrRM_ACKwaiting                0
        cdrRM_totSleep:                 77
        cdrRM_Sleeptime:                153
        cdrRM_Workload:                 0
        cdrRM_optscale:                 4
        cdrRM_MinFloatThreads:          2
        cdrRM_MaxFloatThreads:          7
        cdrRM_AckThreadCount:           2
        cdrRM_AckWaiters:               2
        cdrRM_AckCreateStamp:           Wed Sep 08 11:47:49 2004
        cdrRM_DSCreateStamp:            Wed Sep 08 14:16:35 2004
        cdrRM_acksInList                0
        cdrRM_BlobErrorButs:            0
```

Current Level of Apply Parallelism
0 – greatest parallelism

Indication of the number of collisions between various apply threads

ACKs that have been received, but not yet processes

onstat –g rcv full

Total Summary

By Source

```
Receive Parallelism Statistics
Server Tot.Txn. Pending Active MaxPnd MaxAct  AvgPnd  AvgAct CommitRt
     1      35       0      0     21      3    7.00    1.63    0.00
     5       3       0      0      1      1    1.00    1.00    0.02
     6       6       0      0      1      1    1.00    1.00    0.21

Tot Pending:0     Tot Active:0  Avg Pending:5.77    Avg Active:1.50
Commit Rate:0.01

Time Spent In RM Parallel Pipeline Levels
Lev. TimeInSec  Pcnt.
   0     17405 100.00%
   1         0   0.00%
   2         0   0.00%
```

Parallelism summary

onstat –g rcv full

Number of transactions applied from sources

Most recently applied transaction

```
Statistics by Source

Server 1
Repl        Txn      Ins      Del      Upd   Last Target Apply   Last Source Commit
65541        23        0        1      616   2004/09/08 14:20:15 2004/09/08 14:20:15
65542        11        0        0      253   2004/09/08 14:19:33 2004/09/08 14:19:33
65545         1        0       67        0   2004/09/08 14:20:37 2004/09/08 14:20:37

Server 5
Repl        Txn      Ins      Del      Upd   Last Target Apply   Last Source Commit
65541         3        0        0       81   2004/09/08 16:36:10 2004/09/08 16:36:09

Server 6
Repl        Txn      Ins      Del      Upd   Last Target Apply   Last Source Commit
65548         6        0        0       42   2004/09/08 16:37:59 2004/09/08 16:37:58
```

Inserts/Updates/Deletes within applied transactions

Source Server by ReplicateID
(ReplicateID from syscdr:repdef or onstat –g cat)

Onstat –g nif <CDRID>

```
NIF anchor Block: c00000000b520ef0
    nifGState          RUN
      RetryTimeout      300

Detailed Site Instance Block: c00000000b715a68
    siteId             151
    siteState          256 <RUN>
    siteVersion         8
    siteCompress         0
    siteNote           0
    Send Thread         332 <CDRNsT151>
    Recv Thread         335 <CDRNr151>
    Connection Start     (1141923879) 2006/03/09 17:04:39
    Last Send           (1142001841) 2006/03/10 14:44:01
    Idle Timeout         <Forever>
    Flowblock            Sent 1 Receive 0
    NifInfo: c00000000b4dde30
        Last Updated    (1141923879) 2006/03/09 17:04:39
        State          Connected
        Total sent      4366 (145844 bytes)
        Total recv'd    37165 (3325467 bytes)
        Retry          0 (0 attempts)
        Connected       1
```

Nif flow blocking

‘cdr view’ command

- Prints ER statistics for all active ER nodes
- Connects to all ER servers and queries ER pseudo tables in sysmaster database
- Few important options
  - Profile
  - ddr
  - state
  - servers
  - nif
  - rcv
  - apply

Sysmaster views/tables

- Syscdrs and syscdrserver -- ER node/server related information
- syscdrerror – Error table information
- syscdrreplset – Replicate set related information
- syscdrrepl – Replicate related information
- syscdrpart – Participant related information
- syscdrqueued – Bytes queued information for the given server and replicate
- syscdrtxproc – Datasync transaction related statistics for the given server
- syscdrlatency – Replicate latency related statistics for the given source server and replicate (onstat –g rcv full)
- syscdr_rqm – RQM statistics (onstat –g rqm)
- syscdr_ddr – Log reader related information (onstat –g ddr, cdr view ddr)
- syscdr_state – ER, Capture, Network, and Apply state information (cdr view state)
- syscdr_nif – Network information (onstat –g nif, cdr view nif)
- syscdr_rcv – Receive manager information (onstat –g rcv, cdr view rcv)

Data synchronization

Initial synchronization (cdr sync)

- Replicates complete table from source node to target node(s)
- Performs table scan and queues data into sendq at source server
  - Uses log-bypass technique
- Ideal for setting up replication on new ER node.
- Use --memadjust option to tune send/recv queue memory during sync task execution

Resynchronization (cdr check)

- Only mismatched and missing rows are replicated from source server.
- Performs index scan and computes checksum to find mismatches between source and target server.
- Performs dummy update to replicate mismatched and missing rows from source server
  - This can cause increased logging activity at source server
- Ideal for situations where one of the ER node is down for long time.

Enhancements to ‘cdr sync/check’ task

Sync and Check Progress indicator

- Problem
  - How does one check progress of a sync/check task while the task is still running?
- Solution:
  - Sync/check task statistics information is stored in two new tab
    - replcheck_stat : Global sync/check task progress information and s task.
    - replcheck_stat_node: Sync/check task progress and statistics infor node.
  - Sync/check task now prints timestamp information at the beginning and end of sync/check table scan process. Sample output:

