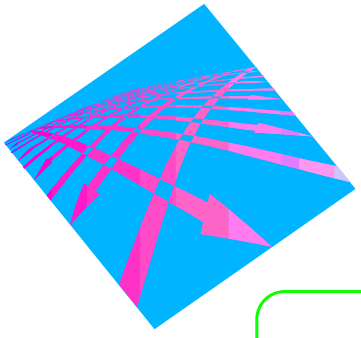


Capture
Apply 詳細

詳細説明
-DPROPR V8 Product Details－

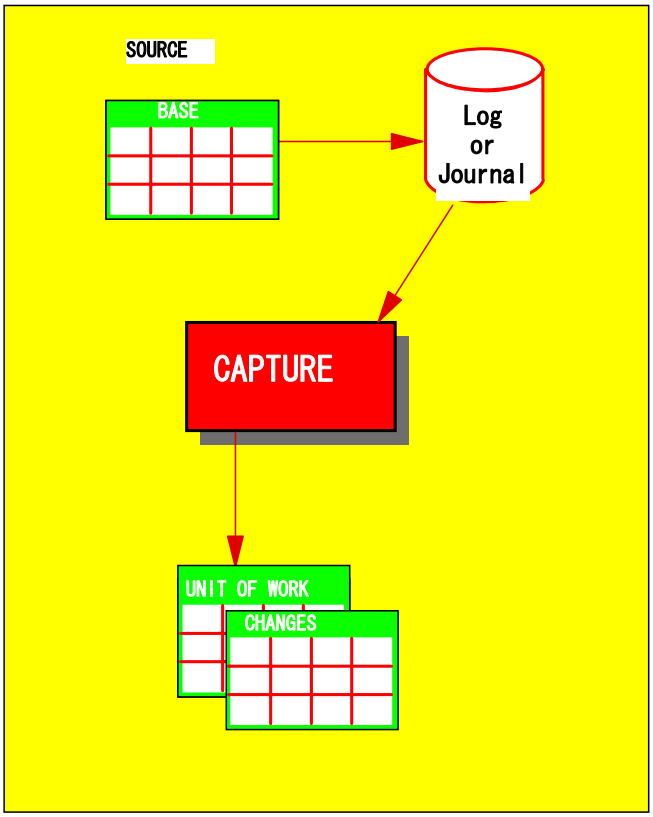
＜第2. OO版＞2006年01月

お断り: 当資料は、DB2 SQL Replication または DpropR V8 をベースに作成されています。

Agenda

- Capture
- Apply

Captureメインタスク



- ソース表での変更データがアクティブログまたは DB2/400 ジャーナルへログイング
- CaptureはLogまたは DB2/400 ジャーナルから対象となっている変更データをを読む
- Captureはその変更されたデータをCD表へまたCommitの情報UOW表へ書き出します。

解説:

■CAPTUREプログラムとは
DB2 は、すべてのトランザクションを診断および回復に備えてログ・ファイルに記録します。
Capture プログラムは、DB2 ログを監視して、複製ソースとして定義されているソース表から変更レコードを検出します。
Capture プログラムは、DB2 (MVS 版) 4.1 以上および DB2 ユニバーサル・データベース上の活動ログおよびアーカイブ・ログから変更およびコミット情報を取り出します。
これらのレコードには、行の変更前イメージと変更後イメージが入っています。
Capture プログラムは、これらの変更を変更データ (CD) 表に取り込みます。
Captureは、さらに、コミット済み作業単位に関する情報を作業単位 (UOW) 表に保管します。
この表は、Applyがコミット済み更新を識別してレプリケーションするためにCD 表と結合されます。
Apply プログラムは、その後、コミット済み更新をターゲット・サイトにコピーし、それらをソース表のコピー(ターゲット表)に適用することができます。

■Capture/MVS,VM,UDBを使用してSOURCE表から変更データを収集する場合にDATA CAPTURE CHANGES属性が必要
例)
```
CREATE TABLE TEST_SOURCE
    (ROW_NUMBR SMALLINT NOT NULL,
     TEST_DATA SMALLINT,
     DESCRIPTION CHAR(20)) DATA CAPTURE CHANGES;
```
または
```
ALTER TABLE TEST_SOURCE DATA CAPTURE CHANGES;
```

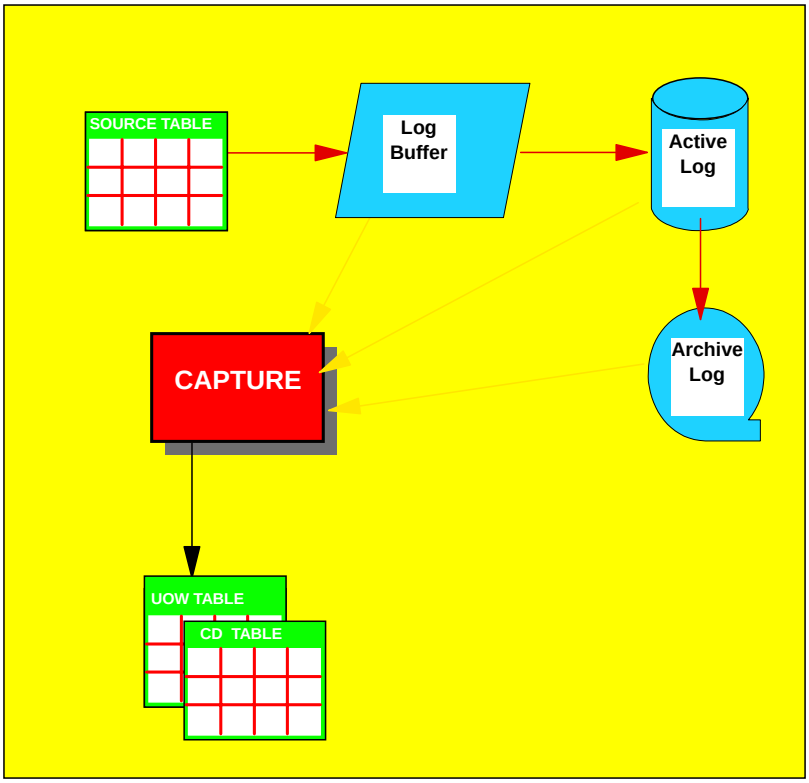
■UDBの場合にはDataBaseの構成でLOGRETAIN ON またはUSEREXIT ONである必要がある
例)
```
UPDATE DB CFG for AZUMADB using LOGRETAIN ON;
```

■AS/400上ではDATA CAPTURE CHANGES属性を持つことはできない

■CAPTUREはSQL更新(SQL INSERT,DELETE,UPDATE)の変更収集を行なう。DB2 UTILITYによる変更を検知することはできない。
DB2/MVS LOAD Utility をLOG(YES)で実行してもCapture/MVSは検知できない
UDB LOAD Utilityは検知できないが、IMPORT Utilityは内部的にはINSERTなので差分収集可能

■DB2/MVS EDITPROCが定義されたTABLEはサポートされない

CAPTUREの動き -INSERT PHASE



- ソース表での変更データはまずログバッファー上に入り、次にアクティブログ、最後にアーカイブログ上に記録される
- CaptureはDB2が提供するAPI(MVS:IFI)を使用しログバッファー、アクティブログ、アーカイブログ上のいずれかにあるログレコードを読み込む
- Captureは変更データをCD/UOWへinsertする。

CD:変更データ表
UOW:作業単位表

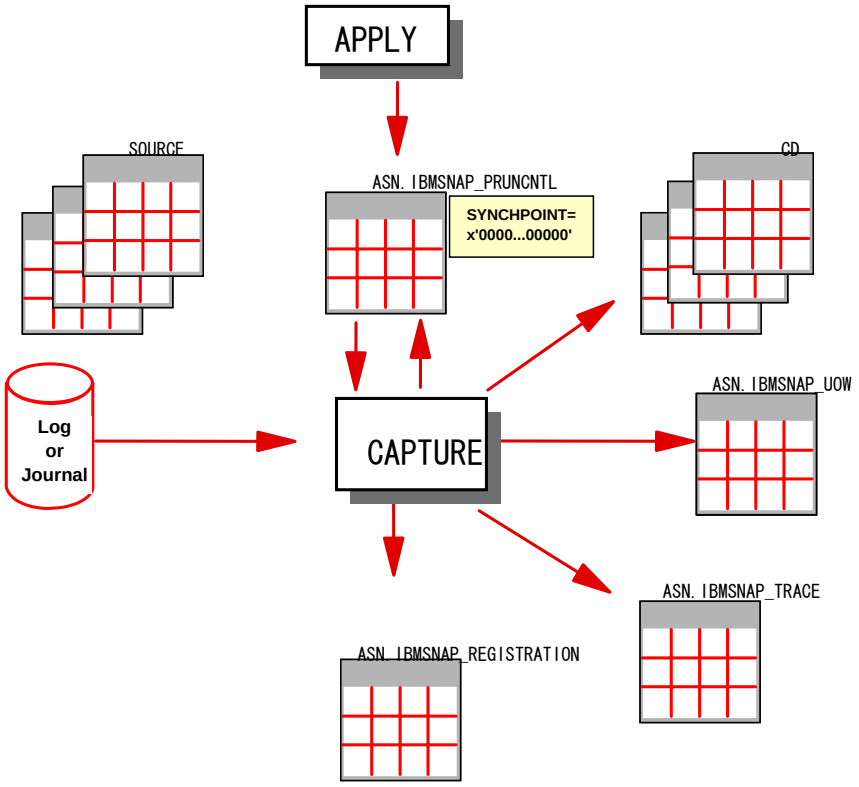
解説:

■CAPTUREはDB2 LOGを直接読み込むのではなく、DB2 APIを発行しDB2 LOGの読み込みを行なう
■DB2はLOG Buffer,Active log,Archived Logのいずれかにある LOG DataをCaptureへ返す
- UDBの場合は、対象のログ番号が既にアーカイブされ別DirectoryへCopyされていても (UserexitがON) Retreive要求がUDB側へ発行される

■V8からCaptureはCOMMITされた変更Dataのみを CD表へ書き込む(INSERT)、未COMMITの変更DataはCaptureタスクのMemory内に格納されROLLBACKされた場合にはCD表には書き出さない。V7までは変更がRollbackされた場合、Captureは削除(DELETE)を行なっていた
■UOW表へのINSERTは対象ソース表のCOMMIT時のたびに行われる。
■CD、UOW表はCaptureの稼動するローカルなロケーションに配置される必要がある。リモートは不可。
- AS/400 Remoteジャーナル使用は例外

■CaptureはDB2/MVSではSubsystem単位、UDBはDataBase単位にまたスキーマ毎に複数稼動可能
■AS/400はシステム単位でスキーマ毎に複数稼動可能、DB2/VM,VSEはデータベースマシン(仮想計算機)単位(V8未対応)

Capture変更収集の開始[V7 Handshaking]



ApplyがFullrefreshする直前にPC表のSYNCHPOINTをX'00..0'で更新する.これはCaptureにその表に関する変更データ収集を開始させるシグナルになる

CaptureはX'00..0'を検知し現行のLog Positionの値に変換する。その値はREGISTER表のCD_OLD_SYNCHPOINT,CD_NEW_SYCHPOINTの初期値ともなる。

CaptureはTRACE上へ"GOCAPT"messageを出力する

解説:

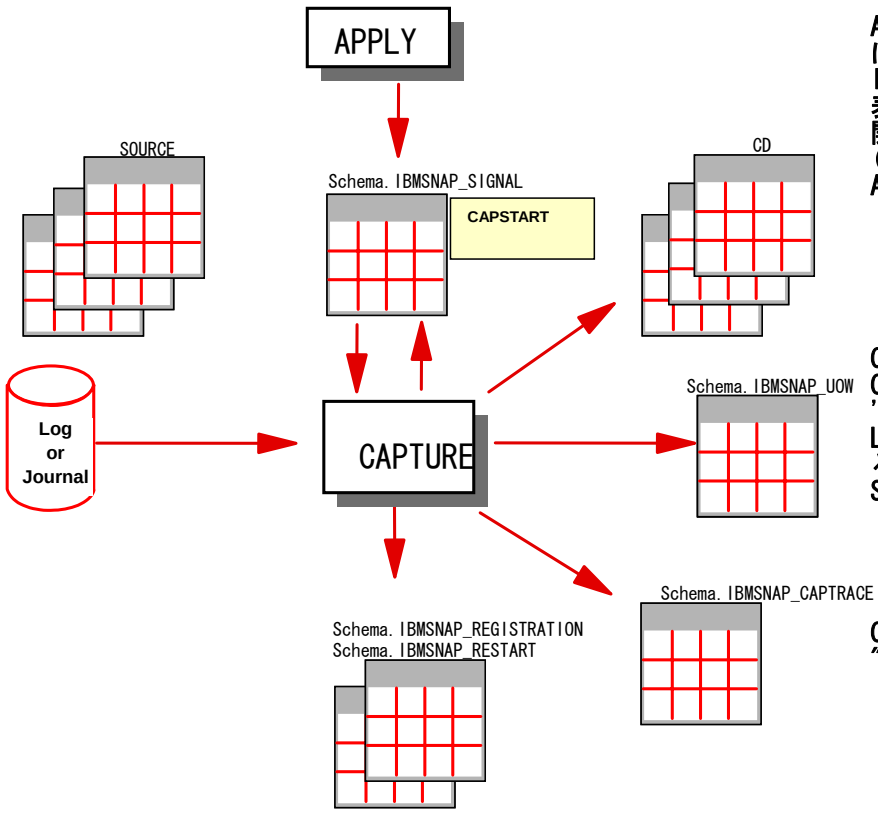
■Applyコンポーネントはソース表からFullrefreshを実行する前にソースサーバーに接続し、PC表にある該当行のSYNCHPOINTをHex'00000000000000000000'、SYNCHTIMEをCurrent Timestamp（ソースサーバー上の現時刻）へ更新する。CaptureはPC表に付加されているData Change Capture属性を介してこの更新を検知、SYNCHPOINTがHexゼロであればその更新されたLog Position番号をSYNCHPOINTへ再び更新する。またREGISTER表のCD_OLD_SYNCHPOINT,CD_NEW_SYNCHPOINT値もそのLog Position値へ変更(初期値はNULL)する。

■Captureは最後にASN.IBMSNAP_TRACE表へ"GOPAPT"メッセージを出力する。

```
OPERATION       DESCRIPTION
---------       ------------------------------------------------------------
GOCAPT          ASN0104I: 変更収集が所有者 = AZUMA,表名 = TEST_SOURCE
                ログ順次番号(LSN)00000000000008FA8A1D)に対して開始されました。
```

■APPLYがFULLREFRESHを実行した時にCAPTUREが停止していた場合、WARM STARTで再始動した場合はGOCAPTを出力可能であるが、COLD STARTで再始動した場合はSYNCHPOINT値はNULLに初期化されてしまう。次回のAPPLYの実行は再びFULLREFRESHになるのでAPPLY起動とCAPTUREの起動の順序には注意が必要。

Capture変更収集の開始[V8 Handshaking]



ApplyがFullrefreshする直前にSIGNAL表へ'CAPSTART'をINSERT.これはCaptureにその表に関する変更データ収集を開始させるシグナルになる(同時にV7と同様にPC表もApplyは更新する)

CaptureはSignal表のData Capture Change属性から'CAPSTART'を検知し現行のLog Positionの値に変換する。その値はSIGNAL表のSIGNAL_LSNに記録される。

CaptureはCAPTRACE上へ"INFO"messageを出力する

解説:

■Applyコンポーネントはソース表からFullrefreshを実行する前にソースサーバーに接続し、SIGNAL表へ対応するソース表のMAP_IDと共に" CAPSTART"をSIGNAL_STATE 'P'(Pending)でInsertする。CaptureはSignal表に付加されているData Change Capture属性を介してこの更新を検知、その更新されたLog Position番号をSIGNAL_LSNへ再び更新する

```
SIGNAL_TYPE   SIGNAL_SUBTYPE   SIGNAL_INPUT_IN  SIGNAL_STATE SIGNAL_LSN
------------- ---------------- ---------------- ------------ ----------------------
CMD           CAPSTART         0                C            x'3D4C2096000000030000'
```

■"CAPSTART"をSIGNAL表へインサート時誤ったMAP_ID(SOURCE_TABLEに対応するID)を挿入すると下記のエラー

```
ASN0064E  CAPTURE "ASN". The registration is not valid for an associated subscription having MAP_ID
          "0". The Capture program cannot start capturing change data for this subscription.
```

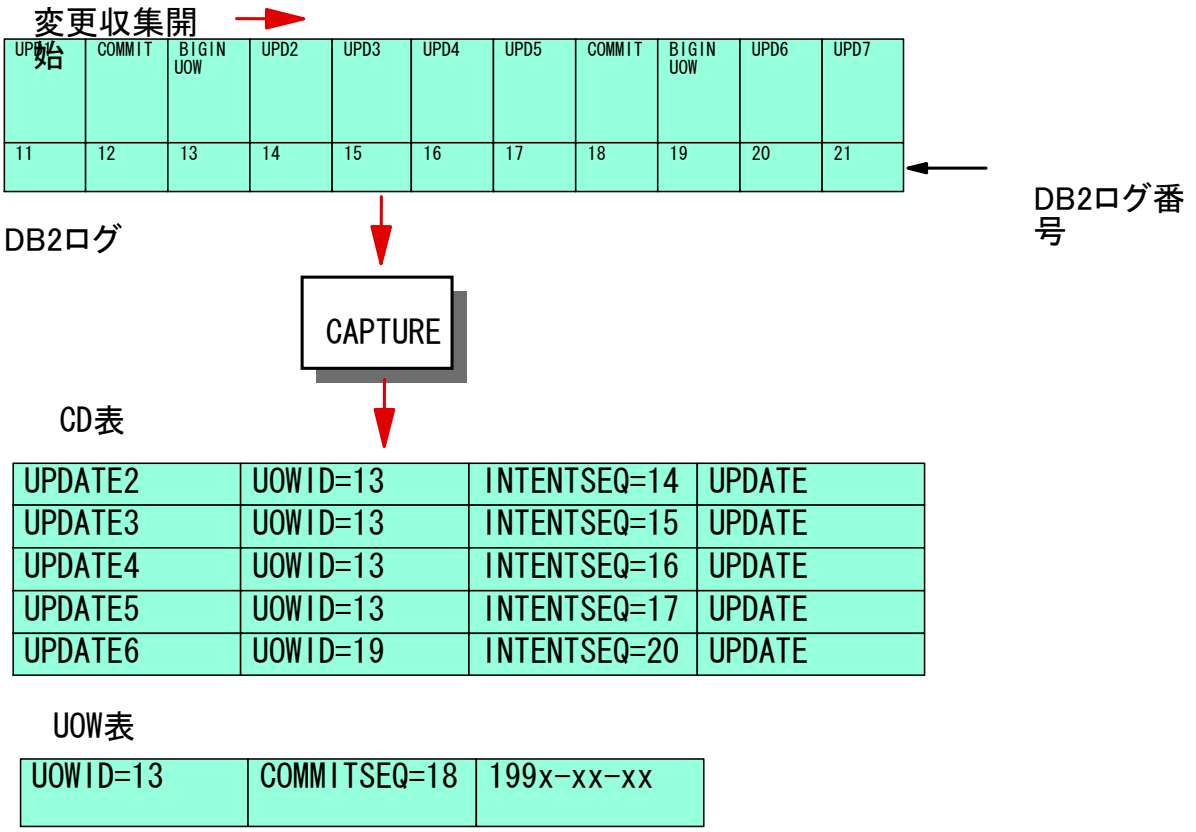
■Captureは最後にIBMSNAP_CAPTRACE表へ"INFO"メッセージを出力する。

```
OPERATION     DESCRIPTION
---------     ------------------------------------------------------------------
INFO          ASN0104I  CAPTURE "ASN". In response to a CAPSTART signal with MAP_ID "0",
              change capture has started for the source table "AZUMA"."TEST_SOURCE"
              for changes found on the log beginning with log sequence number
"0000:0000:0000:4a01:0c60".
```

■V7ではAPPLYがFULLREFRESHを実行した時にCAPTUREが停止していた場合、WARM STARTで再始動した場合は変更収集を可能であるが、COLD STARTで再始動した場合はSYNCHPOINT値はNULLに初期化されてしまう。次回のAPPLYの実行は再びFULLREFRESHになるのでAPPLY起動とCAPTUREの起動の順序には注意が必要であった

■RESTART表内にデータがある場合(CAPTUREが過去起動したことを意味する)、V8ではAPPLYが先行して実行され、その後CAPTUREがCOLDスタートされた場合、Fullrefreshが実行されることは変わらないが、もしRESTART表が空の場合にはAPPLYは最初の起動でFULLREFRESHせず、CAPTUREの起動を待つように変更された。これはCAPTUREのCOLD STARTが間に実行されることによるFULLREFRESHが２回実行されることを防止することを意味する

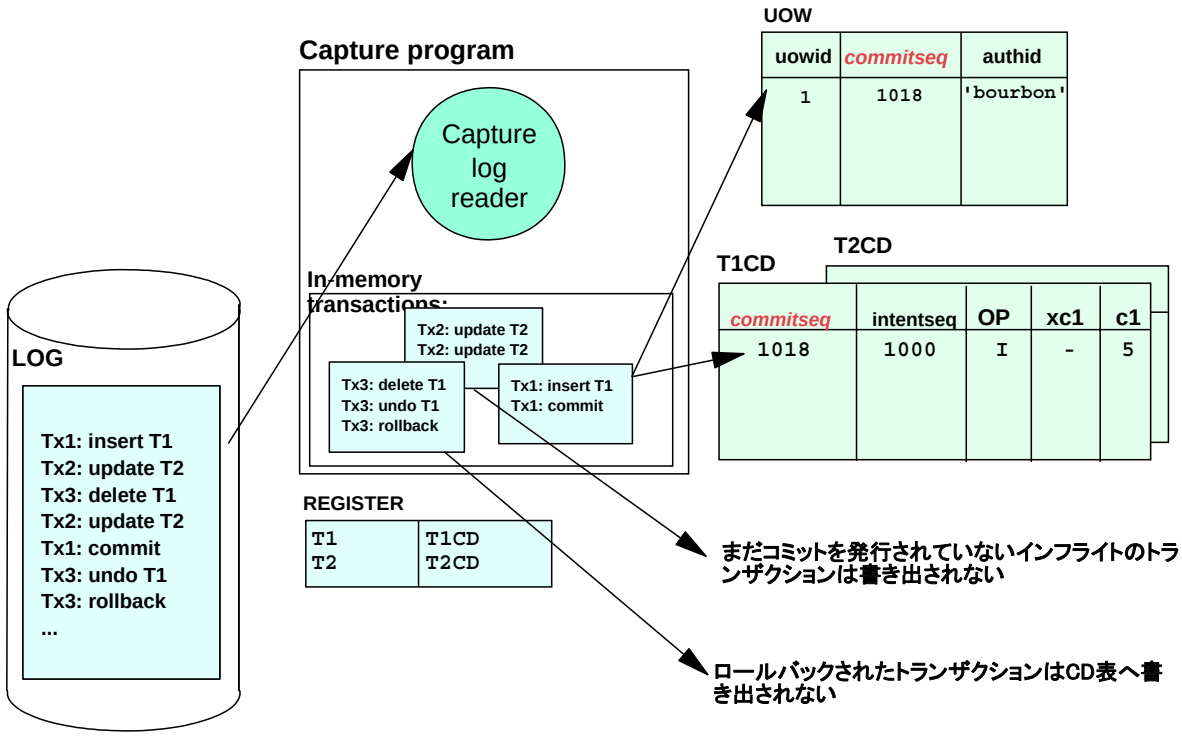
CD表のデータとLOG番号 [V7]



解説:

