CDC Based Dump & Load

Migrating Bothersome Tables

Tom Bascom, White Star Software

Abstract: The “long pole” in many dump & load plans often boils down to a small number of very large tables. Leveraging OpenEdge’s Change Data Capture feature provides a way to migrate these tables without a long outage.

CDC Based Dump & Load

Migrating Bothersome Tables

Tom Bascom, White Star Software
tom@wss.com

protop

White Star software

What is Change Data Capture?

- A way to record Creates, Updates and Deletes so that you can export only the changed data.
- Similar to after-imaging or auditing – but user controllable!
- The target can be anything, it does not need to be an OpenEdge db.
- Unlike trigger based replication (old Pro2) both 4GL and SQL changes are captured.
- A huge improvement over “all or nothing” ETL processes!

CDC Challenges!

- You decide what tables and fields are important.
- You decide when to act on the changes.
- You decide how to export the data.
- You decide how to purge the change records.

- The provided admin tools are “arcane”:
  - OE Explorer: browser based GUI.
  - OEE does have a “generate code” option.
  - OO “helper api” (sort of scriptable).
- Documentation is “sparse” and incomplete.

Understanding CDC Policy “Levels”

1. Minimal: just records the fact that a change occurred, cannot have any identifying fields, no change table is created. Only a timestamp and a rowid are recorded.
2. MinimalWithBitmap: records which fields were changed but not their values.
3. Medium: also records the new values of changed fields.
4. Maximum: adds "before image", this is necessary if fields in unique indexes can be updated.

Change Tables

- If the policy is >= 2 then a “change table” is created.
- The default new table name has a “cdc_” prefix, IOW:
  cdc_customer
- This table has 7 additional “_” fields and 3 extra indexes along with whatever fields are being explicitly tracked.
- The extra fields and indexes support finding the proper changes, in the proper order, to process.
- Base fields might be reordered! (“Identifying fields” will be moved to the front if necessary.)

The Field Map

- _Change-fieldmap is a raw bitmap.
- System fields ("_" prefix) are not part of the bitmap.
- Bits in _Change-fieldmap correspond to _field-rpos:
  - EXCEPT that they are off by 8!
  - Conveniently, the System fields start at 2 and there are 7 of them.
  - So bit #1 maps to _field-rpos 9.
  - byte# + bit# = _field-rpos ☺

Handy Bitmap Debugging Code

```
/* convert a raw bitmap into a string for logging / debugging purposes
 */

function bitmap2string returns character ( bitMap as raw ):
  define variable stringBM as character no-undo.
  define variable i        as integer   no-undo.
  define variable j        as integer   no-undo.
  do i = 1 to length( bitMap ):                                          // byte by byte
    do j = 1 to 8:                                                       // bit by bit
      stringBM = stringBM + string( get-bits( get-byte( bitMap, i ), j, 1 ), "9" ).   // "0" or "1"
    end.
    stringBM = stringBM + " ".                        // separate bytes with a space for easier reading
  end.
  return trim( stringBM ).
end.
```

Large Records

- Records near the 32,000 byte limit are challenging.
- The CDC change record adds 7 fields – so if the base record is near 32,000 bytes it won't fit!
- To solve this, large change records can be "split" or "fragmented" into multiple records.
- The _continuation-position field indicates where fragments start and stop (parsing this is "messy")

Finding Changes to Process

```
for each _cdc-Change-Tracking exclusive-lock
   where _cdc-Change-Tracking._Source-Table-Number = srcTblNum
     and _cdc-Change-Tracking._Change-Sequence > startAfter:

  for each cdc_Customer exclusive-lock
     where cdc_customer._change-sequence = _cdc-Change-Tracking._Change-Sequence
     break by cdc_customer._change-sequence by cdc_customer._operation:

    /* do stuff! */

  end.

end.
```

Some Code Highlights…

Goal

- Migrate troublesome tables from HPUX to Linux with a minimal amount of required downtime (less than 1 hour)

Assumptions

- A small number of very large tables are in scope.
- All relevant tables have at least one unique index.
- The vendor will use existing methods to dump & load “normal” tables.
- This process will be executed on many diverse end user systems.
- The specific tables may vary from customer to customer – thus a need for flexibility.