```
Jan 17 2009 15:46:45 ------   Table scan for repl1 start  --------
Jan 17 2009 15:46:55 ------   Table scan for repl1 end   ---------
```

Sync and Check Progress indicator - examples

- To get sync/check task progress information and statistics, run cdr sync or cdr check with --**name=<task name>** option. Example:

```
$cdr check replset --master=g_serv1 --replset=rset1  --name=test
g_serv2 --repair
```

- Sync/check task statistics can be queried using command cdr stats check <task name> for check task, and cdr stats sync <task name> for sync task. Example:

```
$cdr stats check test
$cdr stats sync test
```

- Once task is completed, statistics can be deleted using the command cdr stats check –delete <task name> command. Example:

```
$cdr stats check --delete test
```

Progress indicator sample output

```
>cdr stat check --repeat=2 test
        Job test
        repl1                     Started Jan 17 16:10:59
        *********+----+----+----+----+----+----+----+----+ Remaining 0:00:08
        ---------------------------------------------------
        Job test
        repl1                     Started Jan 17 16:10:59
        **********************--+----+----+----+----+----+ Remaining 0:00:04
Sample output with verbose option:
>cdr stat check --repeat=4 --verbose test
        Job test
        repl1                     Started Jan 17 16:34:42
        *******--+----+----+----+----+----+----+----+----+ Remaining 0:00:12
        Node                  Total     Extra   Missing  Mismatch     Child Processed
        ---------------- --------- --------- --------- --------- --------- ---------
        cdr1                   9000         0         0         0         0         0
        cdr2                   9000         0         0        99         0        99
        cdr3                   9000         0         0         0         0         0
```

Consistency Checking Performance Improvement using new hidden ifx_replcheck column

- Why this feature?
  - Customers are using the automatic check and repair features of ER more and more.
  - Customers require better ‘cdr check’ performance.
- How does this feature help?
  - The existing consistency check requires reading the data rows and computing a checksum. Customers using this feature on large tables will see significantly better performance with ‘cdr check’ than without this feature.

Consistency Checking Performance Improvement

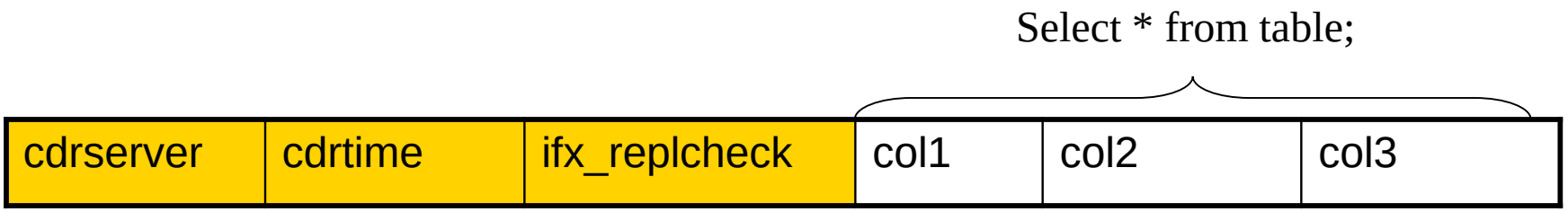
- This feature improves **cdr check** performance by:
  - the addition of new shadow column *ifx_replcheck*
    -and-
  - the creation of a composite index on this primary key and this shadow column on the replicated table.
- ER will check the replicated table for the existence of the *ifx_replcheck* column, and the index that uses it.
- If found, **cdr check** will use this index to perform key-only scan which greatly improves the performance.
- The index on the primary key and the *ifx_replcheck* shadow column need to exist on all participant nodes. If all nodes do not contain this column and index, the feature will not work on those nodes.

Consistency Checking Performance Improvement

- To enable feature for a new table:
  - Create the table with the ‘replcheck’ and ‘crcols’ columns
    - `create table … with replcheck with crcols;`
  - Create a unique index using existing primary key columns + the *ifx_replcheck* column
    - `CREATE UNIQUE INDEX rocket_idx ON rocket (rocket_id, rocket_name, ifx_replcheck);`
- To enable feature for an existing table :
  - Alter the table using the ADD REPLCHECK clause.
    - `ALTER TABLE tablename ADD REPLCHECK;`
      `Note: Adding replcheck hidden column is a slow alter`
  - If necessary, alter the table using the ADD CRCOLS clause.
    - `ALTER TABLE tablename ADD CRCOLS;`
  - Create a unique index based on the existing primary key columns and the ifx_replcheck column.
    - `CREATE UNIQUE INDEX indexname ON tablename (PKcolumns…, ifx_replcheck);`

ifx_replcheck shadow column

- Bigint(8 byte) column. 4 bytes for timestamp + 4 bytes for row version
- Index creation allowed on this column
- Primary purpose is to improve consistency check (cdr check) task performance
- Create table XYZ (……) with CRCOLS with replcheck;



Consistency Checking Performance Improvement

- ifx_replcheck column is a hidden column -- not a shadow column like CRCOLS
- It does appear in DB–Access when you ask for information about the columns of the table.
- It is included in the number of columns (**ncols**) in the **systables** system catalog table entry for *tablename*.

Parallel Checks

- Allows to run consistency check on multiple replicates in para
  - Long form: --process=#
  - Short form: -p #
- New option allows user to specify how many tasks to run in parallel.
  - If you use p=4 - one master process is started and 4 child processes could run at one time, for a total of 5 processes.
  - Not all replicates can be processed in parallel.
- Only works with Replicate Sets, not with individual replicates

Parallel Checks - Example

- The following command starts two parallel processes to generate consistency reports for the replicates in the **replset_1** replicate set, comparing the data on **g_er2** against the data on **g_er1**:

```
cdr check replicateset -m g_er1 -s replset_1 g_er2 -p 2
```

- The best practice when checking Replicate Sets is to use a value of at least: `-p 2`

In-flight transactions during check

- cdr check process might report a row as inconsistent If there is ‘in flight’ data (data already in the replication stream, or values modified after the row has already been checked
- With --inprogress=<seconds> option, you can specify maximum wait time for checking the inconsistent rows.
- cdr check process will recheck the inconsistent rows till the maximum timeout value specified with --inprogress option.
- Default value for --inprogress option is 5 seconds.
- Example:
  - `cdr check replset --master g_serv1 --replset rset1 g_serv2 --inprogress=30`

User Supplied Where Clause

- Customer Problem:
  - Customers who need to check tables containing large amounts of data, especially if the table is a partitioned table, can face performance problems with the check process. For example, if one of the partitions of a replicated table was destroyed, the user might want to resynchronize it after performing a warm recovery of that partition. He could do that by running a repair check and using the where clause used to define that partition.
- Solution:
  - New option for **cdr check replicate** -where=<user supplied where clause>.
  - Enables the user to specify a subset of the table to be included in the check task.
  - The user supplied where clause should be a single string and should not include the word ‘where’.
- Restrictions:
  - only available with **cdr check replicate**
  - not for replicate sets, due to where clause column name
- Examples:
  - `cdr check replicate -r sales_replicate -master=g_node1 -repair --where="region = 'east'" g_node2`

Since Time Support

- Customer Problem:
  - Production systems can contain vast data repositories, the ability to only check rows recently changed could significantly reduce the time needed to complete a check of the entire table.
- Solution:
  - New option --since= (-S) with **cdr check replicate** and **cdr check replicateset** to only check rows inserted/updated since a time value.
  - Only valid if the replicate is defined with *timestamp* or *delete-wins* conflict resolution.
- Example:
  - Check all of the replicates within the stores_set replicate set which were updated within the past 3 days. Repair any inconsistency which is found.
  - `cdr check replset -set=stores_set -master=g_node1 -all -repair --since=3d`