■上記の例はCD表、UOW表とDB2更新LOG内容を簡易的にマッピングした概略図です。
■APPLYがFULLREFRESHを実行し、CAPTUREがGOCAPTを出力した(つまり変更収集を開始した)のはLOG番号13からと仮定。
■CAPTUREはまだCOMMITされていなくてもLOG番号13以降の変更DATAをCD表へINSERTする。この時CD表のUOWIDはトランザクションが開始したBIGIN UOWのLOG番号の13が割り振られる、つまり一つのCOMMITスコープで複数の更新が実行されたのであればCD表のUOWIDは同じ番号が振られる。
■またCD表のINTENTSEQ列には１つのUOW内の各更新トランザクションに実行順序を識別するための相対番号が入る。
■更新トランザクションがCOMMITされるとCAPTUREはUOW表へUOWIDとCOMMITSEQ番号 (COMMITが発行されたLOG上のポジション) をINSERTする
■APPLYはCD,UOW表をUOWIDが一致するレコードを選び出し(COMMITされたレコードのみ) 、TARGETへ反映する。

CD表のデータとLOG番号 [V8]



解説:

■V8のCaptureではできるかぎりJOINの必要性を排除した。
Applyがターゲット表のタイプUSERCOPYの場合、CD/UOW表のJOINをしない。
Captureがプルーニング処理中にCD/UOW表のJOINをしない。
■ロールバックされたトランザクションのINSERT + DELETE処理のコストを排除
■正確なウォームスタートポイント
V8 Captureは常に一番低いインフライト状態のログ番号と一番高いコミット済みのログ番号を追跡する
CD/UOW表にCOMMIT済みの変更のみ書き出し、COMMIT_INTERVAL値で指定された時間でCOMMITを発行する
上記の変更によりGETLSEQコマンドが廃止された(SIGNAL表の情報提供のため)

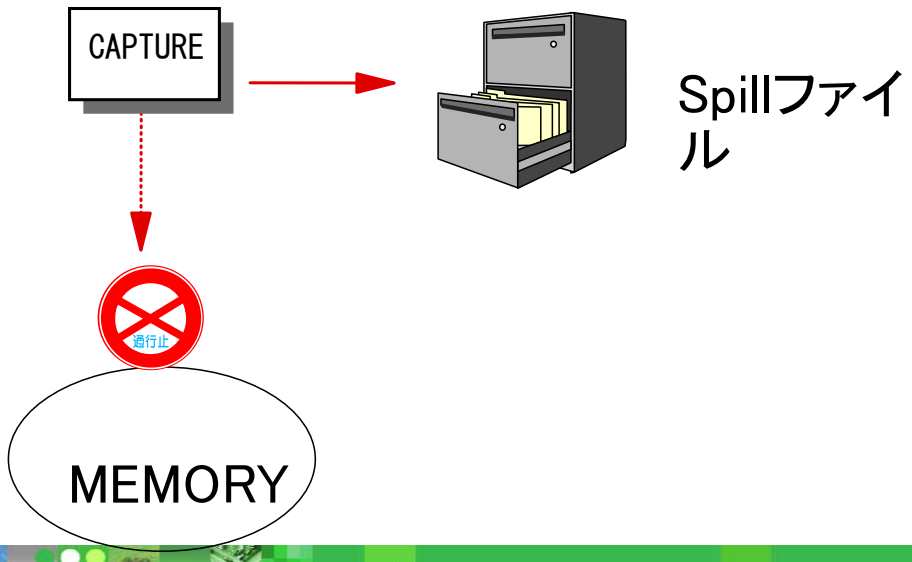
解説:

■CAPTUREは未コミットの変更データがMEMORY_LIMITに達するとスピルファイルへ書き出す。spillファイルはWindows,Unixの場合は１トランザクション毎にCapture_PATHディレクトリーに作成される。z/OSの場合はVIO(仮想入出力) へ書き出される。Capture稼動時にデータがメモリーからSpillファイルへあふれたかどうかはIBMSNAP_CAPMON表のTRAN_SPILL列を調べることによりチェック可能。また同じ表のCURRENT_MEMORY列を調べることにより現在使用中のメモリー量を知ることも可能

■Spillファイルの例(Windows,z/OS)

1,079,892 DB2.V8DB.ASN.CAP.00000000412c.spill

-rw-r--r-- 1 AZUMA OMVSGRP 3132456 Sep 15 00:31 dd:ASNS0001

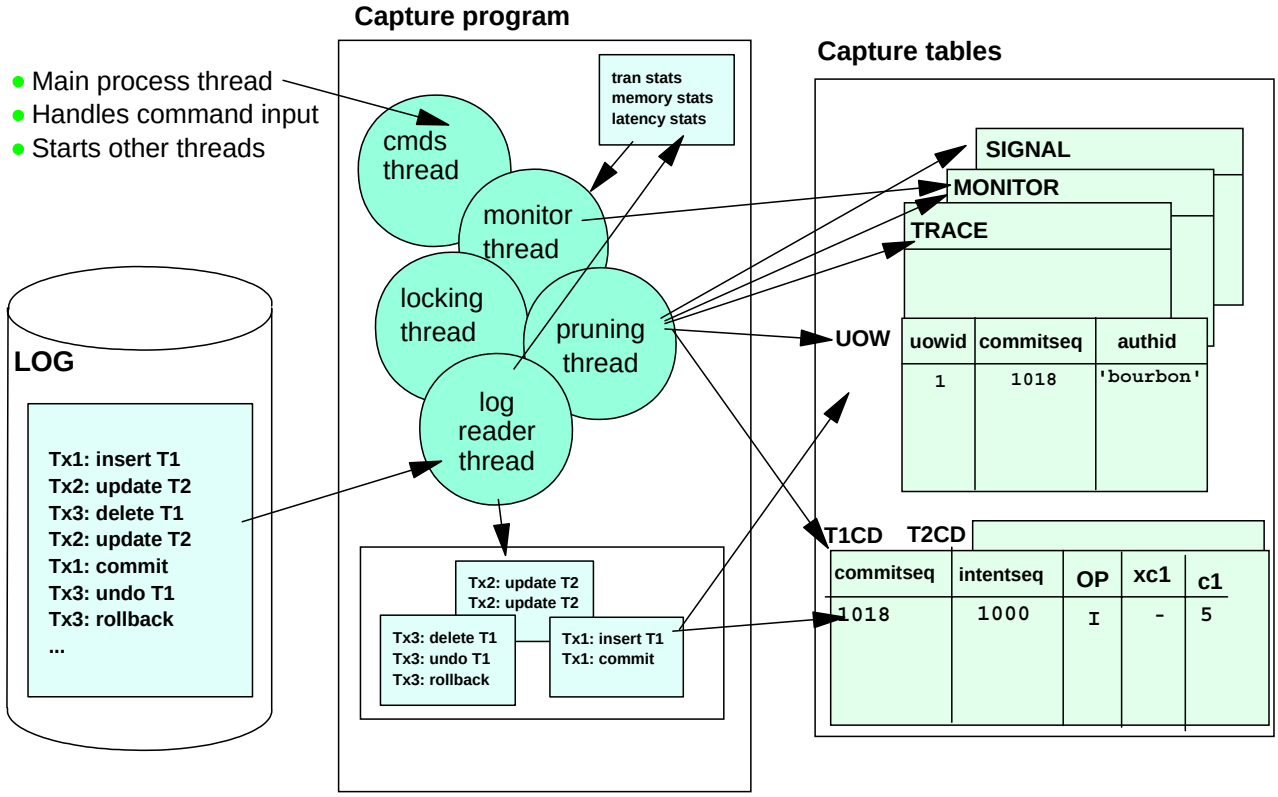


解説:

■このページはブランクです。

マルチスレッドCapture

■５つの役割



解説:

■５つのスレッド
Cmds(Administration) thread
Pruning thread
Monitor thread
Locking(serialization) thread
Log Reader thread

list applicationsの例

```
Auth Id  Application     Appl.      Application Id                    DB         # of
         Name            Handle                                       Name       Agents
-------- --------------- ---------- -------------------------------- ---------- -----
AZUMA    asncap.exe      86         *LOCAL.DB2.009583193122          V8DB       1
AZUMA    asncap.exe      85         *LOCAL.DB2.007D03193121          V8DB       1
AZUMA    asncap.exe      84         *LOCAL.DB2.0044C3193120          V8DB       1
AZUMA    asncap.exe      83         *LOCAL.DB2.007143193119          V8DB       1
AZUMA    asncap.exe      81         *LOCAL.DB2.003703193117          V8DB       1
AZUMA    db2bp.exe       48         *LOCAL.DB2.0075C3182753          V8DB       1
AZUMA    db2bp.exe       2          *LOCAL.DB2.00B9C3172513          V8DB       1
```

■パフォーマンスの向上
CD,UOW表の容量を節約
ApplyのJOIN不要処理での処理時間向上(Order Byの排除)

■ユーザビリティーの向上
モニターデータの提供

解説：

■５つのスレッド (z/OS)
Cmds(Administration) thread
Pruning thread
Monitor thread
Locking(serialization) thread
Log Reader thread

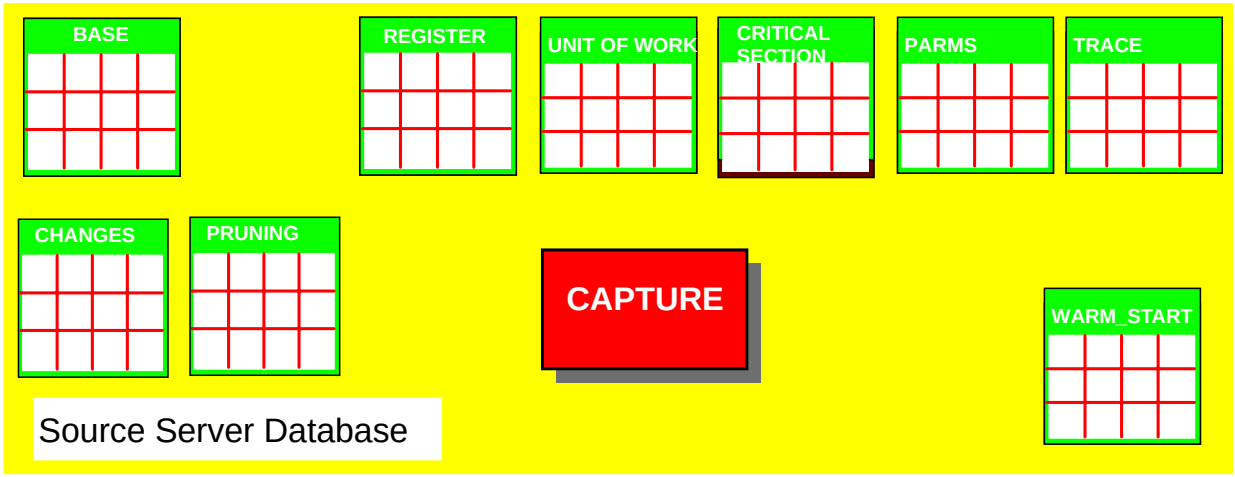
-DIS THREAD(*)の例

```
DB2CALL  T       24 CAPSTSTC     AZUMA     ASNTC810 0040    10
DB2CALL  T        5 CAPSTSTC     AZUMA     ASNTC810 0040    11
DB2CALL  T      100 CAPSTSTC     AZUMA     ASNTC810 0040    12
DB2CALL  T      880 CAPSTSTC     AZUMA     ASNTC810 0040    13
DB2CALL  T      136 CAPSTSTC     AZUMA     ASNTC810 0040    14
TSO      T  *     3 AZUMA        AZUMA               0078    29
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -D71A DSNVDT '-DIS THREAD' NORMAL COMPLETION
***
```


解説：

■このページはブランクです。

CAPTUREコントロールテーブル [V7]



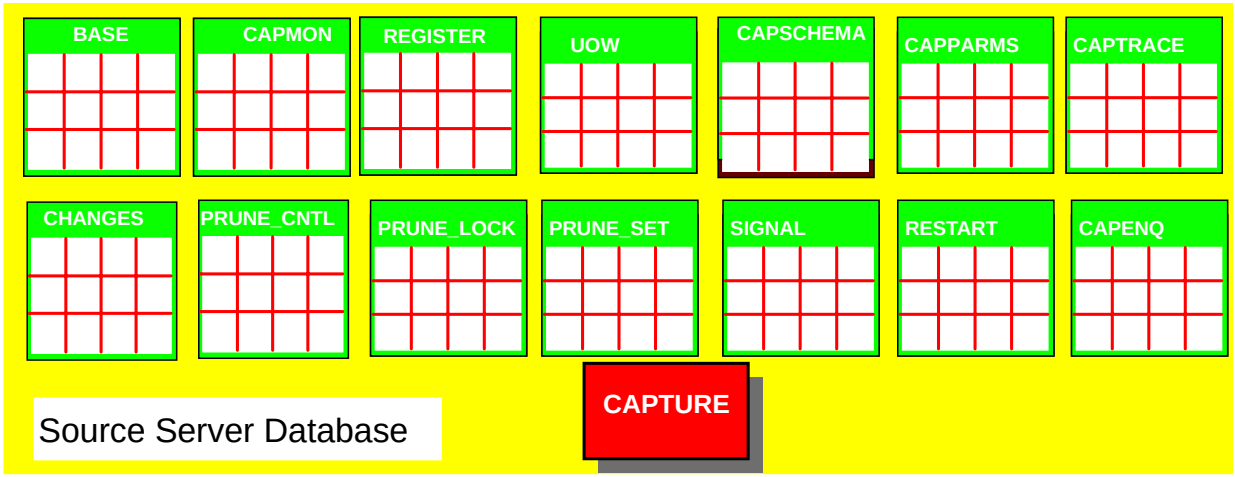
- BASE USER TABLE
- REGISTER ASN.IBMSNAP_REGISER
- UNIT OF WORK ASN.IBMSNAP_UOW
- CRITICAL_SECTION ASN.IBMSNAP_CRITSEC
- PARMS ASN.IBMSNAP_CCPPARMS
- TRACE ASN.IBMSNAP_TRACE
- CHANGES CDxxxxxxxxxx(名前はコントロールセンターが決定、変更可能)
- PRUNING ASN.IBMSNAP_PRUNCNTL
- WARM_START ASN.IBMSNAP_WARM_START

解説:

■CAPTURE V7で必要なテーブルは下記のとうり

CAPTUREコントロールテーブル名	保管されるサーバー	コントロールセンターCCからアクセスされる?	CaptureがData Capture Changes属性経由モニター?	Captureが参照する?	APPLYが参照する?
IBMSNAP_REGISTER	SOURCE_SERVER	YES	NO	YES	YES
IBMSNAP_PRUNCNTL	SOURCE_SERVER	YES	YES	YES	YES
IBMSNAP_TRACE	SOURCE_SERVER	NO	NO	YES	NO
IBMSNAP_WARM_START	SOURCE_SERVER	NO	NO	YES	NO
IBMSNAP_UOW	SOURCE_SERVER	NO	NO	YES	YES
IBMSNAP_CRITSEC	SOURCE_SERVER	NO	YES	YES	YES

CAPTUREコントロールテーブル [V8]



- CAPSCHEMA (ASN.IBMSNAP_CAPSCHEMA)
- CAPENQ (Schema.IBMSNAP_CAPENQ)
- CAPMON (Schema.IBMSNAP_CAPMON)
- CAPPARMS (Schema.IBMSNAP_CAPPARMS)
- CAPTRACE (Schema.IBMSNAP_CAPTRACE)
- PRUNCNTL (Schema.IBMSNAP_PRUNCNTL)
- PRUNE_SET (Schema.IBMSNAP_PRUNE_SET)
- PRUNE_LOCK (Schema.IBMSNAP_PRUNE_LOCK)
- REGISTER (Schema.IBMSNAP_REGISTER)
- SIGNAL (Schema.IBMSNAP_SIGNAL)
- UOW (Schema.IBMSNAP_UOW)

解説:

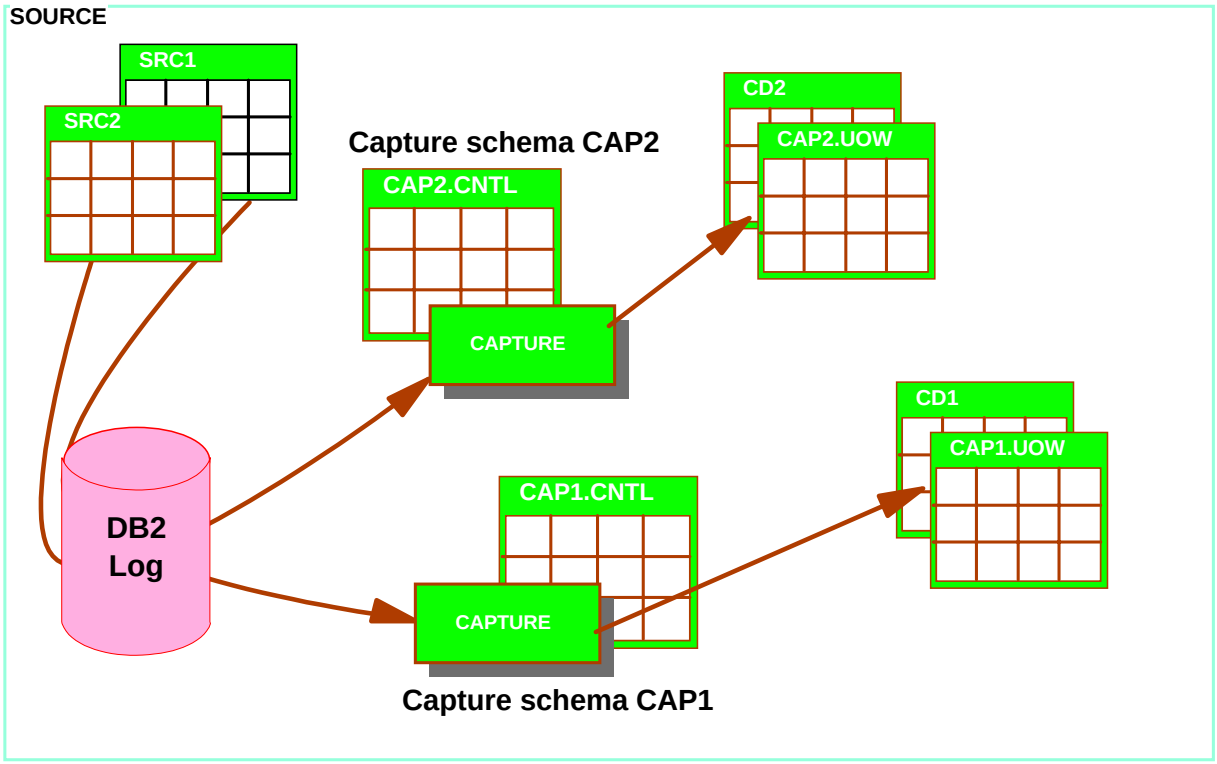
■Capture Control Tableの位置
■DB2 UDB
Replication Centerから作成する場合、UOW表とその他のコントロール表で１つずつの表スペースを割り当てる
■z/OS(OS/390)
1table/１表スペースが理想的
LOCK SIZEに関しては下記表のようにアサイン
REGISTER表やPRUNE_SET表などは同じ表スペースでLock LevelがPageに設定されているとCapture,Apply のDeadLockの原因になりえるので避ける。

z/OS CAPTRE コントロール表	推奨される表スペース LOCKSIZE
UOW	ANY
PRUNE_LOCK	PAGE
CAPMON	PAGE
CAPENQ	PAGE
CAPPARMS	PAGE
RESTRAT	PAGE
CAPTRACE	PAGE
REGISTER	ROW
PRUNE_SET	ROW
PRUNCNTL	ROW
CAPCHEMAS	ROW
SIGNAL	ROW

■OS/400
導入時に自動的に表が作成されるのでReplication Centerから作成してはいけない（トリガーが特定の表に作成される為

複数スキーマ

■Captureは複数のコントロール表を定義可能



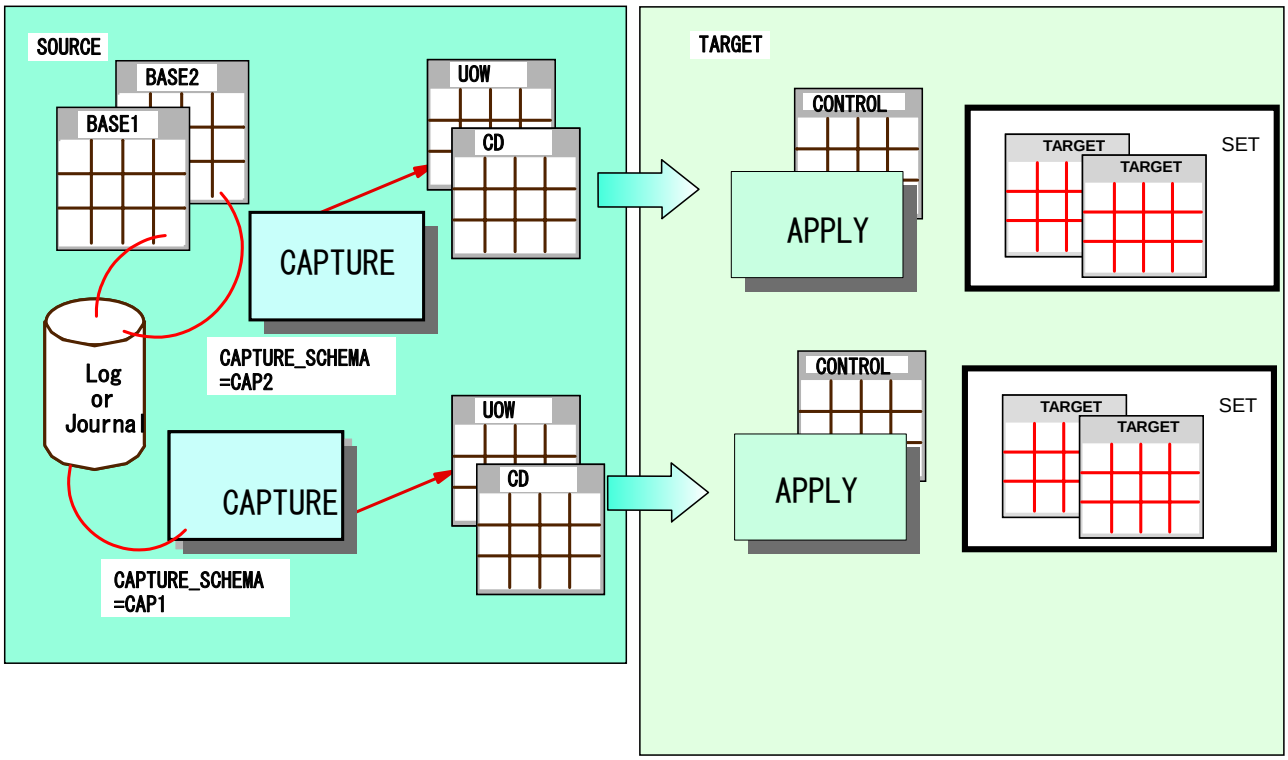
解説:

■複数のスキーマを持つ利点および注意

- １つのCaptureは非常にデータの鮮度が要求されるアプリケーション、もう一方はデータの鮮度がそれほど要求されないケースでCaptureのチューニングパラメータ(COMMIT_INTERVALなど）を変更できる。
- z/OSの環境でUNICODE,EBCDICなどソース表のエンコードが異なる場合
- SMP環境のようにCPUが複数用意されている場合に有利である
- 他社DBからのReplicationを構築する場合、Federated DBを複数用意する必要がない
- Transaction整合性のため、Applyが参照するSET内のメンバーは同じCaptureスキーマを対象にしなければならない
- デメリットとしてLog Read APIの増加によりCPUのOverheadが増加する
- ASN.IBMSNAP_CAPCHEMA表内にスキーマを保持する

解説:

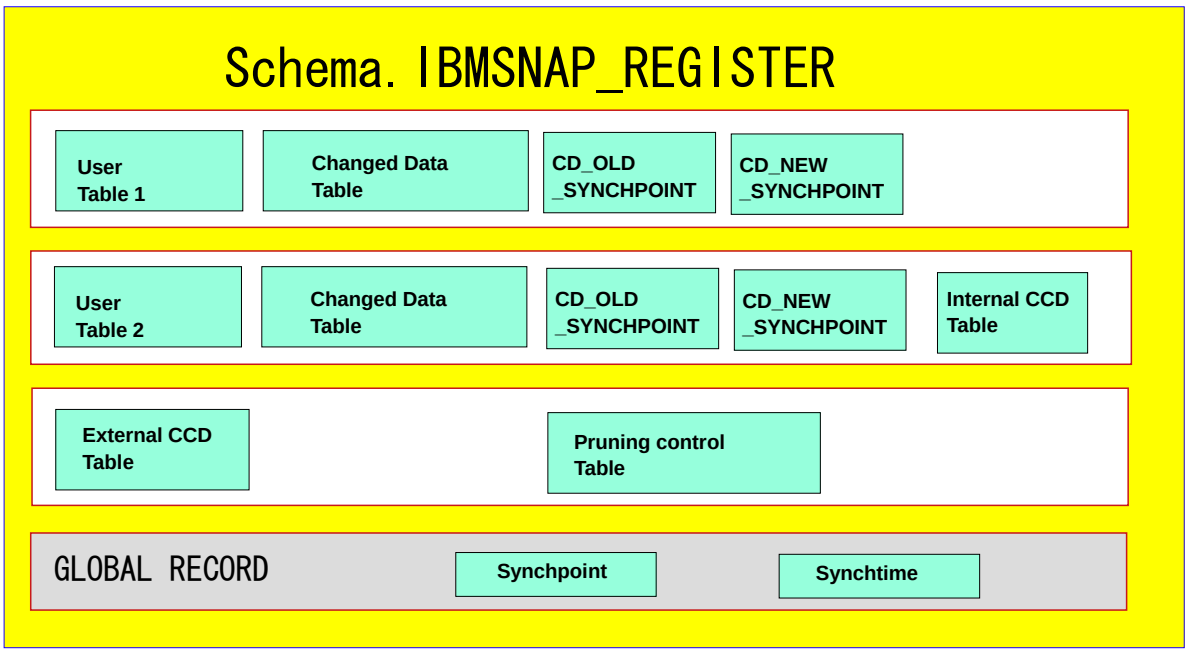
■1SETが複数のCaptureをまたがってはいけない



解説:

■このページはブランクです。

REGISTERテーブル



解説:

■REGISTER表にはCAPTUREが変更収集対象とする複写元ソースとCD表のマッピング情報がコントロールセンターからレプリケーション表の登録を実行した時入力される
■１ソース表に対してREGISTER表に１行INSERTされ、その行のに中に対応するCD表やCCD表の情報が含まれる。
■CD_OLD_SYNCHPOINT,CD_NEW_SYNCHPOINT値は表の登録時は初期値NULL、変更収集開始時時にカレントのLOG番号にCAPTUREによって更新される。
■もしCD_TABLEカラムがNULLであればCAPTUREは差分収集を実行せず、APPLYは毎回FULLREFRESHを実行する
■GLOBALレコードにはCATUREが読み込んだ最高位のLOG番号(SYNCHPOINT)、TIMESTAMP(SYNCHTIME)を記録している。この値はCOMMIT_INTERVAL値ごとにCAPTUREにより更新される。省略時は30秒。

```
GLOBAL_RECORD SOURCE_OWNER     SOURCE_TABLE     CD_OLD_SYNCHPOINT                  CD_NEW_SYNCHPOINT                  SYNCHPOINT                 SYNCHTIME
------------- ---------------- ---------------- ---------------------------------- ---------------------------------- -------------------------- --------------------------
N             AZUMA            TEST_SOURCE      x'00000000000008FA8A1D'            x'00000000000008FA8A1D'            -                          -
Y                                                -                                  -                                  x'00000000000008FB0CD8'  1999-04-10-18.44.35.000000

  2 レコードが選択されました。
```

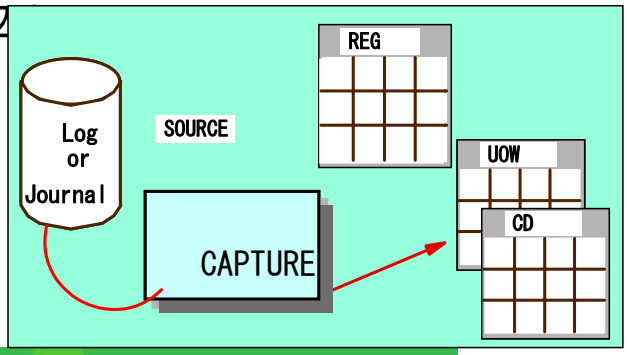
Captureと登録情報

■Captureは始動時、Register表から登録情報を読み込む

- 活動化(active)されている表情報のみメモリーにロード
- Capture稼動中に新規登録された表は活動化された時点で動的に読み込む
  - ►"REINIT"が不要
- 特定の条件変更ではREINITが必要
- エラーになった登録情報
  - ►無視あるいはCaptureの停止(STOP_ON_ERRORカラム...Y/N)

■CD表へのALTER ADD カラム

- CaptureはALTER ADD カラムにてCD表の構造が変更されていてもエラーにせず、システムカタログ情報を再度読み込
  - ►変更収集が開始されCD表のINSERT時



■登録情報の削除

- CAPTUREを停止不要
  - ►CMD, 'CAPSTOP'シグナル

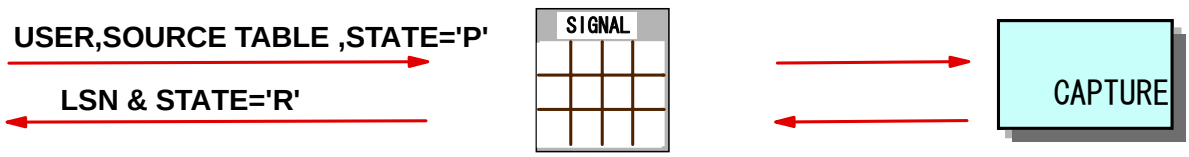
解説:

■登録が追加された時、Captureが稼動中でも再初期化(REINIT)を行う必要はない
■下記の登録情報を変更した場合はREINITを実行する
CHGONLY
CONFLICT_LEVEL
RECAPTURE
DISABLE_REFRESH
CHG_UPD_TO_DEL_INS
STOP_ON_ERROR
BEFORE_IMGAGE_PREFIX (初期値がnullの場合のみ)
■カラムを手動で追加する手順 (SIGNAL表の利用)

1. ソース表の変更活動の停止を確認する
2. USERシグナルを発行し、CAPTUREによってSIGNAL_STATUS='R'に変換された事を確認し、LSN番号(SIGNAL_LSN)を入手する

```
INSERT INTO ASN.IBMSNAP_SIGNAL (SIGNAL_TYPE,SIGNAL_SUBTYPE,SIGNAL_INPUT_IN,SIGNAL_STATE)
VALUES ('USER','AZUMAS','AZUMA.TEST_SOURCE','P');
```

3. 対応するAPPLYがそのLSNまで処理を完了したことを確認
4. ソース表へALTER TABLE COLUMN ADDを実行
5. RCを使用して登録情報を変更(CD表へのALTER TABLE COLUMN ADDを実行)
6. RCを使用してサブスクリプション情報の変更(SUBS_COLS表へINSERT,TARGET表へALTER TABLE COLUMN ADDを実行)



解説:

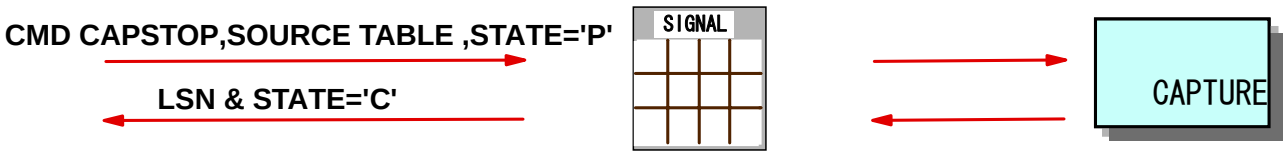
■登録を削除するプロシージャーは以下のようになる

1. 関連するサブスクリプションの非活動化を確認する
2. RCから、または手動でCMDシグナル CAPSTOPを発行し、CAPTUREによって登録済み表の変更収集を停止する
   SIGNAL_STATUS='C'に変換された事を確認する
   REGISTER表のSTATE A->Iに変更される。
   Captureから返却されるLSNはSIGNAL表の挿入を含むDB2 LSNを返す

```
INSERT INTO ASN.IBMSNAP_SIGNAL
  (SIGNAL_TYPE,
   SIGNAL_SUBTYPE,
   SIGNAL_INPUT_IN,
   SIGNAL_STATE)
VALUES ('CMD',
        'CAPSTOP',
        'AZUMA.TEST_SOURCE',
        'P');
```

3. RCからまたは手動でCD表とREGISTER表からのエントリーの削除を実行するを実行)

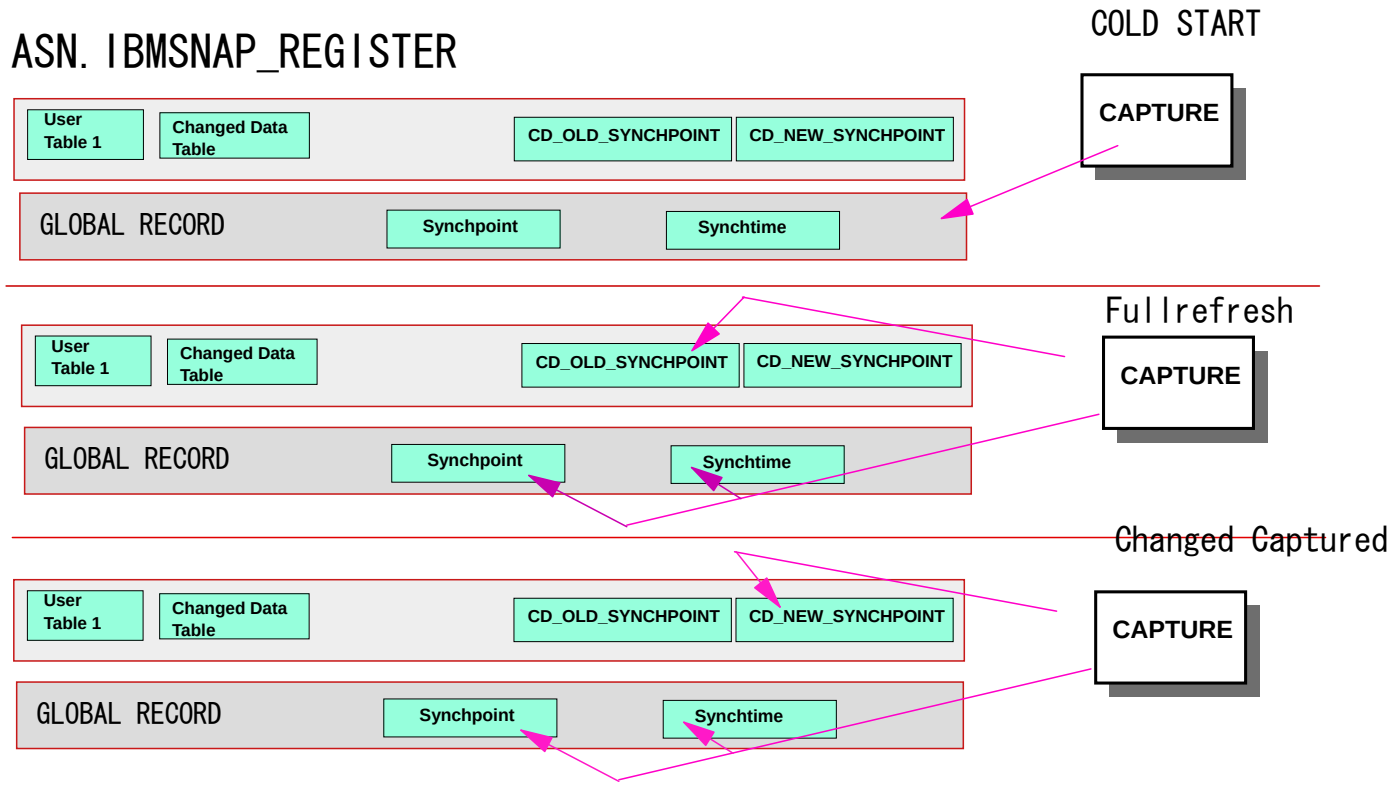
変更収集を停止せず、REGISTER表からエントリーを削除するとCAPTUREは停止してしまうので注意



解説:

■このページはブランクです。

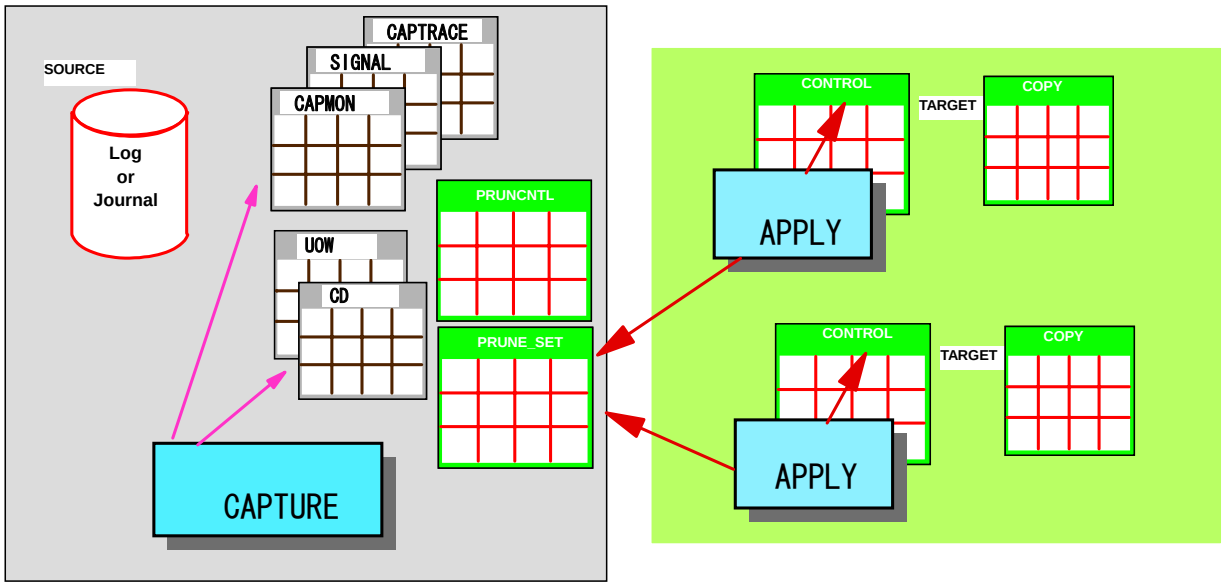
CAPTUREとREGISTER表の内容



解説:

- ■① CAPTUREが起動された時、GLOBAL RECORDがなければREGISTER表へINSERTする
- ■②FULLREFRESH時にCD_OLD_SYNCHPOINT,CD_NEW_SYNCHPOINTをNULL値からLOG番号へ変換する
- ■③複写元表に変更があった場合にUOW表のCOMMITSEQ値でCD_NEW_SYNCHPOINTの値を更新する
  APPLYが変更DATAの読み込み、CD/UOW表のJOINをするかどうかの判断基準となる
  ASN.IBSMSNAP_SUBS_SET表のSYNCHPOINT値 < ASN.IBMSNAP_REGISTER表の.CD_NEW_SYNCHPOINTであれば
  変更データがあると判断する。この比較処理によって実行時不必要なCD/UOW表のJOINを行なわない
  変更データ存在した場合でもV8ではUSER_COPY表の場合にはCD表のみアクセスしUOW表とのJOINは実行しない
- ■④CAPTUREはCOMMIT_INTERVAL値(30秒) のタイミングで読み込んだ最高位のLOG番号と時刻をGLOBAL RECORDに書き込む

CAPTURE プルーニング(Pruning)



- CaptureはApplyがPRUNE_CNTL表に記録したSYNCHPOINT値を使用してのCD表にある既に適用済みのデータのPRUNING(不要データの消し込み)を実施する。
- CD表のdelete処理は1UOWで行なわれる
- Captureはすべてのcd表のDELETE処理終了後、UOW表のDELETEを行なう
- Captureはmonitor_limit値を使用しMONITOR表のDELETE処理
- Captureはretention_limit値を使用しSIGNAL表のDELEET処理
- Captureはtrace_limit値を使用しTRACE表のDELETE処理

解説:

■CAPTUREはREGISTER表からPRUNING対象のCD表を選択する

```
SELECT CD_OWNER, CD_TABLE, SOURCE_STRUCTURE, CONFLICT_LEVEL FROM "ASN".IBMSNAP_REGISTER
WHERE SOURCE_VIEW_QUAL = 0 AND SOURCE_STRUCTURE <> 3 AND PHYS_CHANGE_TABLE IS NOT NULL
AND CD_OLD_SYNCHPOINT IS NOT NULL
```

■１つの複写元表から複数の複写先TARGET表へReplication定義がある場合、一番遅いSYNCHPOINTに検索する。

```
SELECT MIN(A.SYNCHPOINT) FROM "ASN".IBMSNAP_PRUNE_SET A, "ASN".IBMSNAP_PRUNCNTL B
WHERE A.TARGET_SERVER = B.TARGET_SERVER AND A.APPLY_QUAL = B.APPLY_QUAL AND A.SET_NAME = B.SET_NAME
AND B.PHYS_CHANGE_OWNER = ? AND B.PHYS_CHANGE_TABLE = ?
AND B.SYNCHPOINT <> x'00000000000000000000' AND B.SYNCHPOINT IS NOT NULL
```

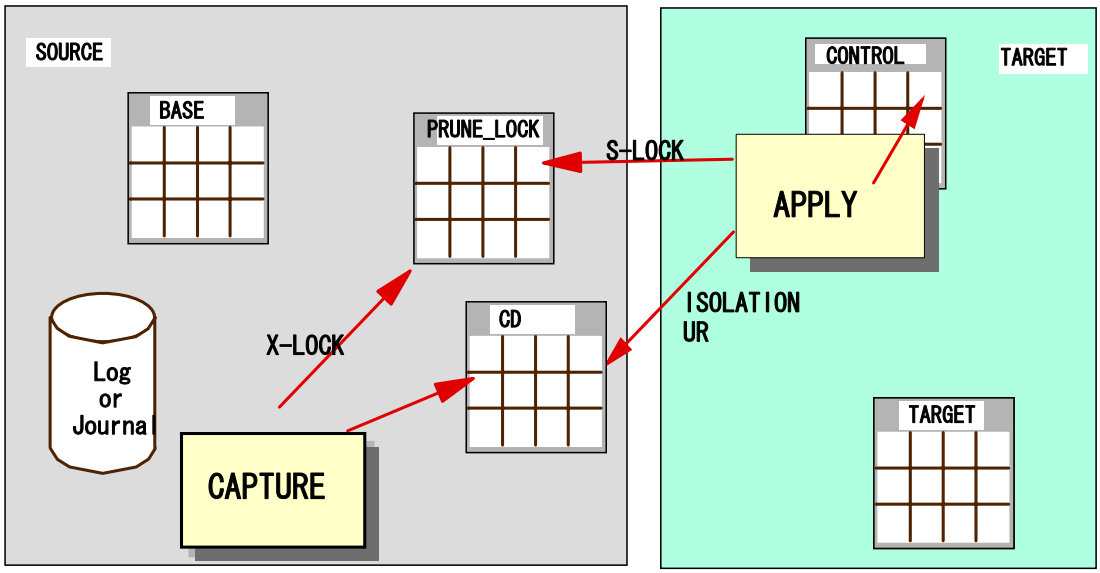
■DELETE処理の実施 (DELETE処理はおのおの別のunit of workで実行される)

```
CD表を行単位のDELETE
  Cursor    : DELCDCUR
  Cursor was blocking: FALSE
  Text      : SELECT 1 FROM "AZUMA"."CD01" WHERE IBMSNAP_COMMITSEQ <= ?
              FOR UPDATE OF IBMSNAP_OPERATION OPTIMIZE FOR 10 ROWS
DELETE FROM "AZUMA"."CD01" WHERE CURRENT OF DELCDCUR

UOW表のDELETE
  Cursor    : DELUOWCUR
  Cursor was blocking: FALSE
  Text      : SELECT 1 FROM "ASN".IBMSNAP_UOW WHERE IBMSNAP_COMMITSEQ <= ? AND IBMSNAP_REJ_CODE = '0'
              FOR UPDATE OF IBMSNAP_REJ_CODE OPTIMIZE FOR 10 ROWS
DELETE FROM "ASN".IBMSNAP_UOW WHERE CURRENT OF DELUOWCUR
```

■NOPRUNEオプションを使用してCAPTUREを稼動させた場合はPRUNE Command実行時にDELETE処理が行なわれる。

PRUNE_LOCK表

コールドスタートおよびRETENTION_LIMITプルーニング時アクセス直列化

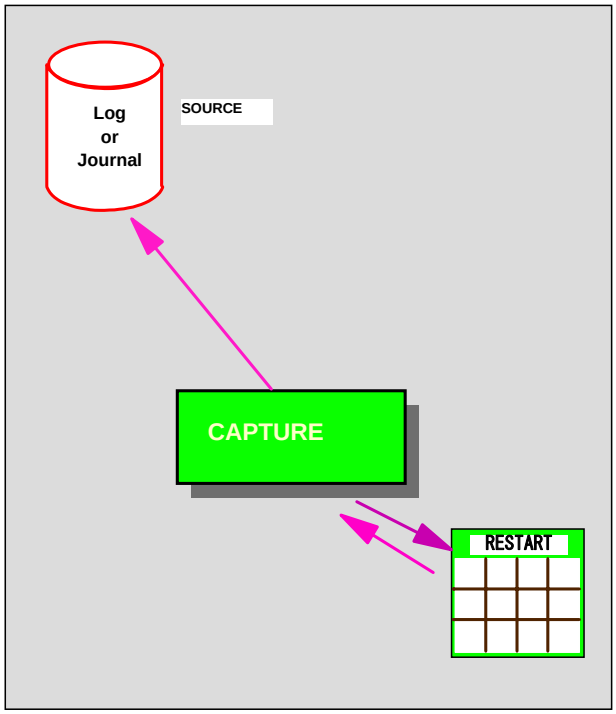


- CaptureはCOLD START,RETENTION_LIMITプルーニング開始前にPRUNE_LOCK表にX-LOCKを取得する。
- APPLYは変更データをREGISTER表やCD/UOW表を読む直前にShare LockをPRUNE_LOCK表に対して取得する、もし取得できない場合にはRETRYする

解説:

■CAPTUREはCOLD START,RETENTION_LIMITプルーニング開始前にPRUNE_LOCK表に対してEXCLUSIVE LOCK(排他ロック) を取得。もしAPPLYが稼動していた場合APPLYのSHARE LOCK(共有ロック) が解放されるまで待ちになる
■APPLY実行時はPRUNE_LOCKC表にはSHARE LOCK(共有ロック) を取得するが、CD/UOW表を読み込む時はISOLATION URを使用する。

CAPTUREウォームスタートプロセス



•Capture WARM Start

Captureは起動時、COLDスタート時RESTART表のデータを削除する。また変更収集を開始した場合にREGISTER表のグローバルレコードを更新する同じUOWでCommit_Internal値毎にLOGの情報をRestart表を更新する。
次回のWARM Start時はその値+1から処理を再開する。

RESTART表が空の場合にはWARMNSスタートは失敗し。WARMSI,WARMSAの場合には自動でCOLDスイッチされる。(WARMSIがSwitchするのはRestart表が空の場合には初期状態と考えるため)

•その他、CaptureがCOLDスタートになるケース
  •ASN.IBMSNAP_CCPPARMSで指定されたLAG_LIMITを超えた
  •WARMSTART時に読もうとしたLOGのポイントが読取れない
など

•CaptureがCOLDスタートへswitchした場合
  •WARNING 2002-08-20-16.02.59.079000 ASN0102W CAPTURE "ASN". The Capture program switches to cold start because the warm start information is insufficient

解説:

■RESTART表の内容

MAX_COMMITSEQ	MAX_COMMIT_TIME	MIN_INFLIGHTSEQ	CURR_COMMIT_TIME
x'3D61E9A304B571C00000'	2002-08-20-16.02.59.000000	x'0000000000000BC7C4C0'	2002-08-20-16.46.56.902000