Outline

CDC Downtime	HPUX - Prod	HPUX - Clone	Linux - Destination
	Install CDC licenses	Allocate Space for copy of DB	Build skeleton db on Linux host
Start CDC Init	*Shutdown Production*		
	Clone DB (PR determines technique)		
cdc_enable.sh	*Enable Source db for CDC*		
	Add CDC policies for target tables		
	Adjust startup parameters: -cdcsize, -prefetch -S etc*		
15 minutes	*Restart Production, CDC is now tracking changes to target tables*	Start dumping target tables (std PR)	Start load of dumped baseline data
		Dump of target tables finishes	
		Clone DB no longer needed	Finish loading baseline data
			Rebuild indexes
cdc_synch.sh			*Start CDC synchronization*
cdc_monitor.sh			*CDC Sync is "close enough"*
cdc_sync.sh	*Stop Production*		
15 minutes			*Finish CDC sync*
	Start "normal" dump		Start "normal" load
	using standard PR D&L process		Finish load
	Dump finishes		Rebuild indexes for loaded tables (NOT for CDC tables)
			Verify new db
	Old Production DB no longer needed		**Restart Production on Linux!**

protop

White Star software

Prepare

- Install CDC Licenses on HPUX Source
- Prepare “Clone” (pick a method, allocate space)
  - Replication target
  - After-image roll-forward target
  - Restored backup
  - Disk snapshot
- Create “skeleton db” on Destination
  - Empty structure (no CDC areas needed)
  - Load .df file (no CDC tables)
  - Pre-grow data extents (optional, requires a full dry run)

Destination Database

- This is an opportunity to improve your storage area configuration!
- Review table, index and LOB assignments:
  - Verify rows per block decisions
  - Check for excessive fragmentation in the source database
  - Check for long “RM Chains” in the source database
- Ensure that CDC target tables and indexes are not mixed with non-CDC tables and indexes:
  - Since these are typically the largest objects in the db they should *already* have dedicated storage areas
  - Regardless of that they need dedicated areas to avoid unwanted index rebuilds at the conclusion of the “normal” dump & load

Clone Source Database

- Shutdown Production
- Clone the Source DB
  - Use a replication target
    - Remove replication target from replication
    - Or backup replication target and restore to another location
  - Use a restore and roll-forward strategy
  - Or restore a backup taken at shutdown
  - Disk snapshot
- The Clone reflects the state of the Source immediately prior to enabling CDC
- CDC will record changes in the Source
- No need to enable CDC on Clone or Destination
- CDC does not use the Clone directly!
- The CDC destination obtains its baseline from the clone.

Enable CDC, Add Policies

- No need to enable CDC on Clone or Destination
- Enable Source:

```
hpux#  prostrct [addonline] s2k add_cdc.st
hpux#  proutil s2k -C enablecdc area CDC_Track_Data indexarea CDC_Track_Idx
```

- Add CDC Policies to Source

```
hpux#  pro s2k –p cdc_policy_new.p –param Customer,OrderLine,Salesrep
```

- Adjust Source Startup Parameters

  -Mm 32600 -prefetchDelay -prefetchFactor 100 -prefetchNumRecs 5000

  -S port#

  No need to adjust –cdcsize, the default of 2000 is more than large enough

cdc_enable.sh

- Adds storage areas (defined in cdc_add.st)
- Enables Change Data Capture
- Creates policies

```
hpux#  cdc_enable.sh s2k Customer OrderLine Salesrep
```

Restart Production, Start Dump, Start Load

- These steps are all standard vendor provided processes
- Restart Production
  - CDC will record changes to the specified tables
- Start dumping the CDC tables from Clone
  - This is the baseline data that changes will be applied to
  - Use standard, well known dump & load techniques (i.e. binary d&l)
- Start the load process on the Destination
  - Use standard, well known dump & load techniques (i.e. binary d&l)