Skip Large Objects

- Customer Problem:
  - Customers need a better method to check tables that contain Large Objects. Often these large objects tend to be static (not updated frequently), thus they could be excluded from row comparisons.

- Solution:
  - Provide the ability to ignore the comparisons of all large objects (BYTE, BLOB, TEXT, CLOB, etc.).
  - Adding the -skipLOB (-L) option to a check will ignore these datatypes.

Background Tasks

- Customer Problem:
  - Running **cdr sync** and **cdr check** from the command line is not always the desired method. The ability to submit a job and then check its status is preferred.
- Solution:
  - New background option:
    - Short Form: -B
    - Long Form: --background
  - Background Tasks are registered with the scheduler via the Admin API. Control is returned immediately to the command line, and the job is scheduled to run now.
  - It is recommended that background tasks are submitted using the --**name= (-n)** option.

Replication without a Primary Key

- New Shadow Column which can be used in place of a primary key

  create table (…) with ERKEY
  alter table xyz add ERKEY

  create replicate –erkey

- Internally used by Flexible Grid to replicate tables without a primary key

Informix Flexible Grid

Informix Flexible Grid - Enhanced Availability and Workload Management

Benefits and Highlights
Create a grid with 2 servers or 1000s of servers Mix hardware, operating systems, and versions of Informix in the grid Centralized, simultaneous administration for all servers in the grid Balance workload across the grid Eliminate planned downtime

Easily install and setup

Administer your servers from a single location

Create a solution using mixed hardware & software

Easily replicate all or part of your database objects or data

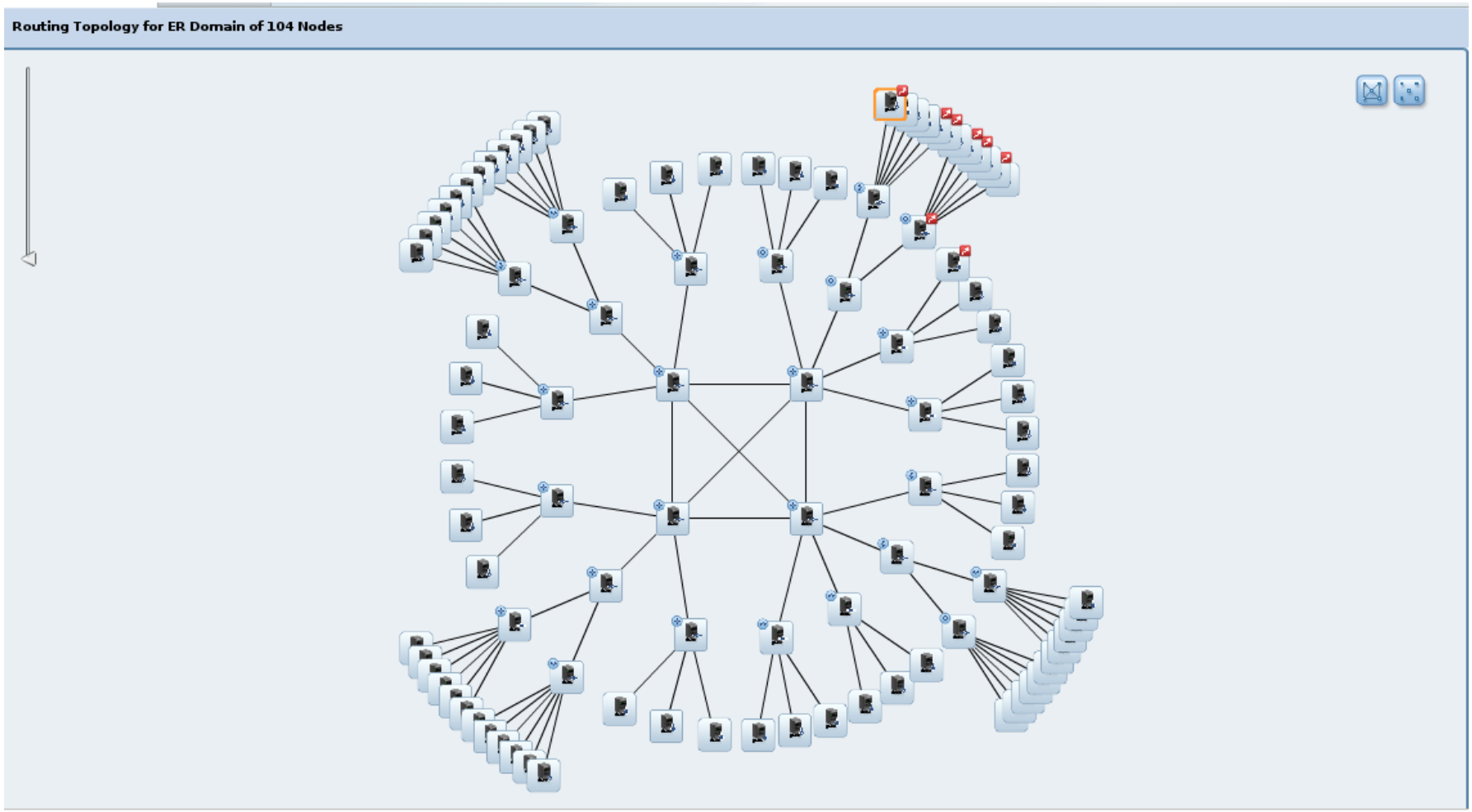
Add or drop nodes in the grid online

Actively balance workloads with Connection Manager

Use rolling upgrades to eliminate downtime

Easily clone an instance to create new servers

How scalable is the Informix Flexible Grid?



Flexible Grid Use Cases

Flexible Grid Use Case 1

- Sue has an ER domain of 20 nodes.
- She needs to purge 4 million rows from a replicated table.
- She could rely on basic ER functionality to do this, but she is concerned about the impact to the network that such an operation would have.

Grid Based Statement Execution

- An individual statement can be executed as a grid statement as well as a stored procedure/function