MAX_COMMITSEQ	CD,UOW表にコミット済みの最大ログシーケンス値
MAX_COMMIT_TIME	MAX_COMMITSEQ列のシーケンス値に関連したタイムスタンプ
MIN_INFLIGHTSEQ	ウォームリスタート時に開始するログシーケンス値
CURR_COMMIT_TIME	この表がCaptureによって更新されたローカルタイムスタンプ
CAPTURE_FIRST_SEQ	最後のCOLDスタート時のログシーケンス値

■CaptureのData鮮度

CURR_COMMIT_TIME - MAX_COMMIT_TIME

■MIN_INFLIGHTSEQ

DB2がバックアップを完了し、LOGファイルを削除したい場合Captureが必要とするLOGファイルの名前を見つける入力値になる

解説：

■DB2FLSN

db2flsn コマンド

- コマンドに指定したログ順序番号で特定されるログ・レコードが、どのログ・ファイルに含まれているかを判断することが可能
- db2flsn -q input_LSN
  - "-q" オプションによりログ・ファイル名のみの出力が可能

EXAMPLE:
〈Input〉　db2flsn -q 000000BF00030
〈Output〉　S0000002.LOG

EXAMPLE:
〈Input〉　db2flsn　000000BF00030
〈Output〉　Given LSN is contained in log file S0000002.LOG

解説：

■z/OSの場合、ログファイルの位置を調べる為のはDSNJU004ユーティリティーを使用する。
まず、RESTART表からMIN_INFLIGHTSEQを入手する

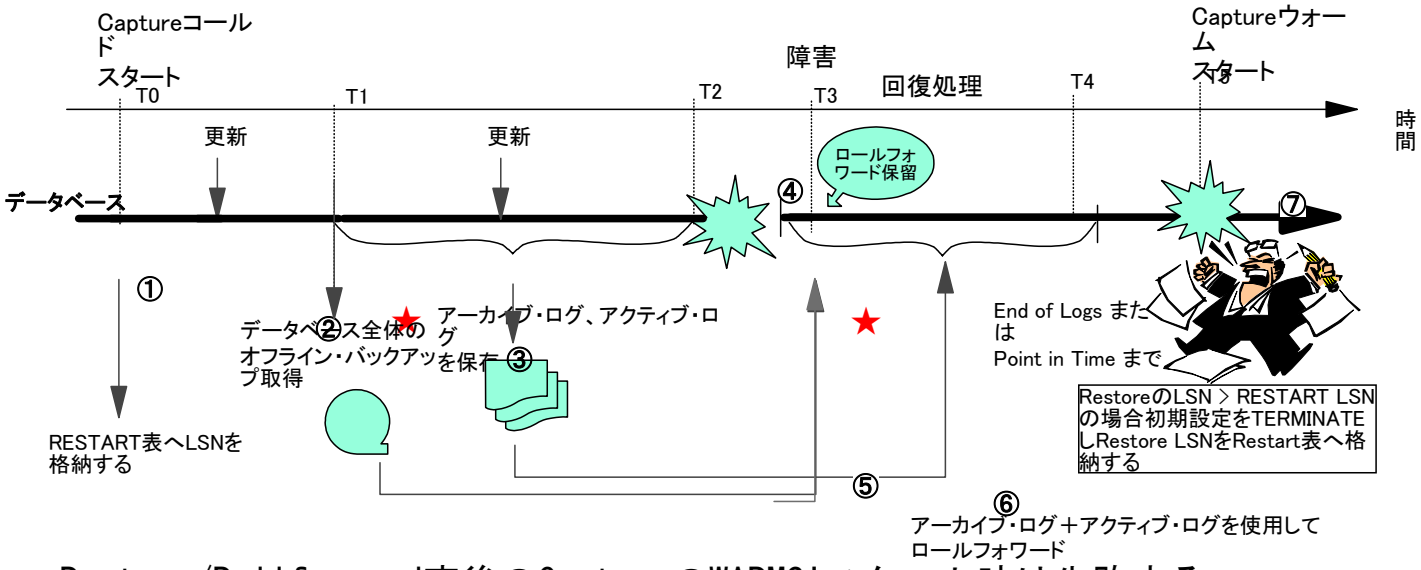
```
==> db2 'select max_commitseq,min_inflightseq from asn.ibmsnap_restart'

MAX_COMMITSEQ                  MIN_INFLIGHTSEQ
------------------------------ ------------------------------
x'000000003BF2A2640000'        x'000000003BF29F2A0000'
```

DSNJU004を実行し、開始と終了RBAの範囲から調べる。最後の4文字が0000になるのはData Sharing環境ではない為で
この4文字はメンバーIDに相当する

```
-ACTIVE LOG COPY 1 DATA SETS
0    START RBA/TIME          END RBA/TIME          DATE     LTIME DATA SET INFORMATION
  -------------------- -------------------- -------- ----- --------------------
0   00003756D000            00003972CFFF       2002.016  17:41 DSN=D71ACAT.LOGCOPY1.DS01
  2002.218  03:11:24.2  2002.248  12:57:18.9  PASSWORD=(NULL) STATUS=REUSABLE
    00003972D000            00003B8ECFFF       2002.016  17:41 DSN=D71ACAT.LOGCOPY1.DS02
  2002.248  12:57:18.9  2002.258  07:37:37.6  PASSWORD=(NULL) STATUS=REUSABLE
    00003B8ED000            00003DAACFFF       2002.016  17:41 DSN=D71ACAT.LOGCOPY1.DS03
  2002.258  07:37:37.6  ........  ..........  PASSWORD=(NULL) STATUS=NOTREUSABLE
1ARCHIVE LOG COPY 1 DATA SETS
0    START RBA/TIME          END RBA/TIME          DATE     LTIME DATA SET INFORMATION
  -------------------- -------------------- -------- ----- --------------------
0   000000000000            0000021BFFFF       2002.058   9:15 DSN=D71ACAT.ARCHLOG1.A0000002
  2002.018  03:56:57.9  2002.057  06:36:47.9                  PASSWORD=(NULL) VOL=ZOS1U1 UNIT=SYSDA
                                                             CATALOGUED
    0000021C0000            00000437FFFF       2002.073  17:32 DSN=D71ACAT.ARCHLOG1.A0000003
  2002.057  06:36:47.9  2002.073  08:32:39.9                  PASSWORD=(NULL) VOL=DB2P01 UNIT=SYSDA
                                                             CATALOGUED
```

DataBase Restore時のLSN Handling （UDBのみ）

■CAPTURE_FIRST_SEQを利用したRestore/Rollforwardの検出



- Restore/Rollforward直後のCaptureのWARMSIスタート時は失敗する
  - ２回目からはNo actionで成功する

ASN0144E CAPTURE "ASN". The program detected that the source database "V8DB" has been restored or rolled forward. A cold start is recommended to restore consistency.
ASN0008I CAPTURE "ASN". The Capture program was stopped.

解説：

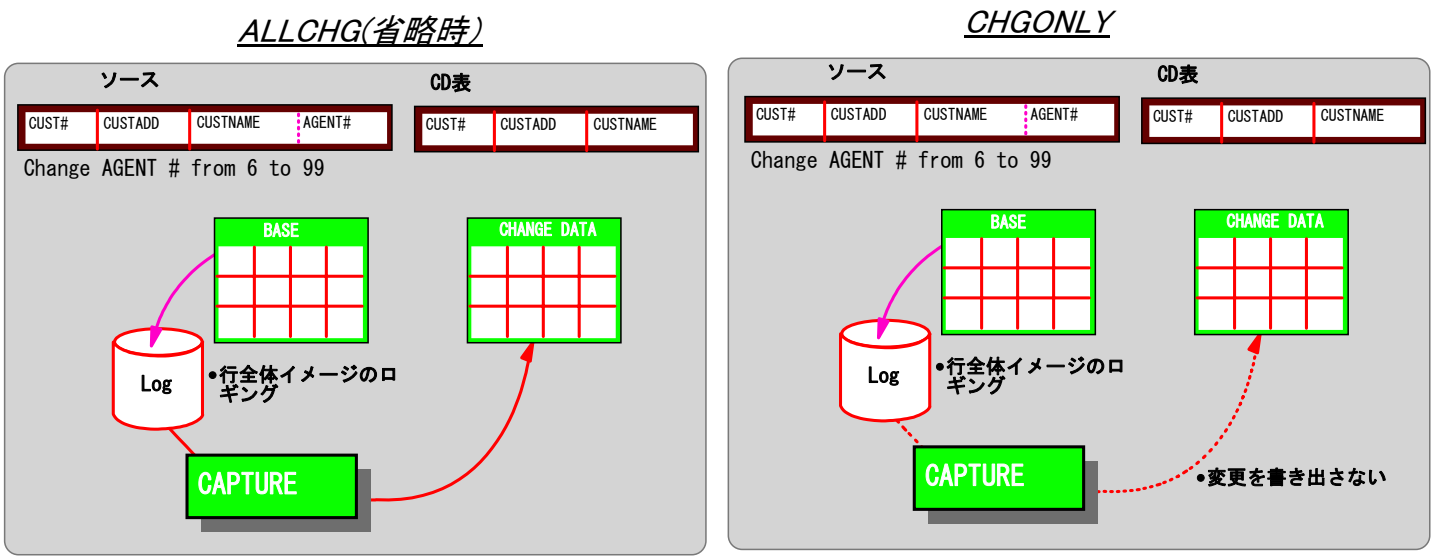
■ASN0144Eが検出される回復処理のシナリオは以下の通りです。

シナリオ	時刻
① CaptureをColdスタートします。	T0
Applyを起動させ全件COPY実行後に対して更新処理を行います。 アーカイブ・ログは削除しないように保存します。	T0-T1
② Captureを停止し、データベース全体のオフライン・バックアップ取得します	T1
③ データベースに対して更新処理を行います。アーカイブ・ログは削除しないように保存します。	T1-T2
④データベースに障害が発生し使用不能になったとします	T2
⑤ ②で保管してあったデータベース全体のオフライン・バックアップをリストアします	T2-T3
⑥ 保管してあったLOGを使用しRollforwardを実施します	T3-T4
⑦CaptureのWARMスタートを試みる -->ASN0144E	T5

Capture Registered Column

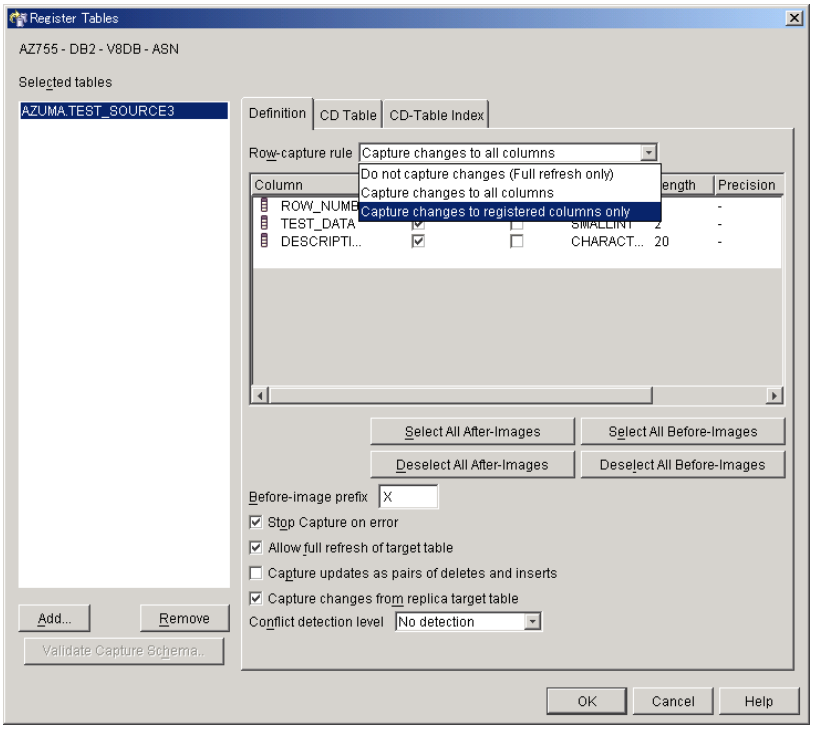
■複製に使用可能な列だけを収集するためのフラグ

- 複製に使用可能なものとしてマークした列だけに関する変更を収集したい場合は、CHGONLYを使って収集プログラムを開始することができます。 デフォルトは、収集プログラムはすべての列のソース表データに加えられた変更を収集します。



解説:

■表の登録時にCHGONLYを指定すると、ソース表に対するカラムの更新がCD表に定義されているカラムと一致する場合のみ、Captureは変更データを書き出すようになります。これによってTarget表に反映させる必要のないReplicationを抑止でき、CD表のSpaceを節約することが可能。V7まではCAPTUREの起動パラメーターであったが表レベルになりました。



CAPTUREの開始パラメーター

■スタートパラメーター

- capture_server
- *capture_schema*
- retention_limit
- lag_limit
- commit_interval
- prune_interval
- *trace_limit*
- *monitor_limit*
- *monitor_interval*
- *memory_limit*
- autoprune
- term **
- autostop **
- logreuse **
- logstdout **
- *sleep_interval* **
- *capture_path*
- *startmode*
- *add_partition* **

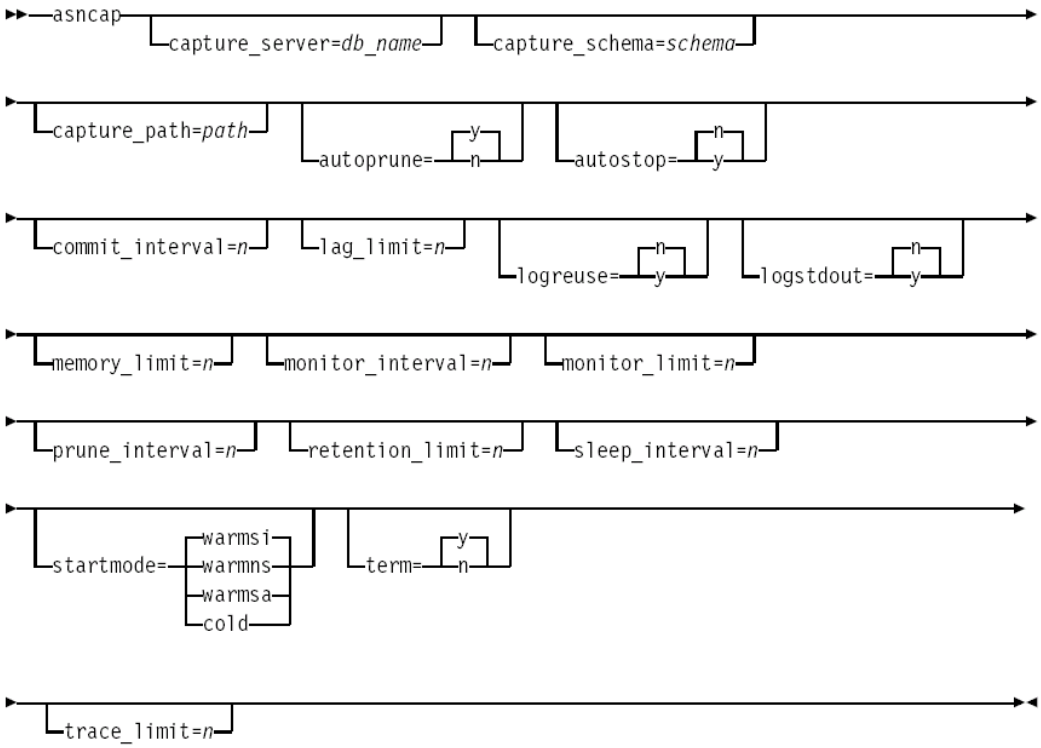
** parameter now available on z/OS, UNIX, and Windows

- ポジションは関係なし keyword = *value*
- ***Highlighted*** parameters(new function)

解説:

■Capture Parameter

asncap コマンドを使用してキャプチャー・プログラムを始動するための構文



解説:

■CAPTURE_SERVER
　Captureのコントロールサーバーの名前
　UnixおよびWindowsの場合は指定なしの場合DB2DBDFT環境変数の値
　z/OSの場合にはSubSystem Name, Data共用環境ではグループアタッチ名でなくメンバーのSubSystem Name
■CAPTURE_SCHEMA
　1-30文字のCAPTUREスキーマ名
■CAPTURE_PATH
　CAPTUREが使用する作業ファイルのロケーション,省略時はasncapが呼び出されたディレクトリー
■AUTOPRUNE=Y/N
　(CD)(UOW)(IBMSNAP_CAPMON)(IBMSNAP_CAPTRACE)(IBMSNAP_SIGNAL)表の自動プルーニングをするかどうか
■AUTOSTOP=Y/N
　CAPTUREの始動前にログに記録されたトランザクションを検索した後、CAPTUREを終了するかどうか
■COMMIT_INTERVAL (30 sec)
　CD,UOW表に書かれた内容をコミットする時間間隔
■LAG_LIMIT
　CAPTUREがログレコードを処理する際に許す最大の遅れを分で指定 (10080 MINUTES=7日)
　CAPTUREが起動時LAG_LIMITを検知した場合ASN0121Eのエラーになる
■LOGREUSE=Y/N
　CAPTUREプログラムがログファイルを再利用するかどうか
■LOGSTDOUT=Y/N
　CAPTUREがメッセージを標準出力(stdout)へ送信するかどうか
■MEMORY_LIMIT
　トランザクションを作成する為にCAPTUREが利用できるメモリーの最大サイズ(MB単位) このメモリー限度に達するとファイルへ書き出す。省略時は32MB
■MONITOR_INTERVAL
　CAPTUREがMONITOR表(IBMSNAP_CAPMON)表に行を挿入する間隔、省略時は5分
■MONITOR_LIMIT
　MONITOR表の行がPRUNING対象になるかの分数を指定、省略時は10,080分

解説:

■PRUNE_INTERVAL
　(CD)(UOW)(IBMSNAP_CAPMON)(IBMSNAP_CAPTRACE)(IBMSNAP_SIGNAL)表をPRUNINGする頻度値
　AUTOPRUNE=Nの場合このパラメーターは無視される
■LITENTION_LIMIT
　CD,UOW,SIGNAL表の行の最大保持期間、省略時は10,080分
■SLEEP_INTERVAL
　CAPTUREがLOG READする際、END OF LOGの場合に何秒スリープするかの秒数、省略時は5秒
■AOTOPRUNE:Y/N
　(CD)(UOW)(IBMSNAP_CAPMON)(IBMSNAP_CAPTRACE)表の自動プルーニングをするかどうか
■TERM=Y/N
　DB2が終了した場合にCAPTUREを終了するかどうか
　Y(省略時) DB2が終了した場合にCAPTUREは終了する。
　N　　DB2がMODE(QUIESCE)で終了した場合CAPTUREは待機モードになりDB2が始動されるとWARMモードで始動を自動開始する、DB2がFORCEまたは異常終了した場合はNでもCAPTUREは終了する
　　UDBでACCESS MAINTを使用して始動するとNであってもCAPTUREは接続できないので結果として終了する
■TRACE_LIMIT
　CAPTRACE表の行がPRUNING対象の適格になる分数を指定。省略時は10,080分
■ADD_PARTITION
　区画を追加時CaptureのWARM Start時に認識させるかどうか

解説:

■STARTMODE

(WARMSI:省略時)

ウォームスタート情報を入手可能な場合、直前の終了時点から再開。最初の起動の場合のみCOLDへ切り替える

これは、デフォルトの開始モードです。 キャプチャー・プログラムはウォーム・スタートで開始されますが、キャプチャー・プログラムを最初に再始動させた場合はコールド・スタートに切り替わります。

この開始モードを使用して、引き続きキャプチャー・プログラムを開始する際に予期せずコールド・スタートが開始されるのを防ぎます。 キャプチャー・プログラムがウォーム・スタートで開始すると、プログラムが終了したところから処理を再開します。キャプチャー・プログラムの開始後にエラーが起きた場合、キャプチャー・プログラムは終了し、すべての表を未処理の状態にします。

(WARMSA)

ウォームスタート情報を入手可能な場合、直前の終了時点から再開。ウォームスタートができない場合は常にCOLDへ切り替える

ウォーム・スタート情報が有効であれば、キャプチャー・プログラムは前回の実行において終了したところから処理を再開します。 通常、ウォーム・スタート情報は再開 (IBMSNAP_RESTART) 表にあります。

一部のケースでは、この情報が存在しないことがあります。 オペレーターがキャプチャー・プログラムを取り消したか、 DB2 を停止したことが考えられます。 そのような場合、キャプチャー・プログラムは CD, UOW または登録 (IBMSNAP_REGISTER) 表を使用して、プログラムが停止した時刻と再同期をとります。キャプチャー・プログラムがウォーム・スタートできない場合、コールド・スタートに切り替わります。 すべてのターゲット表がリフレッシュされてしまうため、通常はコールド・スタートに切り替えないことが多いでしょう

(WARMNS)

ウォームスタート情報を入手可能な場合、直前の終了時点から再開。ウォームスタートができない場合は常に終了する

ウォーム・スタートを妨げているあらゆる問題 (使用不可能なデータベースまたは表スペースなど) を解決する余地があります。この開始モードを使用して、予期せずコールド・スタートが開始されるのを防ぎます。

(COLD)

初期化をしてから起動する、CD,UOW,CAPTRACE表は常に削除される。コールド・スタートの間、キャプチャー・プログラムは 初期化でその CD 表および UOW 表 のすべての列を削除します。 これらのレプリケーション・ソースへのすべてのサブスクリプションは次のアプライ処理サイクルでフル・リフレッシュされます。

(すなわち、すべてのデータがソース表からターゲット表にコピー される)

解説: Capture/MVSスタートアップJCLの例

```
//ASNCAP    JOB CLASS=A,MSGCLASS=H,NOTIFY=AZUMA,
//              USER=SYSADM,PASSWORD=SYSADM,
//              REGION=0M,TIME=NOLIMIT
//*
//ASNCAP    EXEC PGM=ASNCAP,
//   PARM='ENVAR("LANG=en_US")/DB71 COLD prune
//              capture_schema=ASN logSTDout'
//STEPLIB DD  DISP=SHR,DSN=DSN72G.ASN810.SASNAPF
//         DD  DISP=SHR,DSN=CEE.SCEERUN
//         DD  DISP=SHR,DSN=DSN71G.SDSNLOAD
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//              UNIT=SYSDA,SPACE=(CYL,(50,70)),
//              DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS   DD PATH='/usr/lpp/db2repl_08_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD   DUMMY
//SYSTERM  DD   SYSOUT=*
//SYSUDUMP DD   DUMMY
//SYSPRINT DD   SYSOUT=*
//
```

ASN0100I CAPTURE "ASN". The Capture program initialization is successful.
ASN0109I CAPTURE "ASN". The Capture program has successfully 607 initialized and is capturing data changes for "0" registrations. "0" registrations are in a stopped state. "1" registrations are in an inactive state.

USSからの起動例

```
#asncap capture_server=D71A capture_schema=ASN startmode=COLD
```

ASN0100I CAPTURE "ASN". The Capture program initialization is successful.
ASN0109I CAPTURE "ASN". The Capture program has successfully 607 initialized and is capturing data changes for "0" registrations. "0" registrations are in a stopped state. "1" registrations are in an inactive state.