Finish Baseline Load

- Load all of the baseline data from the Clone
- Rebuild indexes on Destination
- Verify by comparing tabanalys between Clone and Destination

Start CDC Synchronization Process

- Baseline Data is Loaded on Destination
- CDC Change Data is queued on Source
- Clone is no longer needed
- Pull Change Data from Source to Destination:

```
linux#  export TBL1=Salesrep
linux#  _progres -b -db s2k -ld src –H sourceHost –S sourcePort \
                 -db s2k_new -ld dst -p ./cdc_sync_table.p \
                 -param "${TBL1},3,no" -debugalert –errorstack \
                 -clientlog cdc_sync_${TBL1}.debug > cdc_sync_${TBL1}.err 2>&1 &
linux#  tail –f cdc_sync_${TBL1}.log
```

- sync_table.p should run on the Destination (Linux) server

cdc_sync.sh

- Runs on the Destination (Linux) server
- Starts one process per CDC enabled table
- Must not start until the baseline data has been loaded

```
linux#  cdc_sync.sh s2k s2k_new Customer OrderLine Salesrep
```

cdc_monitor.sh

- Runs on the Source (HPUX) server
- Monitors the queue of CDC changes
- Helps to predict when you will be “close enough”

```
hpux#  cdc_monitor.sh s2k
```

Final Migration

- Shutdown Production on HPUX
- Dump non-CDC Source Tables using standard techniques
- Load non-CDC Tables on Destination
- Rebuild Indexes for non-CDC Tables
- Restart CDC Synchronization
  - There should only be a modest number of “catch-up” records queued
- Verify All Record counts with proutil –C tabanlys

Go Live!

- Restart Production on Linux

Demo!

Appendix A
Programs, Scripts, and Files

Programs and Scripts

Program Name	Runs On	Purpose
cdc_env.sh	All	Common environment variables used by the scripts.
cdc_enable.sh	Source	Adds storage areas, enables CDC and creates table policies.
cdc_monitor.sh	Source	Monitors the state of the CDC queues.
cdc_policy_delete.p	Source	Remove unwanted CDC policies and change data associated with CDC tables!
cdc_policy_create.p	Source	Helper procedure to create CDC policies. Called by cdc_policy_new.p.
cdc_policy_new.p	Source	Creates policies for tables to be synchronized by sync_table.p.
cdc_qmon.p	Source	Monitors the queue of changes to be synchronized.
cdc_add.st	Source	Structure file containing definitions of CDC storage areas.
cdc_srv.pf	Source	Startup parameters specific to CDC source database.
cdc_sync.sh	Destination	Fetches changed data records from Source, applies them to the Destination.
cdc_sync_table.p	Destination	Does the actual work for cdc_sync.sh (one table per process).

Demo Programs and Scripts

Program Name	Runs On	Purpose
cdc_demo_init.sh	N/A	Initializes the demo environment
cdc_demo.sh	N/A	Sets up a demo run with sports2000. Useful for testing with a copy of sports2000.
cdc_demo_stop.sh	N/A	Stops the demo.
chg_srep.p	N/A	Batch process that makes lots of changes to the Salesrep table.
chg_oline.p	N/A	Batch process that changes, updates and deletes OrderLines.

sync_table.p control and output files

File	Purpose / Usage
sync_Salesrep.flg	flag: remove it and the process stops running `linux> rm –f sync_Salesrep.flg`
sync_Salesrep.log	status of the running process `linux> tail -f sync_Salesrep.log`
sync_Salesrep.dbg	debug level, integer, change it on the fly! `linux> echo 3 > sync_Salesrep.dbg`
sync_Salesrep.err	startup errors `linux> tail sync_Salesrep.err`
sync_Salesrep.debug	runtime errors & 4gl stack trace `linux> tail sync_Salesrep.debug`

sync_table.p -param Salesrep,3,yes,1234

- Start one sync_table.p process per CDC table
- Should run on the Destination (-S SourcePort)
- Parameters:

tableName	the name of the table being synchronized
debugLevel	1-5 level of debugging info desired
cdcPurge	yes/no – purge change records after applying them?
startSeq	what change sequence number should we start at? (useful if you are NOT purging and need to restart)

cdc_add.st

```
hpux> cat cdc_add.st

# cdc_add.st
#
# the area numbers are arbitrary, you can use whatever you would
# like but 990 and 991 are unlikely to already be in use
#
d "CDC_Track_Data":990,64;512 .
#
d "CDC_Track_Idx":991,1;64 .
```

cdc_srv.pf

```
hpux> cat cdc_srv.pf

# cdc_srv.pf
#

-Mm 32600                # maximum -Mm size = 32600
-prefetchDelay           # do not send the first record immediately
-prefetchFactor 100      # try to fill messages 100% full
-prefetchNumRecs 5000    # try to put 5000 records per message

# -cdcsize 2000          # default is 2000
#
# no need to increase unless a very large number of tables are enabled
#
# cdcsize = ((( numTbls * 96 ) + numFields * 32 )) * 1.3 ) / 1024
```

Monitor Synchronization Process

```
hpux>  cdc_monitor.sh s2k
```

CDC Cache

Cache Name	Cache Size	Total Entries	Total Hits	Total Misses
CDC Primary	1536000	3	5251	0
CDC Secondary	512000	0	0	0

CDC Latch Activity

_Latch-Name	_Latch-Lock	_Latch-Wait
MTL_CDC	0	0

CDC Change Queue Monitor

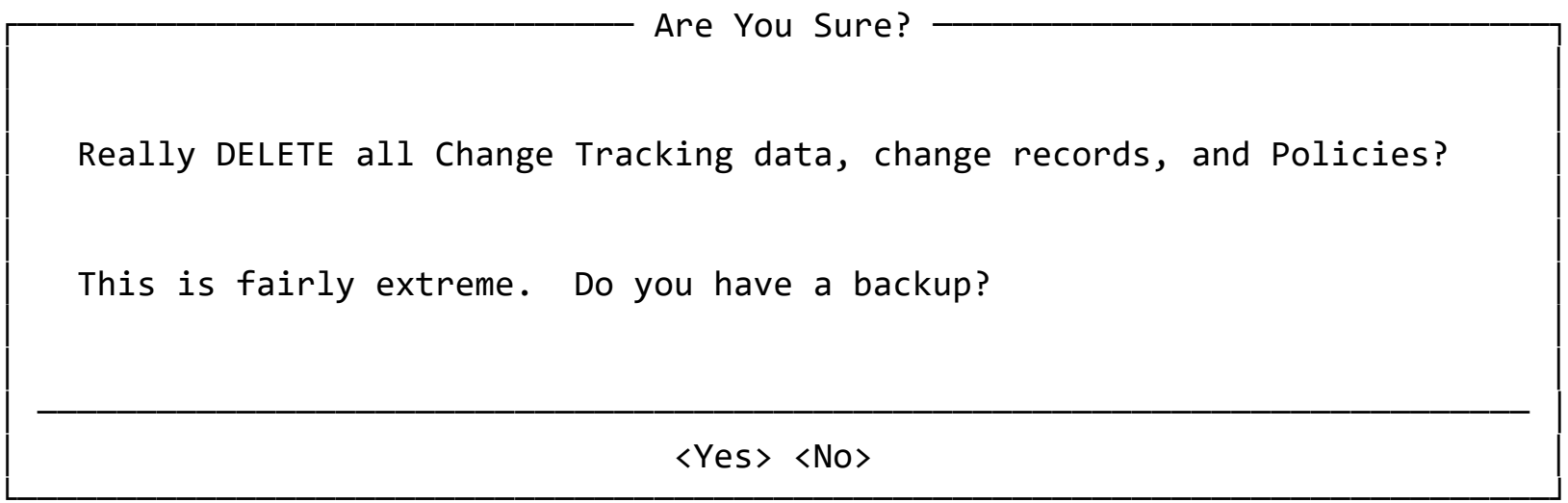
	first	last	queued
CDC_Customer	1	1	1
CDC_OrderLine	?	?	?
CDC_Salesrep	1	5,247	5,247