```
execute procedure ifx_grid_execute('grid1',
 'delete from orders where order_year < 2009');
```

  - The execution is replicated, not the results of the execution
  - The table 'orders' could be a raw table, or even contained within a non-logging database.
  - By default, the results would not be replicated by ER

Flexible Grid Use Case 2

- Jim needs to add 3 rows to a table “job_class" on each of the servers that he is responsible for.
- This table is not replicated and contains job classification for some new position that his company is creating.
- There are several hundred servers that this needs to be done on.
- He’s concerned that it will take forever to complete this task.

Grid Based Procedure Execution

In addition to DDL propagation, we can perform the execution of a procedure, function, or statement as a grid operation.

Create a new procedure across the grid

```
execute ifx_grid_connect('mygrid');
create procedure newjobclass()
insert into job_class values ("class1",…);
insert into job_class values ("class2",…);
insert into job_class values ("class3",…);
end procedure;
execute ifx_grid_disconnect();
```

Execute the new procedure across the grid

```
execute procedure
ifx_gird_procedure("mygrid","newjobclass()");
```

Flexible Grid Use Case 3

- Sam has been asked to add a new table on the databases that he is responsible for.
- Because of the potential size of the table, he wants to isolate it in a dbspace.
- The only problem is that he needs to first create that dbspace on each of the 500 nodes that he is responsible for.

Grid Based Function Execution

- The only difference between a function and a procedure is that a function will have a return value.
  - The return is saved in the syscdr database and can be viewed from

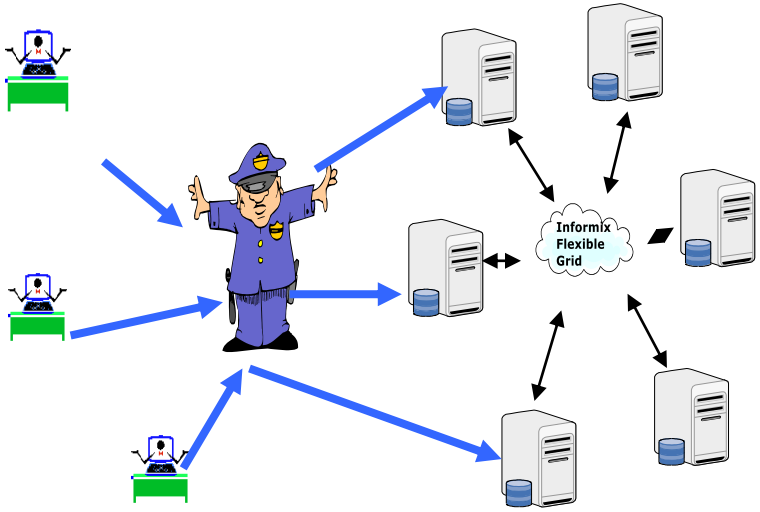
```
cdr list grid