起動時必要なEnvironment Variables

```
(Time Zone)
export TZ=JST-9
(Globalization)
export NLSPATH=$NLSPATH:/usr/lpp/db2repl_08_01/msg/%L/%N
export LANG=En_US
(PATH)
export PATH=$PATH:/usr/lpp/db2repl_08_01/bin
(STEPLIB)
export STEPLIB=$STEPLIB:ASN810.DPROPR.V810.LOADLIB
```

解説：

■Capture for z/OS 稼働条件

- **FMID**
  - HAAW821 DB2 II Replication
  - HAAW820 DB2 Replication Common Library
  - HAAW822 DB2 DataPropagator

- **Programming Requirements**
  - 5694-A01 z/OS Version 1 Release 1 or higher
  - HFS Paths
  - SASNRHFS HFS /usr/lpp/db2repl_08_02/IBM/
  - DB2 for z/OS V7 or V8
  - V810はDB2とのインターフェースはCAF,V820はRRSAF

解説：

■このページはブランクです。

IBMSNAP_CAPPARMS表

■Capture始動時に指定がない場合、使用される値

- retention_limit (10080 mins)
- lag_limit (10080 mins)
- commit_interval (30 secs)
- prune_interval (300 secs)
- trace_limit (10080 mins)
- monitor_limit (10080 mins)
- monitor_interval (300 secs)
- memory_limit (32 MB)
- autoprune (Y)
- term (Y)
- autostop (N)
- logreuse (N)
- logstdout (N)
- sleep_interval (5 secs)
- capture_path (current dir)
- startmode (WARMSI)
- remote_src_server (NULL)

解説:

■CAPTURE_SCHEMAとCAPTURE_SERVERは始動時のみ指定できるパラメーターでCAPPARMS表には指定できない

CAPTURE操作コマンド

■Capture稼動中に実行可能なコマンド

- asnccmdコマンドを使用して以下の操作コマンドを入力可能:
  - PRUNE
  - REINIT
  - CHGPARMS
  - QRYPARMS
  - STATUS
    - もしCaptureが停止していた場合Captureとの接続が確立できない為、エラーが返却される.
  - SUSPEND
  - RESUME
  - STOP
    - CaptureはCMDシグナルでSTOP subtypeを使用して停止することも可能.
- Captureへの操作コマンドは: asnccmd、Applyへはasnacmdとなり、モニターへの操作コマンドはasnmcmd.

解説:

■操作コマンド

- Capture command syntax:

```
>>-asnccmd----+-------------------------------+----------->
              '-capture_server-- = --db_name--'

 >-----+---------------------------------+----------------->
       '-capture_schema-- =  --schema--'

 >-----+-chgparms--| parameters |--+--------------------><
       +-prune--------------------+
       +-qryparms-----------------+
       +-reinit-------------------+
       +-resume-------------------+
       +-status-------------------+
       +-stop---------------------+
       '-suspend------------------'
```

- 例(指定位置規則なし)

```
asnccmd status capture_server=srcdb1 capture_schema=asnprod
asnccmd capture_server=srcdb1 status capture_schema=asnprod
asnccmd capture_server=srcdb1 capture_schema=asnprod status
```

解説:

■指定例２

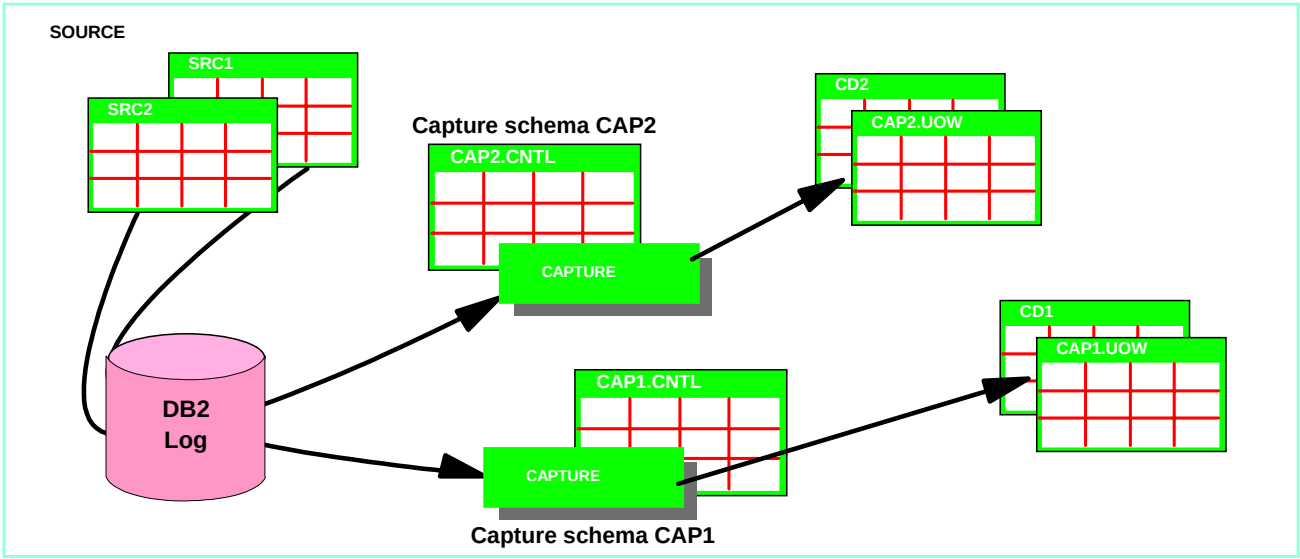
- 省略時設定: commit_interval = 30, prune_interval = 300
- パラメーターがCAPPARMS表の指定が以下にように設定:
  commit_interval = 45, prune_interval is NULL
- Captureを指定なしで起動
  - Capture uses: commit_interval = 45, prune_interval = 300
- CHGPARMS commandを発行: commit_interval = 20
  - Capture uses: commit_interval = 20, prune_interval = 300
- Captureを停止し再起動 (no startup options)
  - Capture uses: commit_interval = 45, prune_interval = 300
- CAPPARMS表を更新: commit_interval = 15, prune_interval = 600
  - Capture uses: commit_interval = 45, prune_interval = 300
- Captureを停止し再起動 (commit interval = 10) :
  - Capture uses: commit_interval = 10, prune_interval = 600

解説:

■このページはブランクです。

Captureパフォーマンス情報

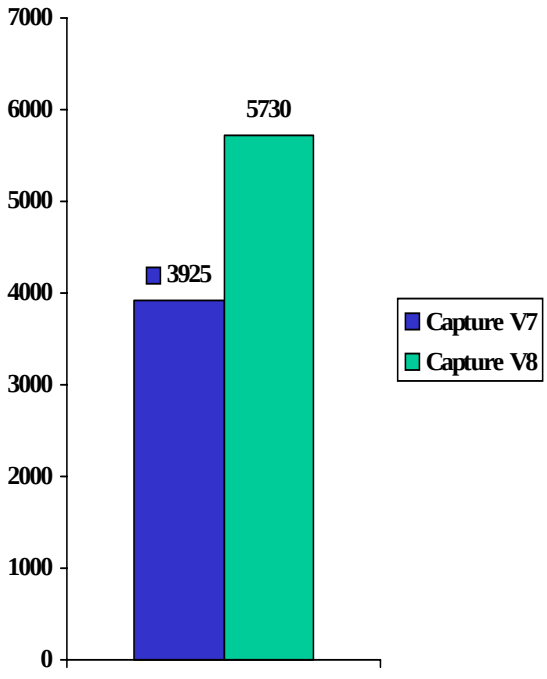
- スループット Capture V8 vs Capture V7
- スケーラビリティ
- 複数 Captures: Performance measurement



解説:

■測定結果は開発元から提供されたパフォーマンス情報
Capture V8 vs Capture V7 on the Same Machine

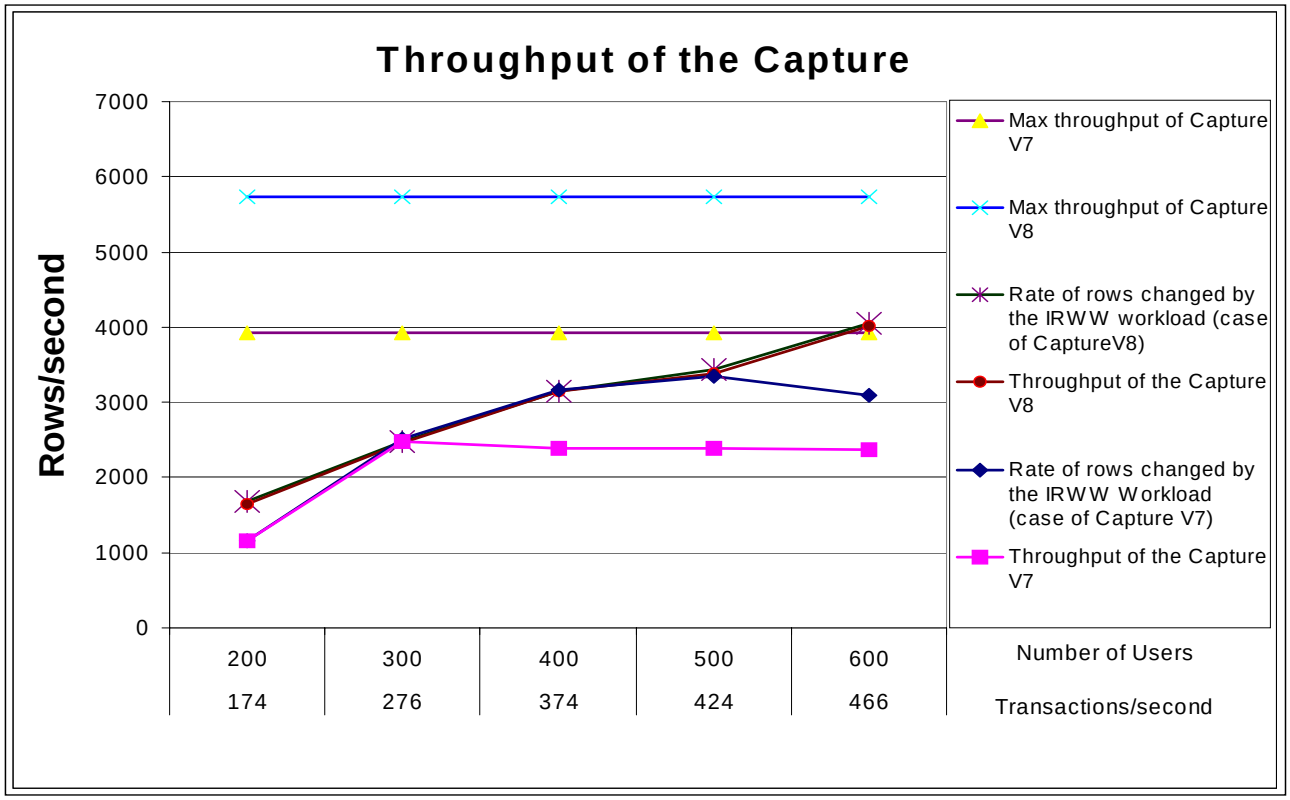
	Throughput (rows/sec)
Capture V8	5730
Capture V7	3925



- Capture V8
  - ►Achieve 5730 rows/sec
  - ►46% improvement

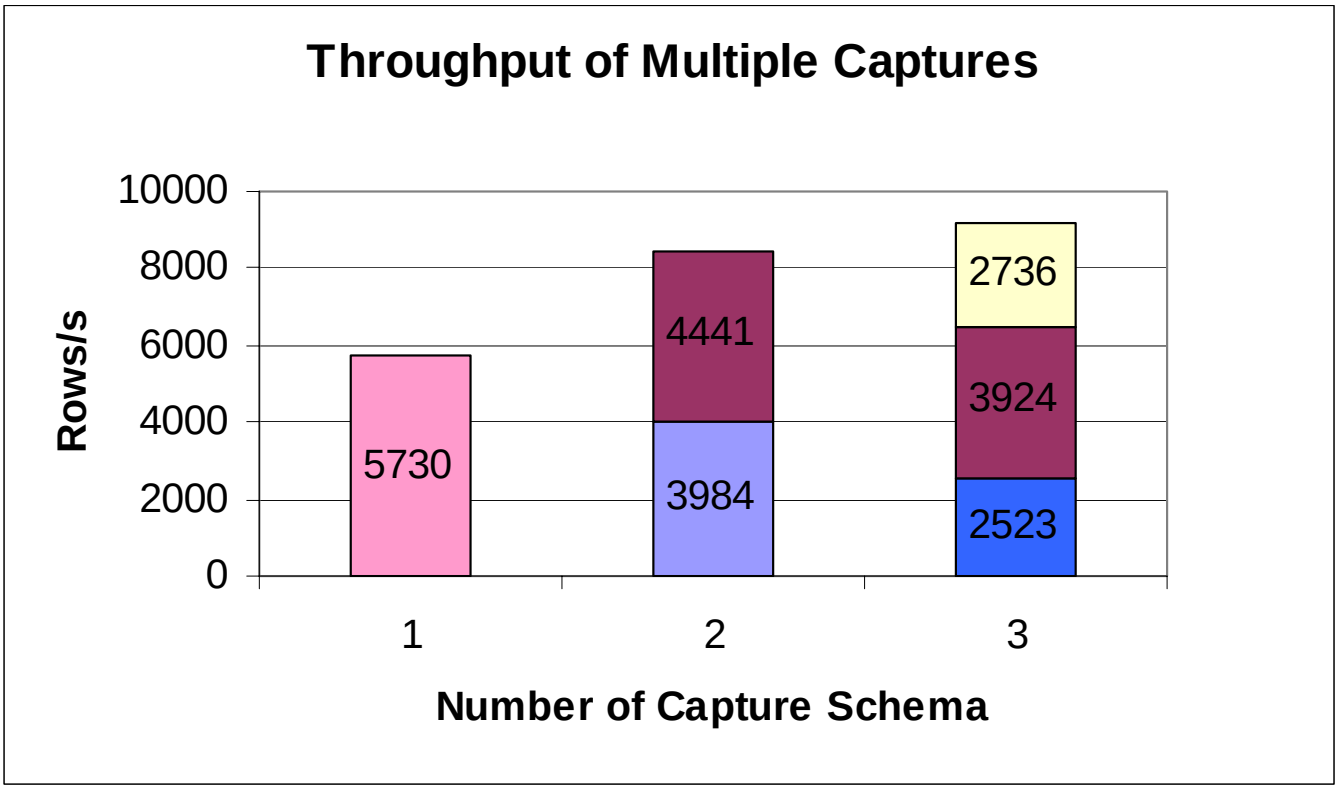
解説:

■測定結果は開発元から提供されたパフォーマンス情報
Scalability



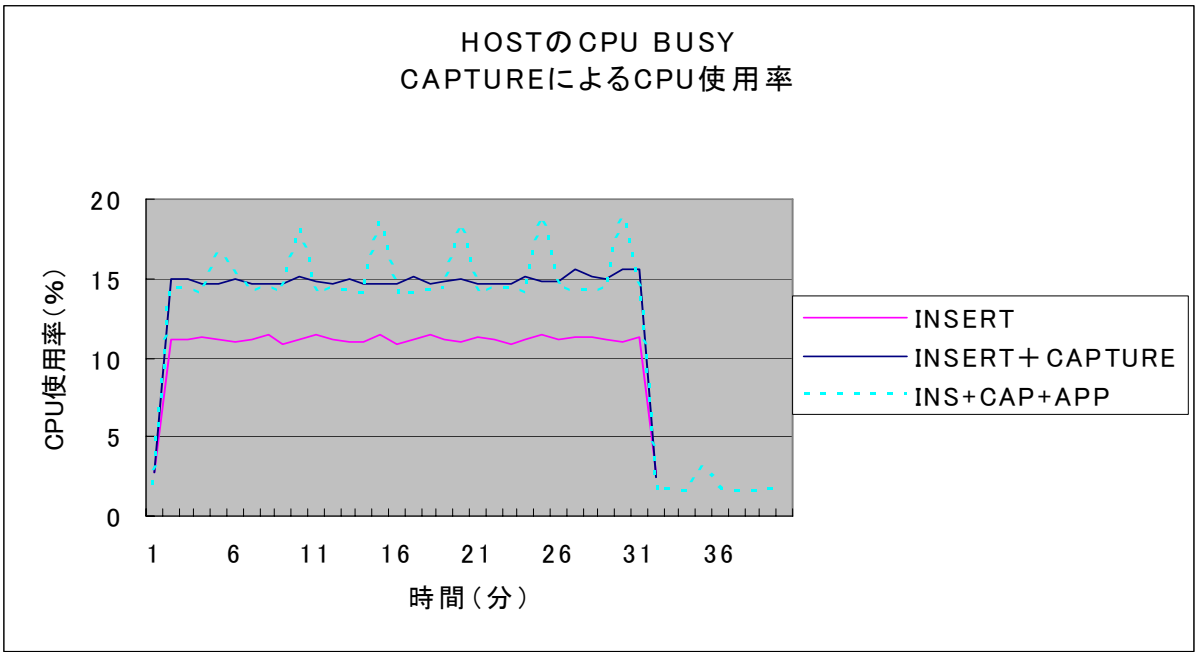
解説:

■測定結果は開発元から提供されたパフォーマンス情報
Throughput of One Capture vs Multiple Captures



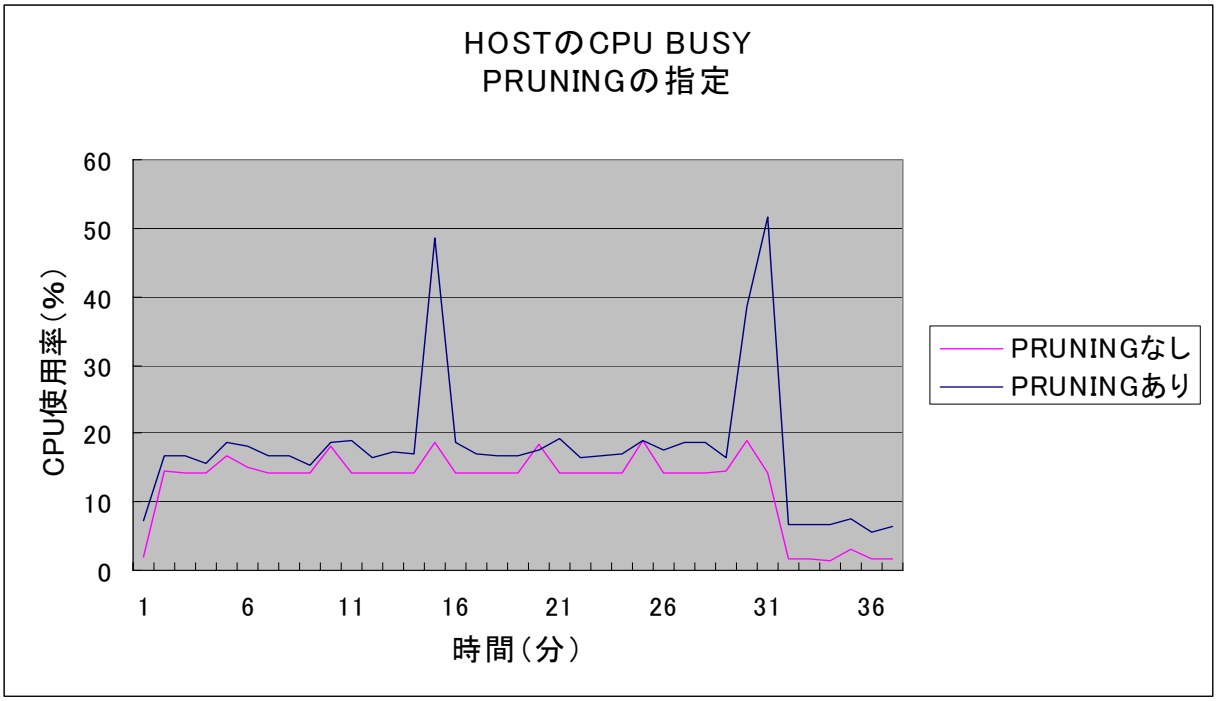
解説:

■測定結果はMKTest環境で測定されたパフォーマンス情報
ホストの表に INSERT のみを行った場合、同時に CAPTURE を行った場合、さらに APPLY も行った場合のホストの CPU 使用率を表したものです。
7500 レコード／分 の処理を行っている環境では CAPTURE を行うことで 4% ほど CPU 使用率が上がります。
Apply Intervalは5分でDB2 for z/OSからUDB V8のApplyを使用してReplicationする形態で測定



解説:

■測定結果はMKTest環境で測定されたパフォーマンス情報
PRUNING を行った場合と行わない場合のホストの CPU 使用率の変化を表したものです。
今回のテストでは PRUNING の INTERVAL を 900 秒に設定しています。

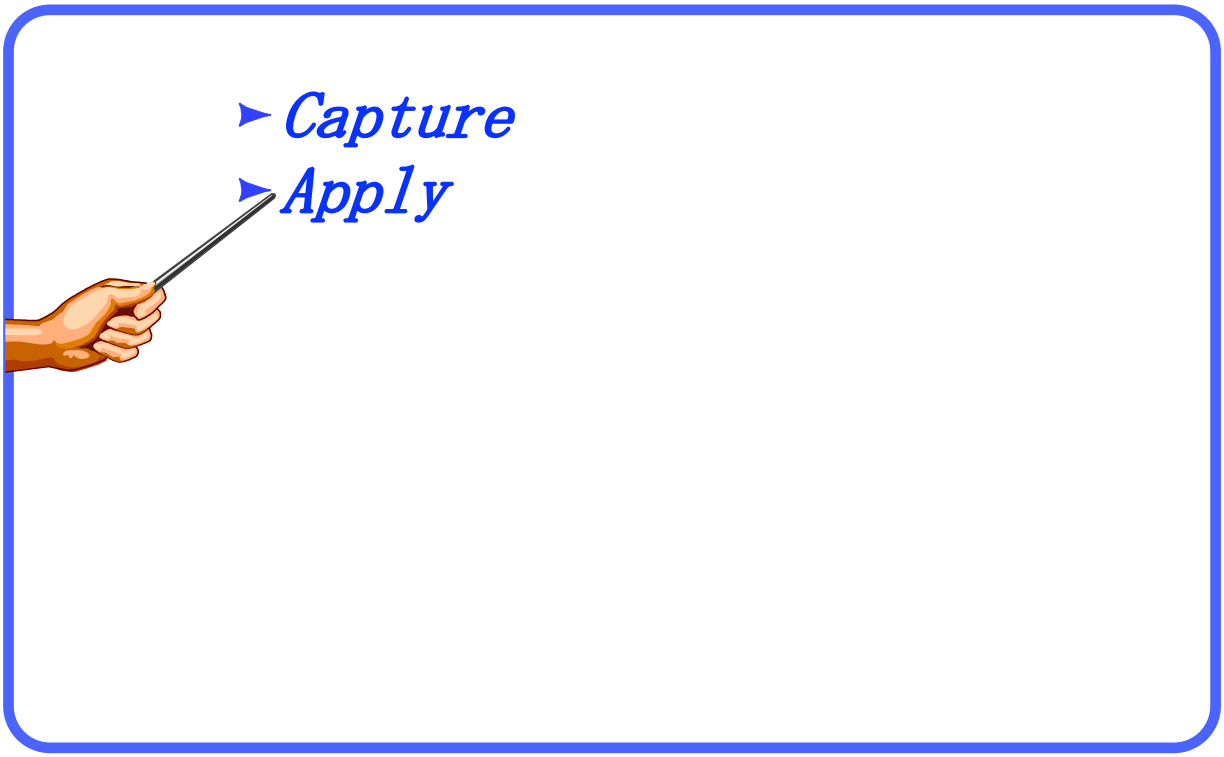


Questions ?