protop

White Star software

Delete CDC Policies

```
hpux>  _progress s2k -p cdc_policy_delete.p
```



sync_tblName.log

```
linux> tail -f sync_Salesrep.log
18/11/2019 14:57:15.620-05:00 synchronizing: Salesrep  src = s2k  dst = s2k_new
18/11/2019 14:57:15.620-05:00 dbgLevel = 3  cdcPurge = no  startSeq = 0
18/11/2019 14:57:15.621-05:00 fieldList =  SalesRep,RepName,Region,MonthQuota
18/11/2019 14:57:15.621-05:00 uniqIdx =  SalesRep
18/11/2019 14:57:15.621-05:00 _time-stamp, lagTime, _tran-id, _change-sequence, _operation, _continuation-position, _arrayIndex, _fragment, record-length, operText,
    tableName, WHERE clause, processing description

18/11/2019 14:57:15.722-05:00 dbgLevel = 3
18/11/2019 14:57:15.722-05:00 waiting

18/11/2019 15:01:28.239-05:00 18/11/2019 15:01:28.174-05:00 .065 27958 1 4 0 0 0 87 update Salesrep where SalesRep = "SLS" 2 fields changed: Region,MonthQuota
18/11/2019 15:01:28.245-05:00 18/11/2019 15:01:28.185-05:00 .06 27959 2 4 0 0 0 94 update Salesrep where SalesRep = "DOS" 2 fields changed: Region,MonthQuota
18/11/2019 15:01:28.246-05:00 18/11/2019 15:01:28.216-05:00 .029 27960 3 4 0 0 0 62 update Salesrep where SalesRep = "DOS" 1 fields changed: Region
18/11/2019 15:01:28.246-05:00 18/11/2019 15:01:28.247-05:00 .021 27964 4 4 0 0 0 84 update Salesrep where SalesRep = "DOS" 2 fields changed: Region,MonthQuota
18/11/2019 15:01:28.347-05:00 waiting
18/11/2019 15:01:28.348-05:00 18/11/2019 15:01:28.267-05:00 .081 27966 5 4 0 0 0 83 update Salesrep where SalesRep = "DOS" 2 fields changed: Region,MonthQuota
18/11/2019 15:01:28.348-05:00 18/11/2019 15:01:28.298-05:00 .05 27967 6 4 0 0 0 86 update Salesrep where SalesRep = "DOS" 2 fields changed: Region,MonthQuota
18/11/2019 15:01:28.449-05:00 waiting
. . .
18/11/2019 15:03:49.969-05:00 18/11/2019 15:03:49.899-05:00 .07 39175 5246 4 0 0 0 79 update Salesrep where SalesRep = "DOS" 2 fields changed: Region,MonthQuota
18/11/2019 15:03:49.970-05:00 18/11/2019 15:03:49.920-05:00 .05 39176 5247 4 0 0 0 79 update Salesrep where SalesRep = "HXM" 2 fields changed: Region,MonthQuota
18/11/2019 15:03:50.071-05:00 waiting
18/11/2019 15:54:03.332-05:00 waiting
18/11/2019 15:56:21.002-05:00 flag ./sync_Salesrep.flg removed; graceful shutdown with an empty Salesrep change queue, last change seq# = 5247
18/11/2019 15:56:21.003-05:00 done
```

Appendix B
Demo Scripts

cdc_demo_init.sh

```
hpux>  cdc_demo_init.sh

Really stop all running CDC demos, delete s2k and s2k_new databases and start over? y

Shutdown is executing. (1613)
Shutdown complete. (1614)
Procopy session begin for tom on batch. (451)
Database copied from /usr/dlc120/dlc/sports2000. (1365)
Procopy session end. (334)
Shutdown is executing. (1613)
Shutdown complete. (1614)

The CDC Demo Environment is initialized.

hpux>
```