database sysadmin;
execute function ifx_grid_function('grid1',
    'task("create dbspace","dbsp3",
"/db/chk/chk3","8G","0")');
```

  - The above command would create a new 8GB dbspace “dbsp3” on all nodes within the grid.
  - By default the results of the function execution would not be replicated by ER

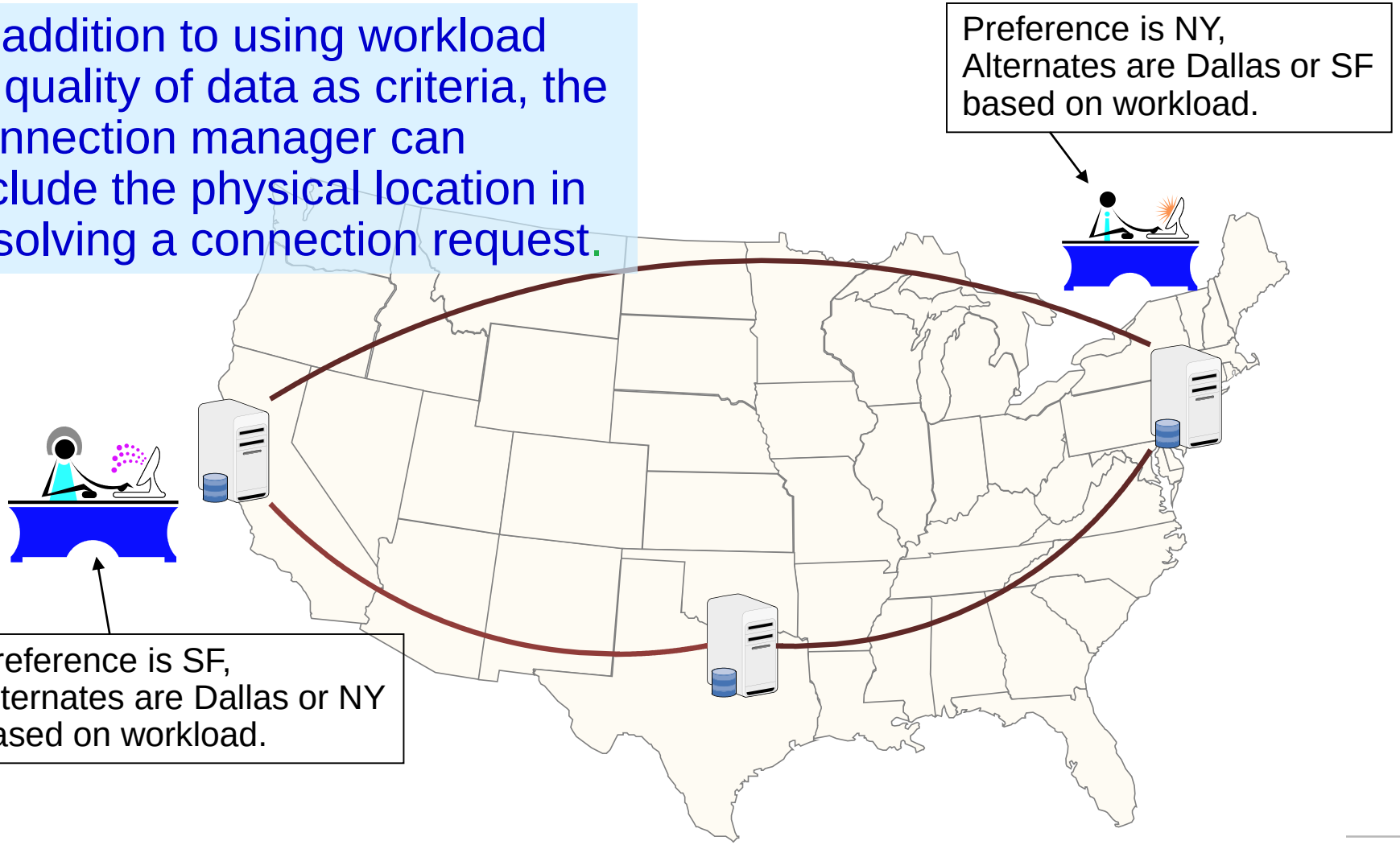
The Connection Manager with the Grid

- Implemented in 11.50 for the MACH11 Cluster
- Expanded to support ER/GRID in 11.70
- Takes location as well as workload into consideration when defining a Service Level Agreement (SLA)

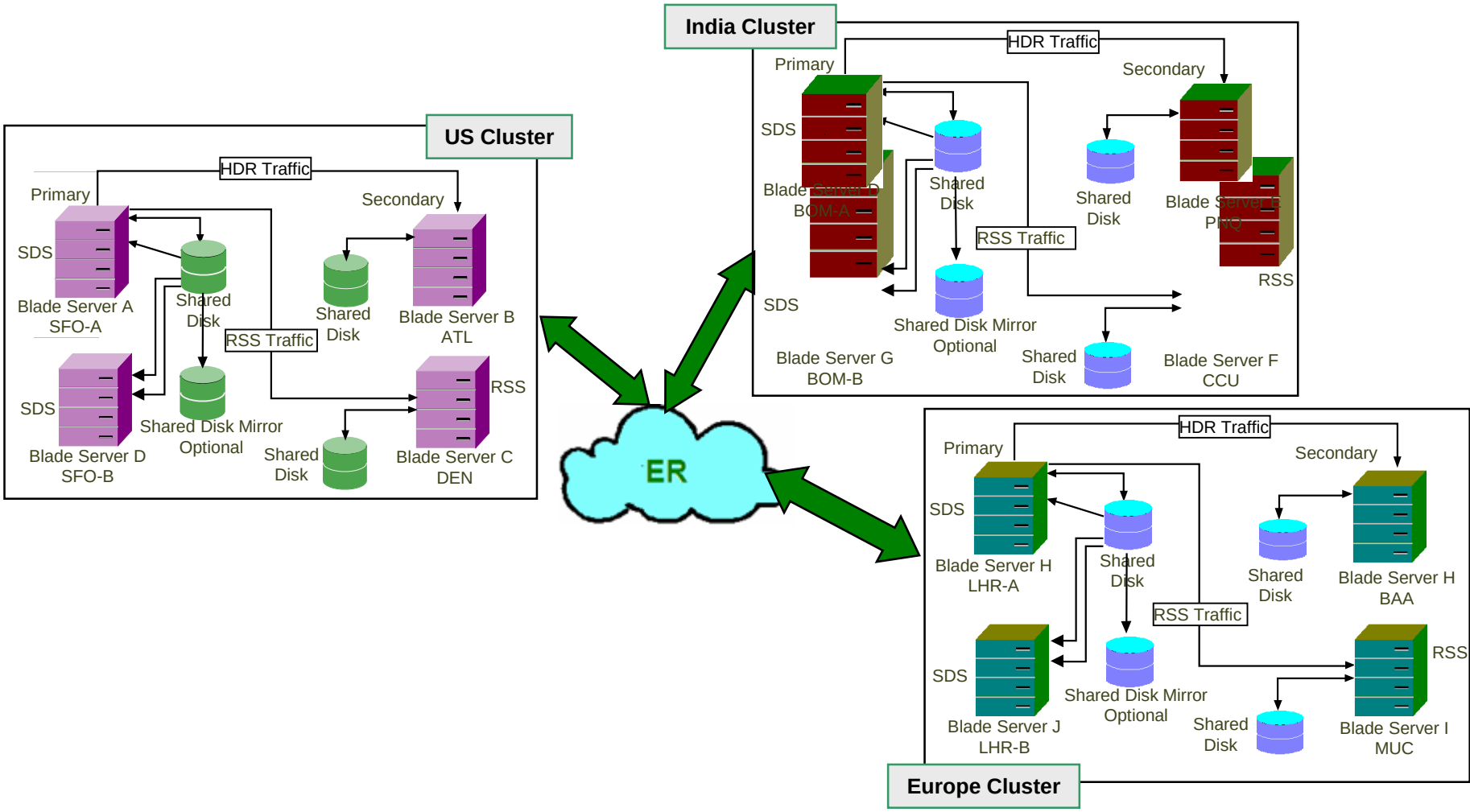


Preference by Location as well as Load Balance

In addition to using workload or quality of data as criteria, the connection manager can include the physical location in resolving a connection request.

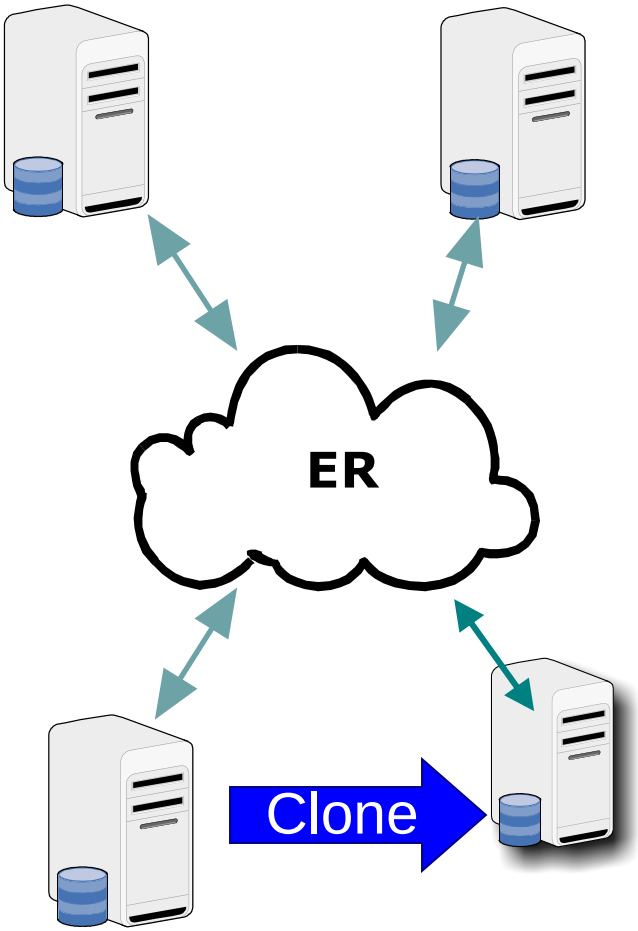