解説:

■このページはブランクです。

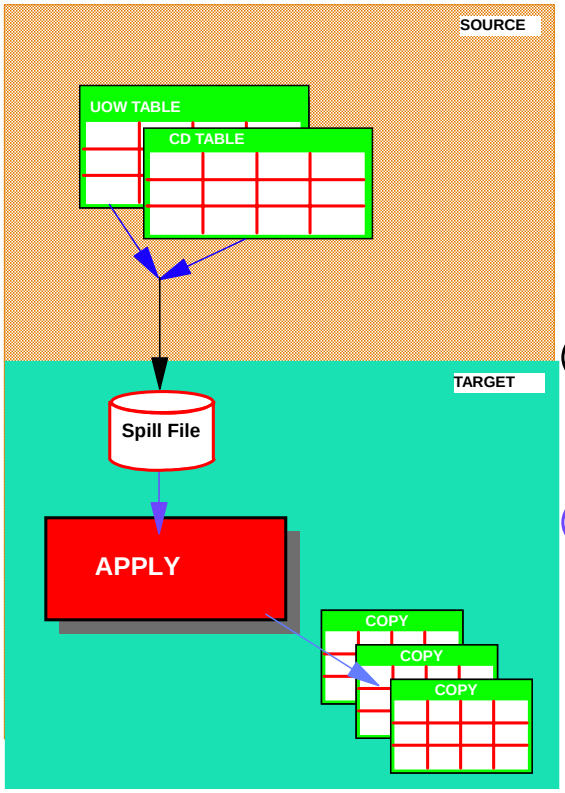
Agenda



解説:

■このページはブランクです。

APPLYメインタスク



1. コミット済み変更DataをCD(またはUOW表をJOINして)から抽出する。(動的SQLを使用している)
(SOURCE側でのCD,UOWのJOIN実行はTargetがUSER_COPY表の場合はせず、CD表のみから抽出)

2. 抽出されたデータをTarget側のSPILL Fileへ書きだす
(Data Tranfer)

3. SPILL FileからSQL文を生成、発行しTarget側へ変更を反映

解説:

■APPLYプログラムとは
Applyプログラムの役割は、ソース表のデータへの変更をターゲット表に複製することです。
Apply プログラムは、CD 表に保管された変更済みデータを読み取り、その変更を
ターゲット表に適用します。 また、Apply プログラムは、ソース表全体をターゲット表にコピーする
(全リフレッシュ・コピー と呼ばれる) ときには、ソース表から直接にデータを読み取ります。

Apply プログラムは、一般にターゲット・サーバーで実行されますが、ソース、制御、およびターゲット・サーバーと接続できる、ネットワーク内の任意のサーバーで実行できます。 複数の Apply プログラム・インスタンスを、同じかまたは異なるサーバーで実行することができます。

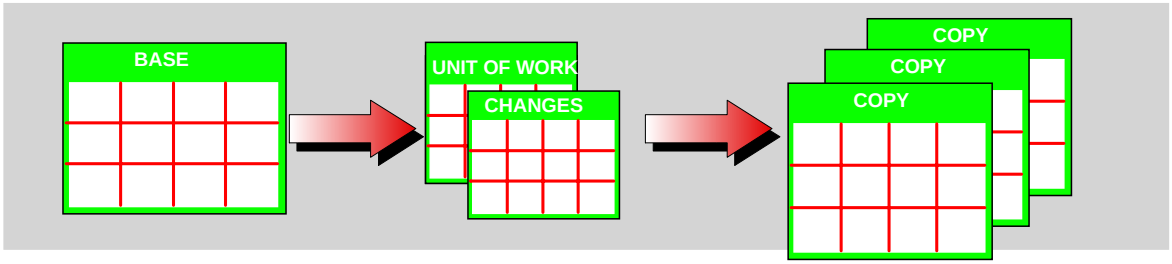
V8よりターゲット表がUSER_COPY表でSUBS_MEMBR.JOIN_UOW_CD='Y'でない限りUOW表とのJOINは行わない

■APPLYは差分収集時 REGISTER表のCD_NEW_SYNCHPOINT > SUBS_SET表のSYNCHPOINTであればCD表を読み込みこみ、不要なアクセスを避けるようにデザインされています

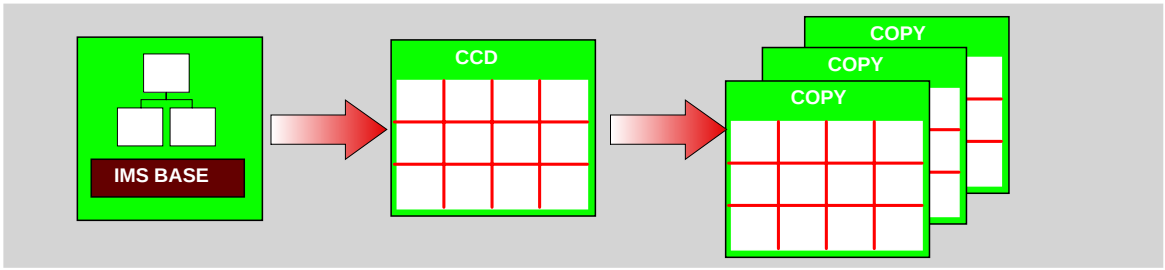
■読み込まれたデータをターゲット側のワークファイル(SPILL FILE)へ書き出す

■SPILL FILEから情報を読み込みTARGET表へSQL UPDATE/INSERT/DELETEを実行し、変更内容を反映する

APPLYステージング



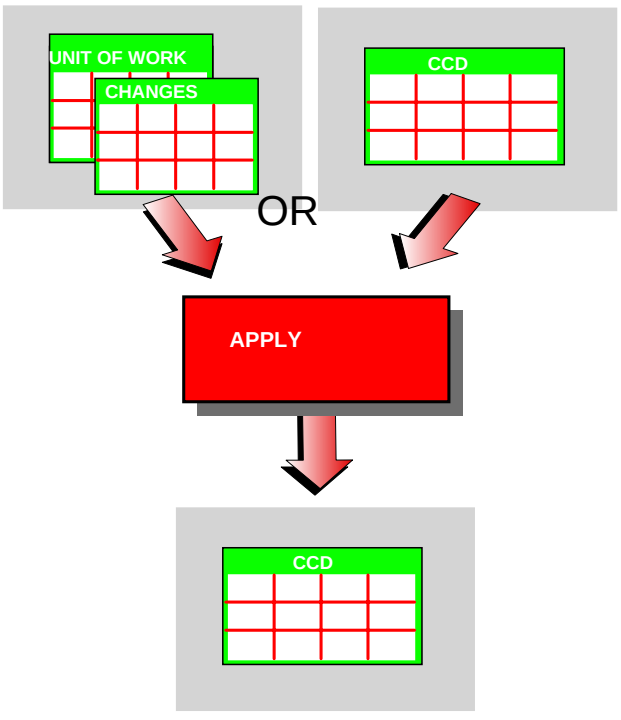
・CaptureがCD表へステージングしCOPY表へApplyが書き出す

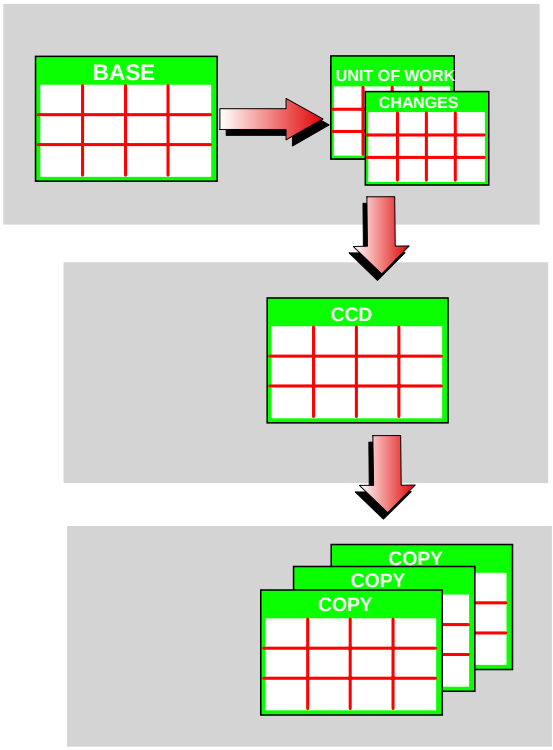


・DPROP-NRがCCD表へステージングしCOPY表へApplyが書き出す

解説:

■CCD表はApplyプログラムのOUTPUT/INPUTになる





解説：

■ソース表レイアウト
■Fullrefresh後のCCD表レイアウト(COMPLETE=Y,CONDENSED=Y)

INTENTSEQ,COMMITSEQの初期値はソースサーバー上のREGISTER表にあるGLOBALレコードのSYNCHPOINT
LOGOMERKERの初期値はソースサーバー上のREGISTER表にあるGLOBALレコードのSYNCHTIME

```
------------------------------入力コマンド------
SELECT * FROM TEST_SOURCE

------------------------------------------------

ROW_NUMBER TEST_DATA DESCRIPTION
---------- --------- --------------------
         1         1 this is row1
         2         2 this is row2
         3         3 this is row3
         4         4 this is row4
         5         5 this is row5
         6         6 this is row6

  6 レコードが選択されました。
```

```
------------------------------入力コマンド--------------------------------
select * from test_source_ccd
--------------------------------------------------------------------------

IBMSNAP_INTENTSEQ          IBMSNAP_OPERATION IBMSNAP_COMMITSEQ          IBMSNAP_LOGMARKER          ROW_NUMBER TEST_DATA DESCRIPTION
-------------------------- ----------------- -------------------------- -------------------------- ---------- --------- --------------------
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          1         1 this is row1
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          2         2 this is row2
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          3         3 this is row3
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          4         4 this is row4
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          5         5 this is row5
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          6         6 this is row6

  6 レコードが選択されました。
```


解説：

■ソース表の更新

```
delete from test_source where row_number=6;
insert into test_source values (99,99, 'This is inserted');
update test_source set test_data=test_data+100 where row_number=1;
```

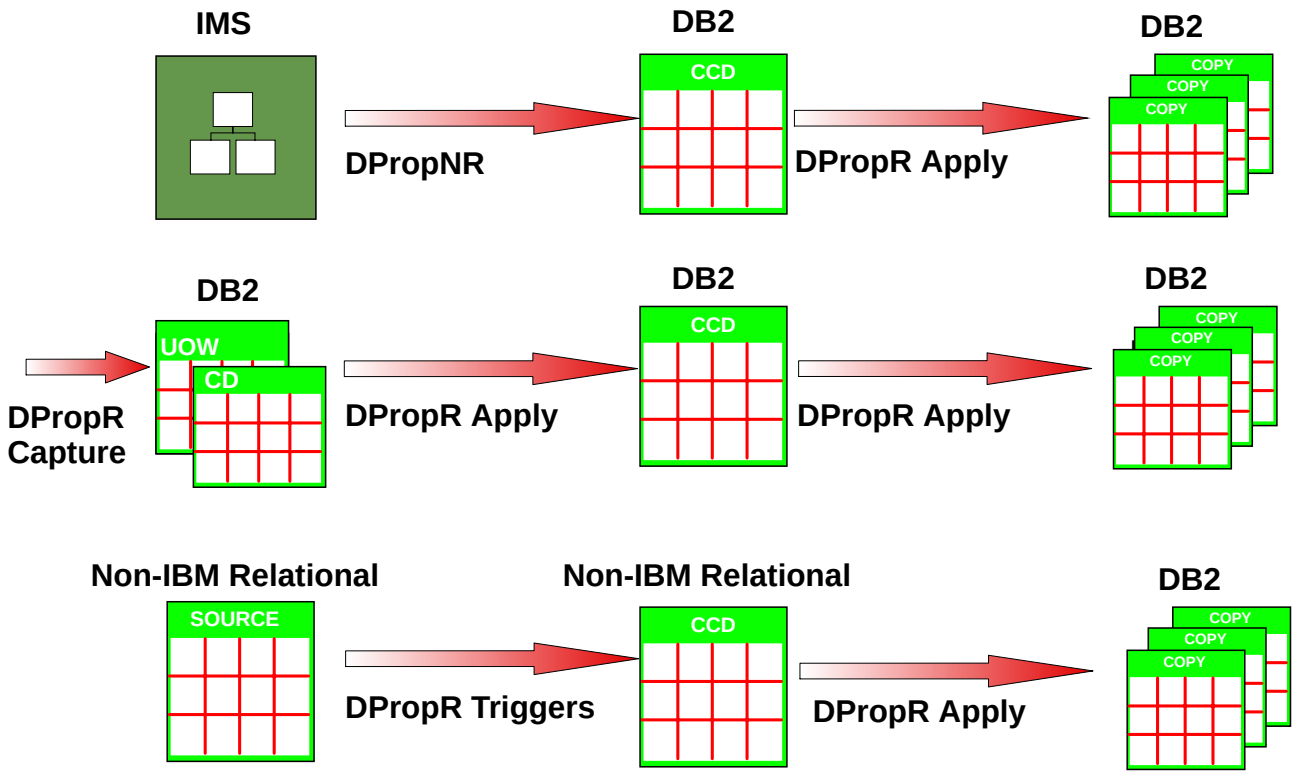
■更新後のソース表

```
------------------------------入力コマンド------------------------------
SELECT * FROM TEST_SOURCE
------------------------------------------------------------------------
ROW_NUMBER TEST_DATA DESCRIPTION
---------- --------- --------------------
         1       101 this is row1
         2         2 this is row2
         3         3 this is row3
         4         4 this is row4
         5         5 this is row5
        99        99 This is inserted
  6 レコードが選択されました。
```

■変更反映後のCCD表

```
------------------------------入力コマンド--------------------------------
select * from test_source_ccd
--------------------------------------------------------------------------

IBMSNAP_INTENTSEQ          IBMSNAP_OPERATION IBMSNAP_COMMITSEQ          IBMSNAP_LOGMARKER          ROW_NUMBER TEST_DATA DESCRIPTION
-------------------------- ----------------- -------------------------- -------------------------- ---------- --------- --------------------
x'00000000000003136243'    U                 x'000000000000031362BF'    1999-07-11-10.03.51.000000          1       101 this is row1
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          2         2 this is row2
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          3         3 this is row3
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          4         4 this is row4
x'00000000000003133790'    I                 x'00000000000003133790'    1999-07-11-09.55.55.610007          5         5 this is row5
x'00000000000003136171'    D                 x'000000000000031361B9'    1999-07-11-10.03.51.000000          6         6 this is row6
x'000000000000031361DA'    I                 x'00000000000003136222'    1999-07-11-10.03.51.000000         99        99 This is inserted

  7 レコードが選択されました。
```

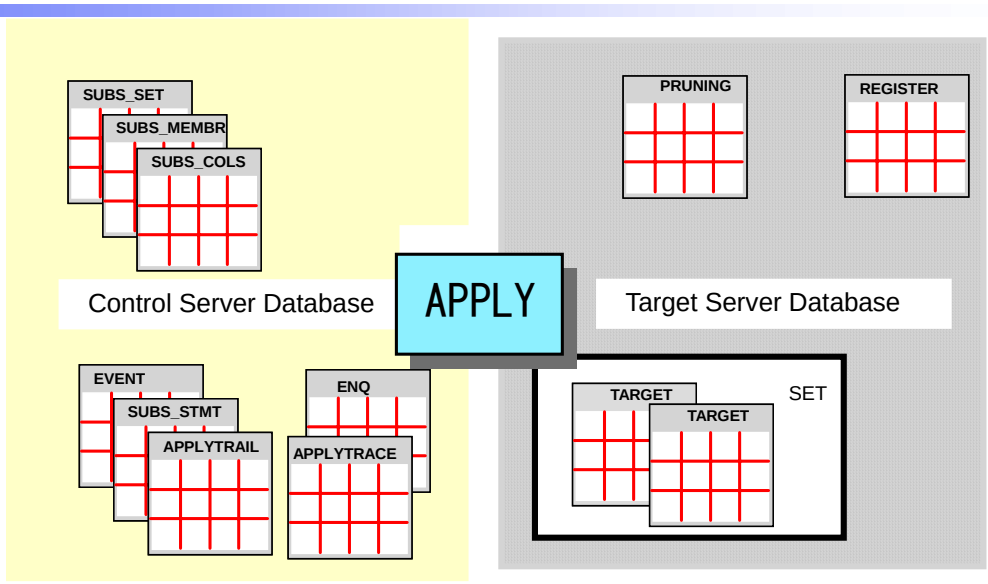
CCD表の種類と役割



解説:

■CCD表の作成者
　-DPROPR-NR
　-CAPTURE/APPLY
　-外部プログラム
　の場合が考えられる
■LOCAL CCD vs REMOTE CCD
　CCD表があるLocationがソースサーバーの場合Local CCD表
　CCD表があるLocationがソースサーバー以外のAPPLYがアクセスできるLocationにある場合 Remote CCD表
■COMPLETE CCD vs NON-COMPLETE CCD
　そのCCD表がターゲット表に対して全件置き換え対象表である場合、COMPLETE CCD表
■CONDENSED CCD vs NON-CONDENSED CCD
　KEYに対して最新の１行のみ含まれる場合CONDENSED CCD表
　変更履歴を持つ場合はNON-CONDENSED CCD表
■INTERNAL CCD(内部CCD表) vs EXTERNAL CCD
　内部 CCD 表は、レプリケーション・ソースの CD 表の代替として存在するターゲット表のタイプです。
　ターゲット表としての内部 CCD 表と一緒にサブスクリプション・セット・メンバーをいったん作成してしまうと、同一のレプリケーション・ソースにサブスクライブする他のすべてのターゲット表は、レプリケーション・ソースの CD 表の代わりに内部 CCD 表から変更データを受け取ります。
　すなわち、アプライ・プログラムは内部 CCD 表から変更データを読み取り、そのデータをターゲット表に複製します。 内部 CCD 表を移植できるのはアプライ・プログラムまたはキャプチャー・トリガーのみであり、外部のアプリケーションからは不可能です。
　内部 CCD 表は常に不完全ですが、コンデンスまたは非コンデンスにすることはできます。 また、レプリケーション・ソースにつき内部 CCD 表を 1 つだけ定義することができます。
■CCD表から変更収集をするためにはREGISTER表のSYNCHPOINT値が上限となる。もしREGISTER表SYNCHPOINTが更新されていない場合はAPPLYはCCD表からデータを反映できない(ISOLATION UR使用の為)

APPLYコントロールテーブル



•SUBS_SET	ASN.IBMSNAP_SUBS_SET
•SUBS_MEMBR	ASN.IBMSNAP_SUBS_MEMBR
•SUBS_COLS	ASN.IBMSNAP_SUBS_COLS
•EVENT	ASN.IBMSNAP_SUBS_EVENT
•SUBS_STMT	ASN.IBMSNAP_SUBS_STMTS
•TRAIL	ASN.IBMSNAP_APPLYTRAIL
•TRACE	ASN.IBMSNAP_APPLYTRACE
•ENQ	ASN.IBMSNAP_APPENQ

解説:

■APPLYが使用するテーブルは下記のようになる
APPLY側はV7と比較して大きくは変更されていない

APPLYコントロール表	Description
ASN.IBMSNAP_APPENQ	1 APPLY QUALIFIERに対し1つのAPPLYのみ稼動させる為に使用する
ASN.IBMSNAP_APPLYTRACE	APPLYからのメッセージの保持
ASN.IBMSNAP_APPLYTRAIL	APPLYの監査情報(REPLICATIONログ)
ASN.IBMSNAP_SUBS_SET	APPLYの処理グループ(SET)情報
ASN.IBMSNAP_SUBS_MEMBR	APPLYの処理グループ(MEMBR)情報
ASN.IBMSNAP_SUBS_COLS	APPLYの処理グループ(COL)情報
ASN.IBMSNAP_SUBS_EVENT	APPLYの起動EVENTを制御
ASN.IBMSNAP_SUBS_STMTS	APPLY SET単位の実行SQLまたはストアードプロシージャを指定

APPLY各コントロールテーブルの情報

■ASN.IBMSNAP_SUBS_SET
ACTIVATE (0:Deactivated,1:Active)
APPLY_QUAL,SET_NAME,WHOS_ON_FIRST(F:First,S:Second)
SOURCE_SERVER,TARGET_SERVER,SOURCE_ALIAS,TARGET_ALIAS
STATUS (-1:failure,0:stable,1:in-progress,2:used,16:SQLERRCONITNUEで許容されたERROR,18:SQLERRCONITNU & MAX_SYNCH_MINUTES指定)
MAX_SYNC_MINUTES,LASTRUN,LASTSUCCESS,REFRESH_TIMING(R:E:B),SLEEP_MINUTES,
EVENT_NAME,SYNCHPOINT,SYNCHTIME
MAX_SYNCH_MINUTES,AUX_STMTS
SET_TYPE(R,U),COMMIT_COUNT　R..Read-Only, U..UpdateAnywheer
CAPTURE_SCHEMA,TGT_CAPTURE_SCHEMA,FEDERATED_SRC_SRVR,FEDERATED_TGT_SRVR

■SUBS_SET定義例

```
INSERT INTO ASN.IBMSNAP_SUBS_SET (
 APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, SET_TYPE,  ACTIVATE, SOURCE_SERVER, SOURCE_ALIAS,
 TARGET_SERVER, TARGET_ALIAS, STATUS,  REFRESH_TYPE, SLEEP_MINUTES, EVENT_NAME,
 MAX_SYNCH_MINUTES, AUX_STMTS, ARCH_LEVEL,  LASTRUN, LASTSUCCESS,
 CAPTURE_SCHEMA, TGT_CAPTURE_SCHEMA, OPTION_FLAGS,
 FEDERATED_SRC_SRVR, FEDERATED_TGT_SRVR, COMMIT_COUNT, JRN_LIB, JRN_NAME
 ) VALUES (
'QUAL1', 'SET1', 'S', 'R', 1, 'V8DB', 'V8DB', 'SAMPLE', 'SAMPLE', 0, 'R', 1, null,
 20, 0, '0801', '2002-07-08-16.39.14.0', null, null, null, null, null, null
 );
```

一分ごとにAPPLY_QUAL='QUAL1'のSUBSCRIPTIONを実行する。データベースV8DB -->SAMPLE DBへのREPLICATIONを定義している。SQL-BEFORE/AFTERは定義されていない。(AUX_STMTS=0)

解説:

■ASN.IBMSNAP_SUBS_SET

```
CREATE TABLE ASN.IBMSNAP_SUBS_SET(
APPLY_QUAL                      CHAR( 18) NOT NULL,
SET_NAME                        CHAR( 18) NOT NULL,
SET_TYPE                        CHAR(  1) NOT NULL,
WHOS_ON_FIRST                   CHAR(  1) NOT NULL,
ACTIVATE                        SMALLINT NOT NULL,
SOURCE_SERVER                   CHAR( 18) NOT NULL,
SOURCE_ALIAS                    CHAR(  8),
TARGET_SERVER                   CHAR( 18) NOT NULL,
TARGET_ALIAS                    CHAR(  8),
STATUS                          SMALLINT NOT NULL,
LASTRUN                         TIMESTAMP NOT NULL,
REFRESH_TYPE                    CHAR( 1) NOT NULL,
SLEEP_MINUTES                   INT,
EVENT_NAME                      CHAR( 18),
LASTSUCCESS                     TIMESTAMP,
SYNCHPOINT                      CHAR( 10) FOR BIT DATA,
SYNCHTIME                       TIMESTAMP,
CAPTURE_SCHEMA                  VARCHAR( 30) NOT NULL,
TGT_CAPTURE_SCHEMA              VARCHAR( 30),
FEDERATED_SRC_SRVR              VARCHAR( 18),
FEDERATED_TGT_SRVR              VARCHAR( 18),
JRN_LIB                         CHAR( 10),
JRN_NAME                        CHAR( 10),
OPTION_FLAGS                    CHAR(  4) NOT NULL,
COMMIT_COUNT                    SMALLINT,
MAX_SYNCH_MINUTES               SMALLINT,
AUX_STMTS                       SMALLINT NOT NULL,
ARCH_LEVEL                      CHAR( 4) NOT NULL);
CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_SETX
ON ASN.IBMSNAP_SUBS_SET(
APPLY_QUAL                      ASC,
SET_NAME                        ASC,
WHOS_ON_FIRST                   ASC);
```

APPLY各コントロールテーブルの情報

■**ASN.IBMSNAP_SUBS_MEMBR**

APPLY_QUAL,SET_NAME,WHOS_ON_FIRST
SOURCE_OWNER,SOURCE_TABLE
TARGET_OWNER,TARGET_TABLE
SOURCE_VIEW_QUAL
TARGET_CONDENSED(Y:N),TARGET_COMPLETE(Y:N)
TARGET_STRUCTURE(1:3:4:5:6:7:8)
PREDICATES
MEMBER_STATE(N,L,S) N..New,L..Loaded,S..Synchronized
TARGET_KEY_CHG(Y,N)
JOIN_UOW_CD,UOW_CD_PREDICATES
LOADX_TYPE,LOAD_SRC_N_OWNER,LOAD_SRC_N_TABLE