cdc_enable.sh s2k Customer Orderline . . .

```
hpux>  cdc_enable.sh s2k Customer Orderline Salesrep
The structure file format is valid. (12619)
Device: /home/, KBytes needed: 2304, KBytes available: 182322257 (12616)
There is sufficient free space to initialize the defined extents. (12618)


Converting relative path database to absolute path database. (8461)

Formatting extents:
   size                area name   path name
    512           CDC_Track_Data /home/tom/src/s2k_990.d1 00:00:00
     64            CDC_Track_Idx /home/tom/src/s2k_991.d1 00:00:00
OpenEdge Release 12.0 as of Fri Feb 22 18:13:59 EST 2019

BEGIN: Enable CDC for Database s2k has begun. (18036)
Change Table area in use: CDC_Track_Data. (18043)
Change Table index area in use: CDC_Track_Idx. (18044)
Adding CDC files _Cdc-Table-Policy (18102)
Adding CDC files _Cdc-Field-Policy (18102)
Adding CDC files _Cdc-Change-Tracking (18102)
CDC feature has been successfully enabled. (18039)
END: Enable CDC for Database s2k has ended. (18036)
CDC policies enabled for: Customer,Orderline,Salesrep
hpux>
```

cdc_demo.sh

```
hpux>  cdc_demo.sh
procopy session begin for tom on batch. (451)
Database copied from /usr/dlc120/dlc/sports2000. (1365)
Procopy session end. (334)
OpenEdge Release 12.0 as of Fri Feb 22 18:13:59 EST 2019
14:33:57 BROKER     The startup of this database requires 17Mb of shared memory.  Maximum segment size is 1024Mb.
14:33:57 BROKER  0: Multi-user session begin. (333)
14:33:57 BROKER  0: Before Image Log Initialization at block 0  offset 633. (15321)
14:33:58 BROKER  0: Login by tom on /dev/pts/3. (452)
OpenEdge Release 12.0 as of Fri Feb 22 18:13:59 EST 2019
14:34:00 BROKER     The startup of this database requires 23Mb of shared memory.  Maximum segment size is 1024Mb.
14:34:00 BROKER  0: Multi-user session begin. (333)
14:34:00 BROKER  0: Before Image Log Initialization at block 30  offset 8164. (15321)
14:34:00 BROKER  0: Login by tom on /dev/pts/3. (452)
14:34:00 BROKER  0: Started for 8888 using TCP IPV4 address 0.0.0.0, pid 13790. (5644)
hpux>
```

cdc_sync.sh s2k s2k_new Customer Orderline . . .

```
linux>  cdc_sync.sh s2k s2k_new Customer Orderline Salesrep
running cdc_sync processes:
root     14984 14977  0 17:13 pts/3    00:00:00 _progres -b -db s2k -ld src -S 8888 -db s2k_new -ld dst -p cdc_sync_table.p -param Customer,3,yes -debugalert
root     14985 14977  0 17:13 pts/3    00:00:00 _progres -b -db s2k -ld src -S 8888 -db s2k_new -ld dst -p cdc_sync_table.p -param Orderline,3,yes -debugalert
root     14986 14977  0 17:13 pts/3    00:00:00 _progres -b -db s2k -ld src -S 8888 -db s2k_new -ld dst -p cdc_sync_table.p -param Salesrep,3,yes -debugalert
```

cdc_monitor.sh s2k

```
hpux>  cdc_monitor.sh s2k
```

CDC Cache

Cache Name	Cache Size	Total Entries	Total Hits	Total Misses
CDC Primary	1536000	3	5251	0
CDC Secondary	512000	0	0	0

CDC Latch Activity

_Latch-Name	_Latch-Lock	_Latch-Wait
MTL_CDC	0	0

CDC Change Queue Monitor

	first	last	queued
CDC_Customer	1	1	1
CDC_OrderLine	?	?	?
CDC_Salesrep	1	5,247	5,247

Type “q” to quit the monitor.

cdc_demo_stop.sh

```
hpux>  cdc_demo_init.sh

Really stop all running CDC demos? y

The CDC Demo has been stopped.

hpux>
```