ER co-existence with High-Availability Cluster



ifxclone

- Allows a snapshot copy of a system
- Utilizes backup/recovery logic
- Final clone can be an independent server, an RSS node, or an ER clone of the source node
- Requires that source allows the clone to be taken



Enterprise Replication Best Practices

Enterprise Replication best practices (1)

- Make sure system clocks are in sync for replication servers
- Configure ER queue memory(CDR_QUEMEM) to be at least 10 percent of SHMVIRTSIZE
- Configure logical log files such that log wrap do not happen within a 4-hour interval
- Avoid frequent spooling activity in sendq
- If possible avoid replicating large transactions (500+ rows)
- Configure at least two sbspaces for CDR_QDATA_SBSPACE -- one with logging and one without logging

Enterprise Replication best practices (2)

- Always create replicated tables with row locking. Page locking tables may reduce apply performance.
- If you are replicating buffered logging database, consider changing it to unbuffered logging database
- Monitor ER events using an alarm program
- Recommended values:
  - LTXHWM 40
  - LTXEHWM 70
- Execute all related ER admin commands –like ‘define repl’ and ‘start repl’-- from the same server
- Check status of ER admin commands by running ‘cdr error’ command
- If you want to improve ‘cdr check’ task performance consider adding ‘ifx_replcheck’ shadow column to replicated table, and create composite index on primary key columns and ifx_replcheck column.

Further references

- Knowledge center
- Informix community Blog
- Informix Replication redbook
- IDS experts blog

Questions?

Thank You!

Informix

Flexible Grid