■SUBS_MEMBR定義例

```
INSERT INTO ASN.IBMSNAP_SUBS_MEMBR (
 APPLY_QUAL, SET_NAME, WHOS_ON_FIRST,
 SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL,
 TARGET_OWNER, TARGET_TABLE, TARGET_STRUCTURE,
 TARGET_CONDENSED, TARGET_COMPLETE,
 PREDICATES, UOW_CD_PREDICATES, JOIN_UOW_CD,
 MEMBER_STATE,TARGET_KEY_CHG
 ) VALUES (
 'QUAL1', 'SET1', 'S', 'AZUMA', 'TEST_SOURCE',
 0, 'AZUMA', 'TGTEST_SOURCE',
 8, 'Y', 'Y', null, null, null, 'N', 'N'
 );
```

TEST_SOURCE-->TGTEST_SOURCE表へのReplication定義。TARGET表のタイプはUSER COPY表。WHERE条件なし。

解説:

■ASN.IBMSNAP_SUBS_MEMBR

```
CREATE TABLE ASN.IBMSNAP_SUBS_MEMBR(
APPLY_QUAL                      CHAR( 18) NOT NULL,
SET_NAME                        CHAR( 18) NOT NULL,
WHOS_ON_FIRST                   CHAR(  1) NOT NULL,
SOURCE_OWNER                    VARCHAR( 30) NOT NULL,
SOURCE_TABLE                    VARCHAR(128) NOT NULL,
SOURCE_VIEW_QUAL                SMALLINT NOT NULL,
TARGET_OWNER                    VARCHAR(30) NOT NULL,
TARGET_TABLE                    VARCHAR(128) NOT NULL,
TARGET_CONDENSED                CHAR(  1) NOT NULL,
TARGET_COMPLETE                 CHAR(  1) NOT NULL,
TARGET_STRUCTURE                SMALLINT NOT NULL,
PREDICATES                      VARCHAR(1024),
MEMBER_STATE                    CHAR(  1),
TARGET_KEY_CHG                  CHAR(  1) NOT NULL,
JOIN_UOW_CD                     CHAR(  1),
UOW_CD_PREDICATES               VARCHAR(1024),
LOADX_TYPE                      SMALLINT,
LOADX_SRC_N_OWNER               VARCHAR( 30),
LOADX_SRC_N_TABLE               VARCHAR(128));
CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_MEMBXON ASN.IBMSNAP_SUBS_MEMBR(
APPLY_QUAL                      ASC,
SET_NAME                        ASC,
WHOS_ON_FIRST                   ASC,
SOURCE_OWNER                    ASC,
SOURCE_TABLE                    ASC,
SOURCE_VIEW_QUAL                ASC,
TARGET_OWNER                    ASC,
TARGET_TABLE                    ASC);
```

APPLY各コントロールテーブルの情報

■ASN.IBMSNAP_SUBS_COLS
APPLY_QUAL,SET_NAME,WHOS_ON_FIRST
TARGET_OWNER,TARGET_TABLE
COL_TYPE(A:B:C:D:F:L:P:R)
TARGET_NAME(Target Column name)
IS_KEY(Y:N)
COLNO
EXPRESSION(Source Column expression)

■SUBS_COLS定義例

```
INSERT INTO ASN.IBMSNAP_SUBS_COLS (
 APPLY_QUAL, SET_NAME, WHOS_ON_FIRST,
 TARGET_OWNER, TARGET_TABLE, TARGET_NAME,
 COL_TYPE, IS_KEY, COLNO, EXPRESSION
 ) VALUES ( 'QUAL1', 'SET1', 'S', 'AZUMA', 'TGTEST_SOURCE', 'ROW_NUMBER', 'A', 'Y', 1, 'ROW_NUMBER' );
INSERT INTO ASN.IBMSNAP_SUBS_COLS (
 APPLY_QUAL, SET_NAME, WHOS_ON_FIRST,
 TARGET_OWNER, TARGET_TABLE, TARGET_NAME,
 COL_TYPE, IS_KEY, COLNO, EXPRESSION
 ) VALUES ( 'QUAL1', 'SET1', 'S', 'AZUMA', 'TGTEST_SOURCE', 'TEST_DATA', 'A', 'N', 2, 'TEST_DATA' );
INSERT INTO ASN.IBMSNAP_SUBS_COLS (
 APPLY_QUAL, SET_NAME, WHOS_ON_FIRST,
 TARGET_OWNER, TARGET_TABLE, TARGET_NAME,
 COL_TYPE, IS_KEY, COLNO, EXPRESSION
 ) VALUES ( 'QUAL1', 'SET1', 'S', 'AZUMA', 'TGTEST_SOURCE', 'DESCRIPTION', 'A', 'N', 3, 'DESCRIPTION' );
```

IS_KEY　　-->そのカラムがKEY列かどうか
EXPRESSION -->ソース表のカラム名
TARGET_NAME-->ターゲット表のカラム名、ソース表と一致させる必要はない

解説:

■ASN.IBMSNAP_SUBS_COLS

```
CREATE TABLE ASN.IBMSNAP_SUBS_COLS(
APPLY_QUAL                      CHAR( 18) NOT NULL,
SET_NAME                        CHAR( 18) NOT NULL,
WHOS_ON_FIRST                   CHAR(  1) NOT NULL,
TARGET_OWNER                    VARCHAR(30) NOT NULL,
TARGET_TABLE                    VARCHAR(128) NOT NULL,
COL_TYPE                        CHAR(  1) NOT NULL,
TARGET_NAME                     VARCHAR( 30) NOT NULL,
IS_KEY                          CHAR(  1) NOT NULL,
COLNO                           SMALLINT NOT NULL,
EXPRESSION                      VARCHAR(254) NOT NULL)
;
CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_COLSX
ON ASN.IBMSNAP_SUBS_COLS(
APPLY_QUAL                      ASC,
SET_NAME                        ASC,
WHOS_ON_FIRST                   ASC,
TARGET_OWNER                    ASC,
TARGET_TABLE                    ASC,
TARGET_NAME                     ASC);
```

APPLY各コントロールテーブルの情報

- ASN.IBMSNAP_SUBS_EVENT(Optional)
  EVENT_NAME
  EVENT_TIME
  END_SYNCHPOINT
  END_OF_PERIOD

解説:

■ASN.IBMSNAP_SUBS_EVENT

```
-------------------------------------------------------------------
--    Subscription Events Table              (All IBM platforms  )
-------------------------------------------------------------------

CREATE TABLE ASN.IBMSNAP_SUBS_EVENT(
EVENT_NAME                      CHAR( 18) NOT NULL,
EVENT_TIME                      TIMESTAMP NOT NULL,
END_SYNCHPOINT                  CHAR( 10) FOR BIT DATA,
END_OF_PERIOD                   TIMESTAMP)
;
CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_EVENX
ON ASN.IBMSNAP_SUBS_EVENT(
EVENT_NAME                      ASC,
EVENT_TIME                      ASC);
```

APPLY各コントロールテーブルの情報

■ASN.IBMSNAP_SUBS_STMTS(Optional)
APPLY_QUAL,SET_NAME,WHOS_ON_FIRST
BEFORE_OR_AFTER(A:B:S:G:X)
STMT_NUMBER
EI_OR_CALL(E:C)
SQL_STMT
ACCEPT_SQLSTATES

解説:

■ASN.IBMSNAP_SUBS_STMTS

```
-------------------------------------------------------------------
--    Subscription Statements Table          (All IBM platforms  )
-------------------------------------------------------------------

CREATE TABLE ASN.IBMSNAP_SUBS_STMTS(
APPLY_QUAL                      CHAR( 18) NOT NULL,
SET_NAME                        CHAR( 18) NOT NULL,
WHOS_ON_FIRST                   CHAR(  1) NOT NULL,
BEFORE_OR_AFTER                 CHAR(  1) NOT NULL,
STMT_NUMBER                     SMALLINT NOT NULL,
EI_OR_CALL                      CHAR(  1) NOT NULL,
SQL_STMT                        VARCHAR(1024),
ACCEPT_SQLSTATES                VARCHAR( 50))
;
CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_STMTX
ON ASN.IBMSNAP_SUBS_STMTS(
APPLY_QUAL                      ASC,
SET_NAME                        ASC,
WHOS_ON_FIRST                   ASC,
BEFORE_OR_AFTER                 ASC,
STMT_NUMBER                     ASC)
;
```

APPLY各コントロールテーブルの情報

- ASN.IBMSNAP_APPLYTRAIL
  APPLY_QUAL,SET_NAME,WHOS_ON_FIRST
  ASNLOAD
  MASS_DELETE,
  SET_INSERTED,SET_DELETED,SET_UPDATED,SET_REWORKED,SET_REJECTED_TRXS
  STATUS
  LASTRUN,LASTSUCCESS
  SYNCHPOINT,SYNCHTIME
  SOURCE_OWNER,SOURCE_TABLE
  TARGET_OWNER,TARGET_TABLE
  SOURCE_VIEW_QUAL
  SQLCODE,SQLSTATE,SQLERRP,APPERRM
  CAPTURE_SCHEMA,TGT_CAPTURE_SCHEMA,
  FEDERATED_SRC_SRVR,FEDERATED_TGT_SRVR
  JRN_NAME,JRN_LIB
  COMMIT_COUNT
  OPTION_FLAGS
  EVENT_NAME
  SOURCE_CONN_TIME,ENDTIME

解説:

■ASN.IBMSNAP_APPLYTRAIL

```
CREATE TABLE ASN.IBMSNAP_APPLYTRAIL(
APPLY_QUAL                      CHAR( 18) NOT NULL,
SET_NAME                        CHAR( 18) NOT NULL,
SET_TYPE                        CHAR(  1) NOT NULL,
WHOS_ON_FIRST                   CHAR(  1) NOT NULL,
ASNLOAD                         CHAR(  1),
FULL_REFRESH                    CHAR(  1),
EFFECTIVE_MEMBERS               INT,
SET_INSERTED                    INT NOT NULL,
SET_DELETED                     INT NOT NULL,
SET_UPDATED                     INT NOT NULL,
SET_REWORKED                    INT NOT NULL,
SET_REJECTED_TRXS               INT NOT NULL,
STATUS                          SMALLINT NOT NULL,
LASTRUN                         TIMESTAMP NOT NULL,
LASTSUCCESS                     TIMESTAMP,
SYNCHPOINT                      CHAR( 10) FOR BIT DATA,
SYNCHTIME                       TIMESTAMP,
SOURCE_SERVER                   CHAR( 18) NOT NULL,
SOURCE_ALIAS                    CHAR(  8),
SOURCE_OWNER                    VARCHAR(30),
SOURCE_TABLE                    VARCHAR(128),
SOURCE_VIEW_QUAL                SMALLINT,
TARGET_SERVER                   CHAR( 18) NOT NULL,
TARGET_ALIAS                    CHAR(  8),
TARGET_OWNER                    VARCHAR(30) NOT NULL,
TARGET_TABLE                    VARCHAR(128) NOT NULL,
CAPTURE_SCHEMA                  VARCHAR(30) NOT NULL,
TGT_CAPTURE_SCHEMA              VARCHAR(30),
FEDERATED_SRC_SRVR              VARCHAR( 18),
FEDERATED_TGT_SRVR              VARCHAR( 18),
JRN_LIB                         CHAR( 10),
JRN_NAME                        CHAR( 10),
COMMIT_COUNT                    SMALLINT,
OPTION_FLAGS                    CHAR(  4) NOT NULL,
EVENT_NAME                      CHAR( 18),
ENDTIME                         TIMESTAMP NOT NULL WITH DEFAULT ,
SOURCE_CONN_TIME                TIMESTAMP,
SQLSTATE                        CHAR(  5),
SQLCODE                         INT,
SQLERRP                         CHAR(  8),
SQLERRM                         VARCHAR( 70),
APPERRM                         VARCHAR(760))
;
CREATE  INDEX ASN.IBMSNAP_APPLYTRLX
ON ASN.IBMSNAP_APPLYTRAIL(
LASTRUN                         DESC,
APPLY_QUAL                      ASC);
```

APPLY各コントロールテーブルの情報

- ASN.IBMSNAP_APPLYTRACE
  - *APPLY_QUAL*
  - *TRACE_TIME*
  - *OPERATION*
  - *DESCRIPTION*

解説:

■ASN.IBMSNAP_APPLYTRACE

```
-------------------------------------------------------------------
--    Subscription Events Table              (All IBM platforms  )
-------------------------------------------------------------------

CREATE TABLE ASN.IBMSNAP_APPLYTRACE(
APPLY_QUAL                      CHAR(18) NOT NULL,
TRACE_TIME                      TIMESTAMP NOT NULL,
OPERATION                       CHAR(  8) NOT NULL,
DESCRIPTION                     VARCHAR(1024) NOT NULL)
;
CREATE  INDEX ASN.IBMSNAP_APPLYTRACX
ON ASN.IBMSNAP_APPLYTRACE(
APPLY_QUAL                      ASC,
TRACE_TIME                      ASC)
;
```

APPLYターゲット表タイプ

- Source Table(1)
  -複写元Table
- CCD Table(3)
  -Consistent Change Data表
  -Condensed(Y:N),Complete(Y:N)属性をもつ
- Point-in-time table(4)
  -Source表のある時点での複写表,Timestamp列(IBMSNAP_LOGMARKER)あり
- Base Aggregate Table(5)
  -複写元から集約されたデータを含む表
- Change Aggregate Table(6)
  -複写元の変更にもとずいたデータ集約を含む表
- Replica Table(7)
  -更新可能なTarget表
- User Copy Table(8)
  -Source表のある時点での複写表,Timestamp列(IBMSNAP_LOGMARKER)なし

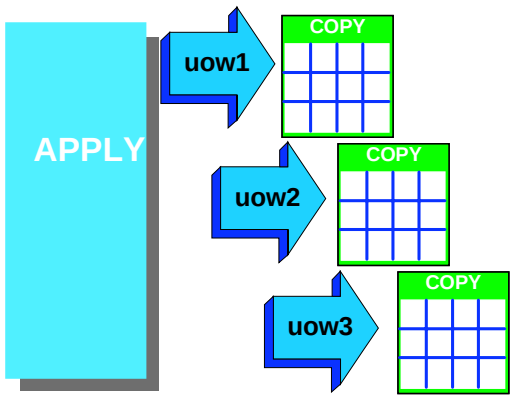
（N）はTarget_Structure番号

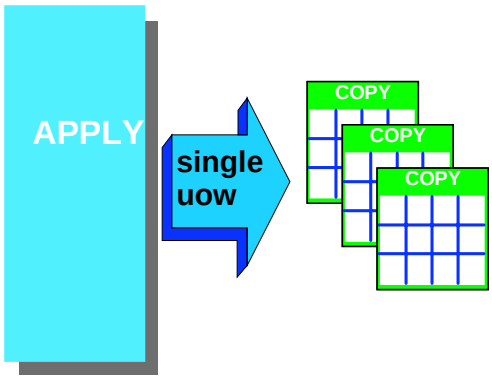
解説:

■ターゲット表の構造は番号で下記のように区分されている。

1 ソース表
3 CCD 表
4 時刻表
5 基礎集約表
6 変更集約表
7 レプリカ
8 ユーザー・コピー

サブスクリプションセット





- 1Member/1Setは変更DataをTargetへ適用する場合３つの異なるLUWで適用していた。

- 3Member/1Setは変更DataをTargetへ適用する同じLUWで適用する。
- 1MemberがFailすればSET全体をRollback

➡ 接続負荷軽減
➡ SPILL Fileの容量増加
➡ Copy Table間での整合性保証

解説:

■SUBSCRIPTION SETとはV5からの新しい概念でReplicationをグループ化すること
-VIEW Replication
-Update Anywhere
を使用する場合は必須であるが、通常のReplicatioでUSER_COPYをターゲット表に使用している場合は1SET-1MEMBERの定義でも十分である。

■複数SET、複数MEMBERの処理順序
Applyが複数SETを処理する必要がある場合はLASTRUN(ORDER BY LASTRUN)の古いSETから処理を行う。
1SET内に複数Memberがある場合は、ターゲット表の名前順に処理を行なう(ORDER BY TARGET_OWNER,TARGET_NAME)。
Applyは起動されると、コントロールサーバーから上記の順序でSubscriptionの情報を読み込む為、ユーザーの指定する順序では処理できない。(Table Mode)

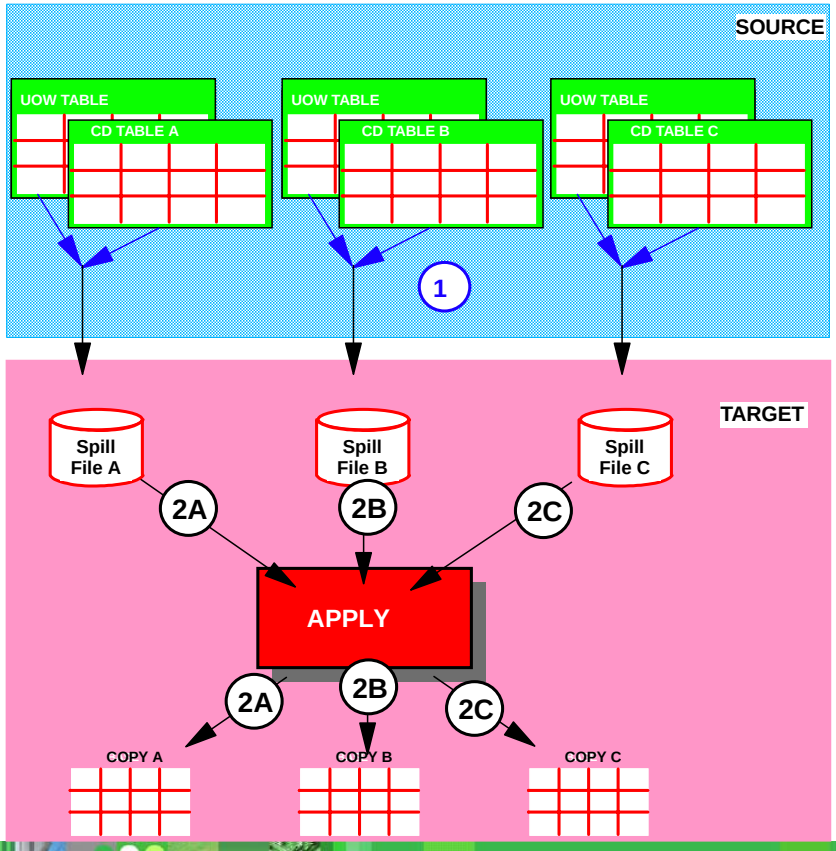
トランザクションモード - COMMIT_Count(n)

■COMMIT_Count(n)

- Apply内の実行順序が変更される
- APPLYは"Table Mode"ではなくreplica表の変更反映のように"Transaction Mode"で実行される
  - ➤RI制約のあるCOPY表のReplicationに有効
  - ➤Target表のタイプがReplica、Point_in_timeまたはUSER_COPYの場合のみ有効
    UDB V8.1 Fixpak2よりターゲットがCCD表でも使用可能
  - ➤Target表内にはLOBまたはDataLinkタイプがあってはならない
- nトランザクション毎にTarget表はCommitされる
  - ➤Active Logの削減
- SET単位の指定が可能
  - ➤V7まではAPPLY始動パラメーター
- Commit_Count(x)の３種類の値
  - ➤ NULL　transaction based replicationは使用しない(省略時)
  - ➤ 0　transaction based replicationは使用されるがApplyはすべてのデータを処理して１度COMMITを発行する
  - ➤ N　transaction based replicationは使用されるApplyはnトランザクション毎にコミットを発行する

解説:

■SET内処理順序 "Table Mode"



SOURCE表A,B,Cに対応するCOPY表A,B,Cが定義されている。
1set内に３つのSubscription Memberが定義され実行時Commit(n)オプションは指定されていない。

Applyは最初３回CD/UOW表のJOINを実行し、おのおのに対するSPILLファイルを作成する

Applyは次にSPILLファイルの内容を

-> 2A
-> 2B
-> 2C

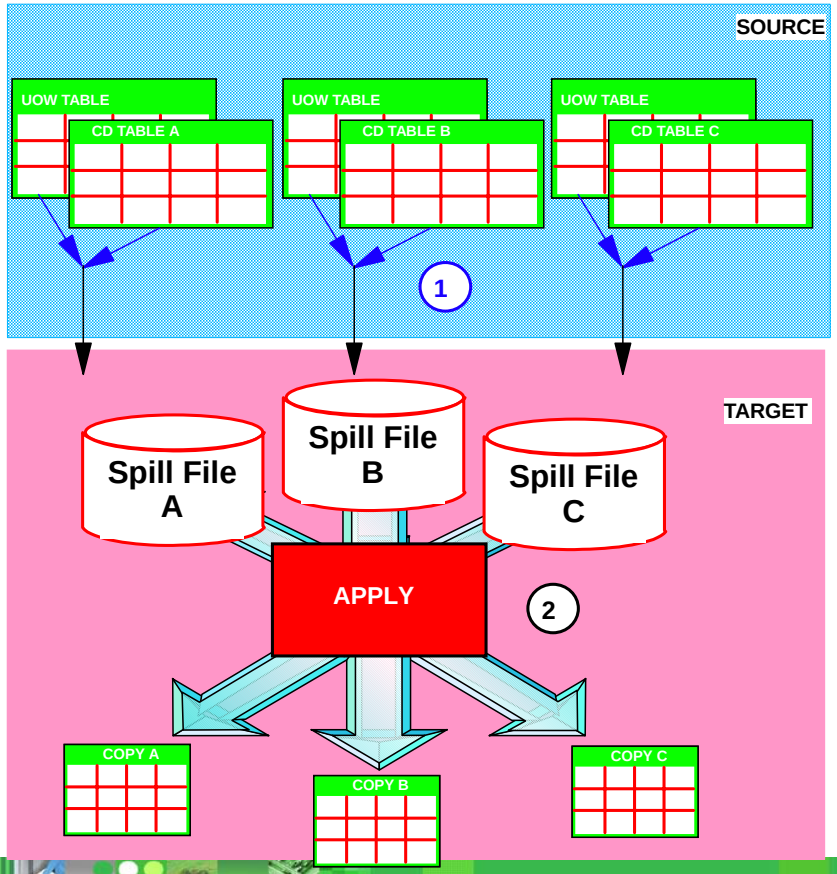
と実行しTARGET表へ反映後

-> COMMIT

を一回実行する。

解説:

■SET内処理順序 "Transaction Mode"



SOURCE表A,B,Cに対応するCOPY表A,B,Cが定義されている。
1set内に３つのSubscription Memberが定義され実行時Commit(n)オプションが指定されている。
Target表のタイプはReplica,USERCOPY,Point_In_Timeのいずれか。

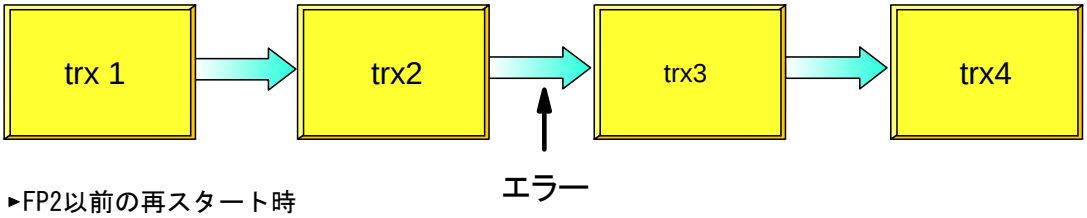
Applyは最初３回CD/UOW表のJOINを実行し、おのおのに対するSPILLファイルを作成する

Applyは次にSPILLファイルの内容を同時にOpenし、Transactionモードの場合はソースサーバーでの発生順序で反映し、N Transaction毎にCOMMITを実行する。

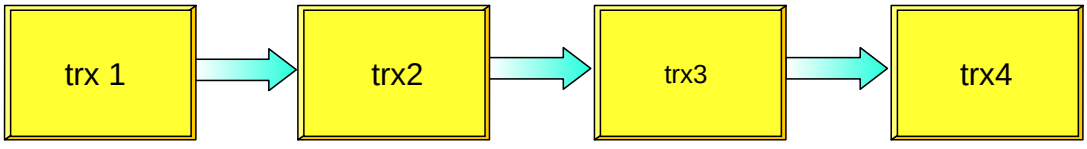
解説:

■TransactionモードのCOPYではApplyはすべてのSpillファイルを開き同時に処理をする。この指定はV7まではApplyの起動パラメーターで指定していたがV8からはSET毎に指定可能
■Replicaがターゲットの場合には常にTransaction Modeで実行される
■Commit_Count(n)が有効時、そのミニUOW内でエラーが発生した場合、STATUS=-1,ACTIVATE=0になる
UDB FP2以前はSYNCHPOINTが更新されず、次回の実行時,初期のSYNCHPOINTから実行され、REWORKが発生する
UDB FP2以降では成功したUOW内のSYNCHPOINTを記録するように変更された為、再起動時にREWORKは発生しない。

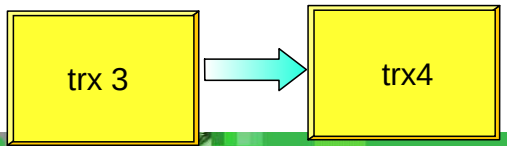
ミニUOWの途中でエラーが発生した場合



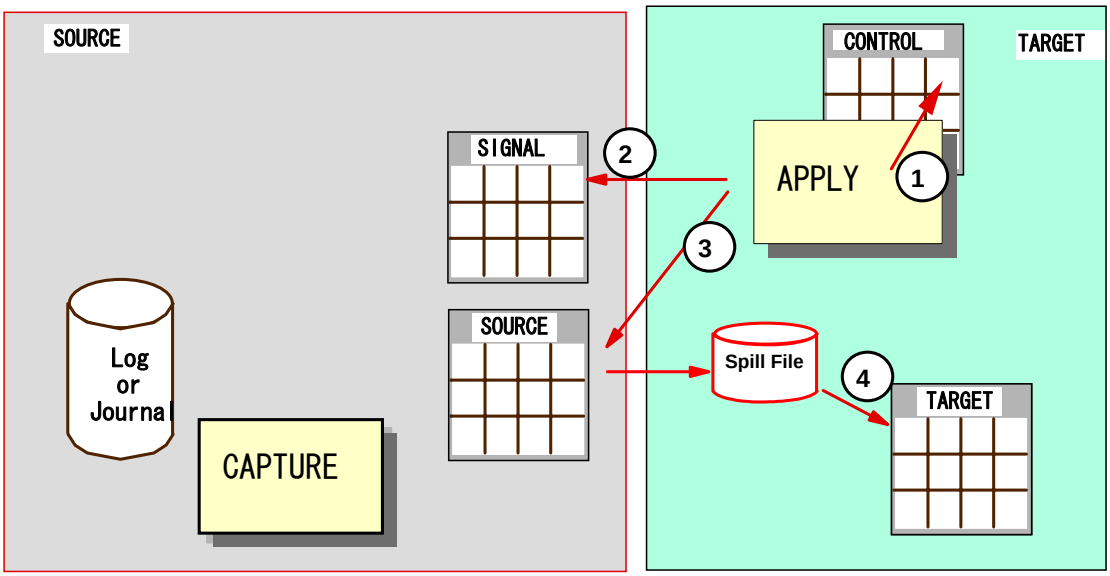
▸FP2以前の再スタート時



▸FP2以降再スタート時



初期フルリフレシュ(Fullrefresh)



- ApplyはSUBS_SET表のSYNCHTIME,SYNCHPOINTがNULLであればFullrefreshを実施する。
- SIGNAL表へのCAPSTARTをInsert
- SOURCE表からDataをSELECT/FETCHしSpill Fileに格納
- Spill FileからTarget表をMASS DELETE/INSERT

解説:

■Apply フルリフレッシュ
Applyプログラムは、フルリフレッシュまたは差分コピーのいずれかによって、ソースからターゲットにデータをコピーします。
フルリフレッシュ・コピーの場合、Applyプログラムはソース表全体をコピーし、それをターゲット表にコピーします、またCaptureプログラムは変更を取り込まないため、関係するCDまたはCCD表はありません。
フルリフレッシュ・コピーONLYを定義する場合は、複製ソース表の定義時にCD表名をNULLに指定することで可能。
また差分コピーの場合も初期のみフルリフレシュを行なう必要があります。
大きい表の場合は、LOADX=Y指定で高速ロード・プログラム(ASNLOAD)を使って初期全リフレッシュ・コピーをシミュレートしたほうがよいかもしれません。

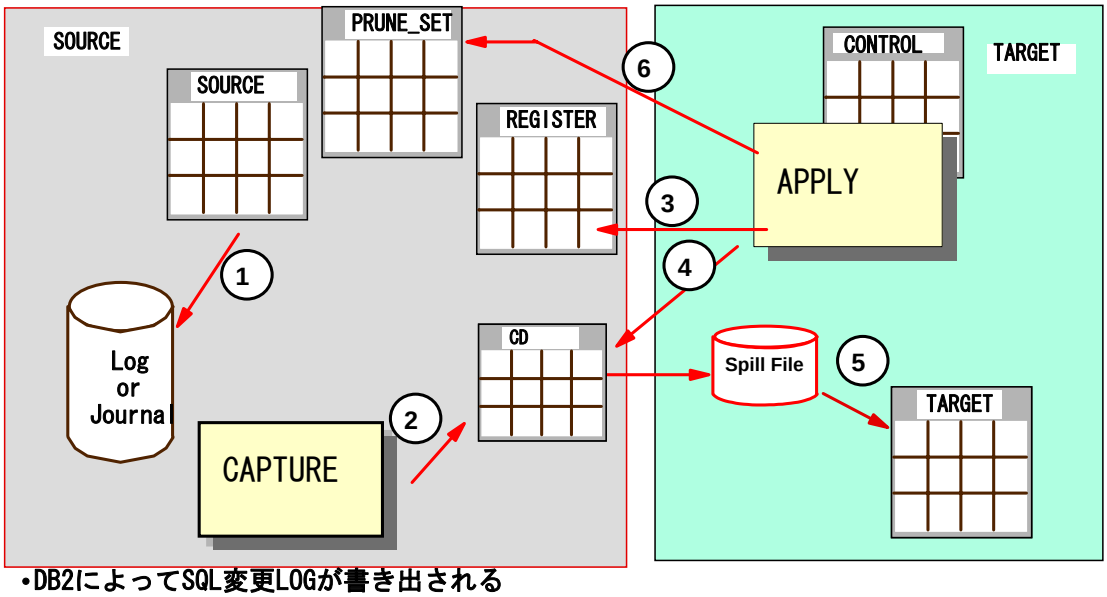
■Apply フルリフレッシュの選択順序
Applyプログラムがターゲットをフルリフレッシュまたは更新するときには、使用可能なソース表のリストから次の順序で選択を行います。

1. 定義済みの複製ソース表に関連した CCD 表
2. 複製サブスクリプションに関連した複製ソース表

CCD 表が完全ではない（つまり、レジスター制御表で CCD_COMPLETE=N である）場合は、CCD表の複製ソース表がフルリフレッシュのソースとして選択されます。(完全属性は、CCD 表の作成時に設定されます。)

■APPLY フルリフレッシュ時の動き
1. CNTL_SERVERに接続し、Subscription情報を読み込む
2. SIGNAL表へCAPSTARTをINSERT
3. SOURCE表を読み込む
4. ターゲットサーバー上のSPILL FILEへ書き込む
5. ターゲット表を全件削除し(MASS DELETE),SPILL FILEからINSERT処理を行なう。これは一つのUOWで行われる

差分COPY



- DB2によってSQL変更LOGが書き出される
- Captureは変更DataをCD表にCommit情報をUOW表に書き出す
- ApplyはREGISTER表のCD_NEW_SYNCHPOINT>SUBS_SET表のSYNCHPOINTの条件であればCD表に変更データありと判断、GLOBAL_RECORDのSYNCHPOINT値をUpper Boundとして抽出
- CD表からSET.SYNCHPOINTより大きくGlobal Synchpointより小さいCommitされたデータのみ抽出
- Spill FileへFetchされたデータを書き出し、Targetへ反映(DETETE,UPDATE,INSERT)
- PRUNE_SET表SYNCHPOINTを更新(プルーニングの為)

解説:

■APPLY差分COPY

差分コピー場合、Apply プログラムは変更されたデータのみを CD 表からターゲット表にコピーします。
Apply プログラムが最初にデータをターゲット表にコピーするとき、または Capture プログラムのコールド・スタート後には、Apply プログラムはフルリフレッシュ・コピーを使用してターゲット表にデータを読み込みます。 ターゲット表にデータが入れられた後は、差分コピーが使用されます。
このタイプのコピーがデフォルトであり、複製ソースの定義時に特に指定しない限り、これが使用されます。

■APPLY差分COPY実行SQL(targetがUSER_COPYの場合)

```
Text     : SELECT  IBMSNAP_OPERATION, IBMSNAP_INTENTSEQ, IBMSNAP_COMMITSEQ, "ROW_NUMBER",
           "TEST_DATA", "DESCRIPTION" FROM  "AZUMA"."CD01" A
            WHERE   A.IBMSNAP_COMMITSEQ > ? <---SUB_SET.SYNCHPOINT値よりも大きいレコード抽出
            AND A.IBMSNAP_COMMITSEQ <= ?    <---REG.SYNCHPOINT値よりも小さいレコード抽出
            ORDER BY  A.IBMSNAP_COMMITSEQ ASC,  A.IBMSNAP_INTENTSEQ ASC <--CD表のINDEX順
```

■PRUNE_SET表の更新

```
Text     :UPDATE "ASN".IBMSNAP_PRUNE_SET SET  SYNCHPOINT = ?, SYNCHTIME = ?
          WHERE APPLY_QUAL = 'QUAL1'  AND SET_NAME = 'SET1' AND TARGET_SERVER = 'V8DB'
```

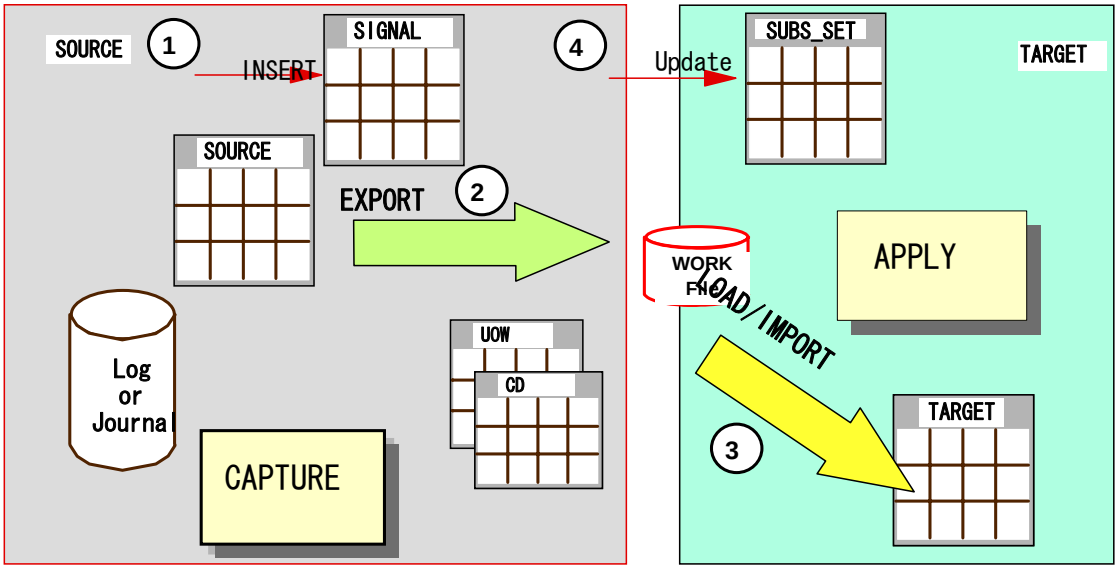
■パフォーマンス考慮点

CD表やUOW表に大量のデータがある場合、DB2 Catalogの統計情報が-1で一度もRUNSTATS,REORG等を実行していないと表スペースSCANを起こし、Elaps Time,CPU Timeに影響を与える場合がある。
このような事を避けるにはData量が多い段階で一度Catalog情報を更新し、Index Accessを選ばせるようにしておく

■"U" -> "I","I"->"U"の自動変換

Applyは差分COPY実行時、Updateが失敗するとINSERTへ、INSERTが失敗するとUpdateへ自動的に変換し、エラーにはしない。この場合APPLYTRAIL表のSET_REWORKEDの中に件数が記録される。
DELETEはFailしてもなにもしない。

Manual初期Load



- SOURCE ServerのSIGNAL表へ挿入
- SOURCE Table Unload
- ExportされたWork FileよりTarget表へLoadまたはImport
- TARGET ServerのSUBS_SET/SUBS_MEMBERを更新

```
update asn.ibmsnap_subs_set set lastrun=current timestamp,
lastsuccess=current timestamp,synchtime=current timestamp,synchpoint=null
where apply_qual='QUAL1',set_name='SET1';
update asn.ibmsnap_subs_member set member_state='L'
where apply_qual='QUAL1',set_name='SET1';
```


解説:

■Manual Loadの必要性

以下のような場合は、フルリフレシュをManual Loadで行なうことが有効。

- -表の大量コピーのロードを自動化するため
- -参照保全制約のある表を全リフレッシュするときに、参照保全検査をバイパスするため
- -複写元表および複写先表の整合性がくずれ(ターゲット表をdeleteした) 全件COPYが必要な場合

■SQL

キャプチャー・コントロール・サーバーで実行

```
UPDATE ASN.IBMSNAP_REGISTER SET DISABLE_REFRESH = 1 WHERE SOURCE_OWNER = 'E503230' AND SOURCE_TABLE = 'EMP'
AND SOURCE_VIEW_QUAL = 0;

UPDATE ASN.IBMSNAP_PRUNCNTL SET SYNCHPOINT = NULL, SYNCHTIME = NULL
WHERE SOURCE_OWNER = 'E503230' AND SOURCE_TABLE = 'EMP'
AND SOURCE_VIEW_QUAL = 0 AND APPLY_QUAL = 'QUAL1' AND SET_NAME = 'SET1'
AND TARGET_OWNER = 'E503230' AND TARGET_TABLE = 'TGEMP';

UPDATE ASN.IBMSNAP_PRUNCNTL SET SYNCHPOINT = X'00000000000000000000', SYNCHTIME = CURRENT TIMESTAMP
WHERE SOURCE_OWNER = 'E503230' AND SOURCE_TABLE = 'EMP'
AND SOURCE_VIEW_QUAL = 0 AND APPLY_QUAL = 'QUAL1' AND SET_NAME = 'SET1'
AND TARGET_OWNER = 'E503230' AND TARGET_TABLE = 'TGEMP';

INSERT INTO ASN.IBMSNAP_SIGNAL
(SIGNAL_TIME, SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN, SIGNAL_STATE) VALUES
(CURRENT TIMESTAMP, 'CMD', 'CAPSTART', 'I          ', 'P');
```

アプライ・コントロール・サーバーで実行

```
UPDATE ASN.IBMSNAP_SUBS_SET SET ACTIVATE = 0 WHERE
SET_NAME = 'SET1' AND APPLY_QUAL = 'QUAL1';
```

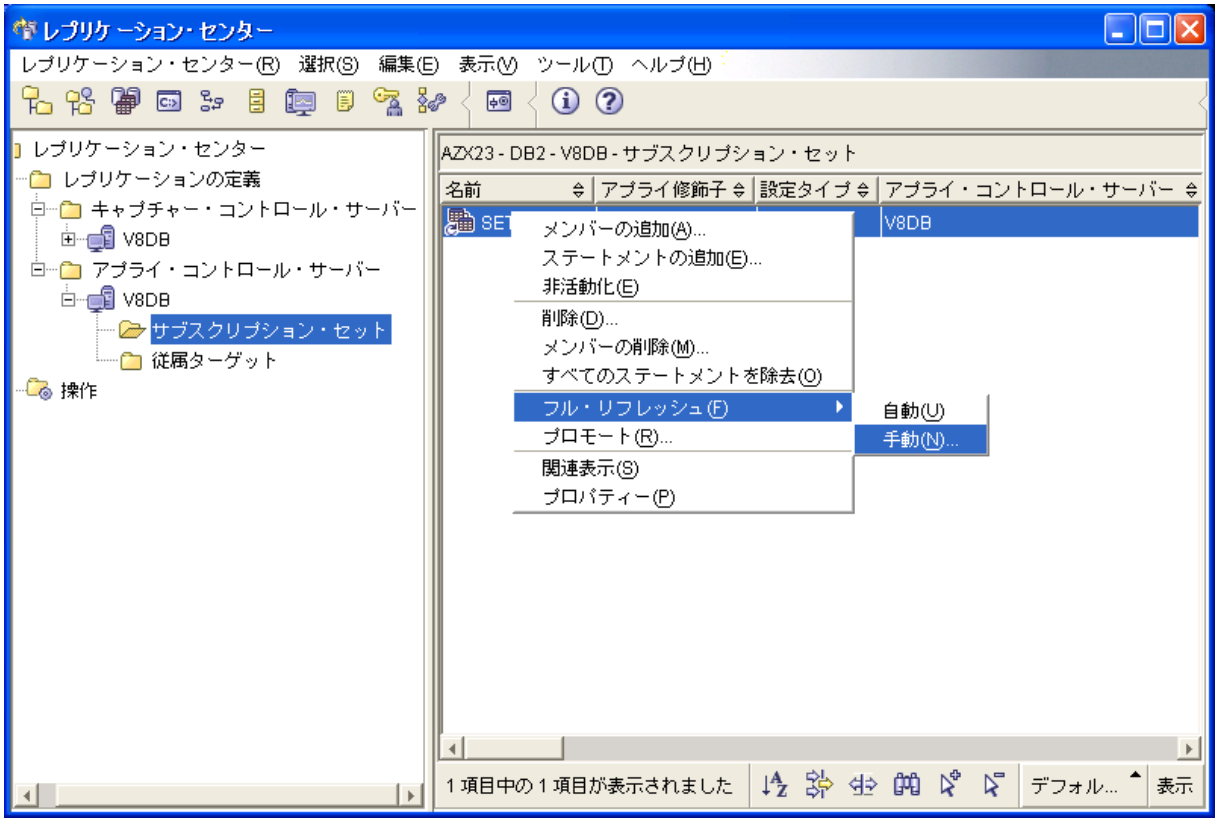
--- 手動フルリフレッシュ-----

アプライ・コントロール・サーバーで実行

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR SET MEMBER_STATE = 'L' WHERE
SET_NAME = 'SET1' AND APPLY_QUAL = 'QUAL1' AND WHOS_ON_FIRST = 'S';

UPDATE ASN.IBMSNAP_SUBS_SET SET ACTIVATE = 1, LASTRUN = CURRENT TIMESTAMP,
LASTSUCCESS = CURRENT TIMESTAMP, SYNCHTIME = CURRENT TIMESTAMP, SYNCHPOINT = NULL
WHERE SET_NAME = 'SET1' AND APPLY_QUAL = 'QUAL1'
```

解説:

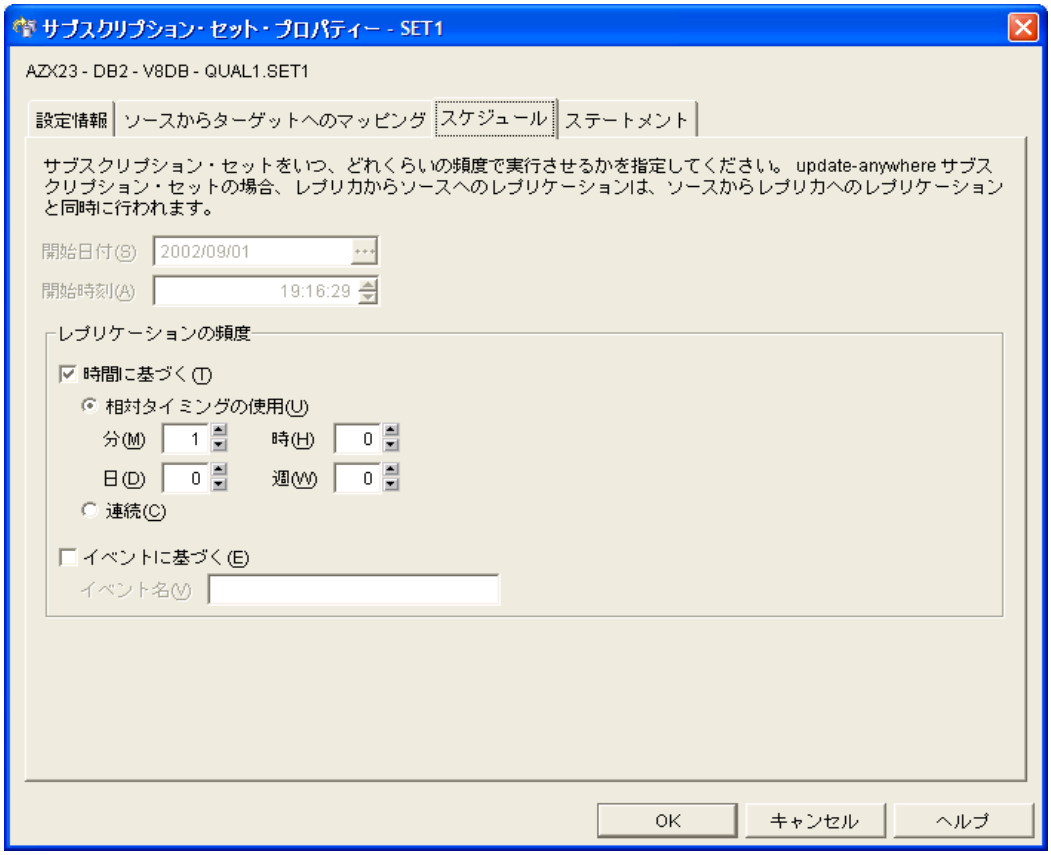
■RCからの起動



解説:

■このページはブランクです。

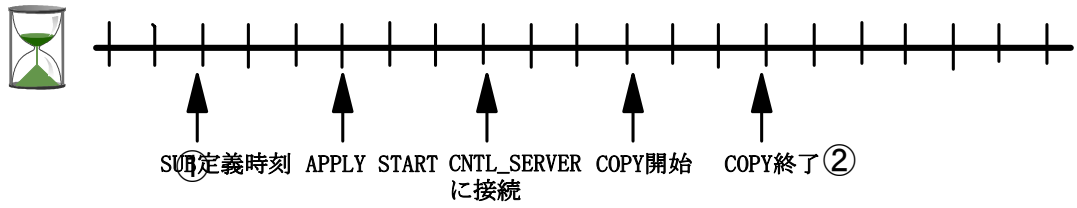
APPLY開始時刻



解説:

■APPLYはコントロールサーバーの時刻を基準に実行される
■GUIで指定した時刻はASN.IBMSNAP_SUBS_SET表のLASTRUN,LASTSUCCESSにセットされる

LASTRUN,LASTSUCCESS



LASTRUN,LASTSUCCESSの初期値はSubscription定義時刻のﾀｲﾑｽﾀﾝﾌﾟ

```
 CEXPC: connect to DBA
 GCST: Control server timestamp is
1997-05-25-12.09.28.620000
 R1NES: No eligible named event subscription at this moment
CS: Stop command has not been issued
 GMI: set type is READ_ONLY
 ------------------
     set_info
 ------------------
 ACTIVATE       = 1
 APPLY_QUAL     = QUAL1
 SET_NAME       = SET1
 WHOS_ON_FIRST = S
 SOURCE_SERVER = DBA
 SOURCE_ALIAS   = DBA
 TARGET_SERVER = DBA
 TARGET_ALIAS   = DBA
 STATUS         = 0
 LASTRUN        = 1997-05-25-12.08.28.310000
 REFRESH_TIMING = R
 SLEEP_MINUTES = 1
 EVENT_NAME     = null
 LASTSUCCESS    = 1997-05-25-12.08.28.310000
 SYNCPOINT      = null
 SYNCHTIME      = null
```

LASTRUN,LASTSUCCESSにサブスクリプション定義時刻ををSUBS_SETに記録する

LASTRUN	LASTSUCCESS
1997-05-25-12.08.28.310000	1997-05-25-12.08.28.310000

①

FULLREFRESH直後

LASTRUN = CNTL_SERVER接続開始時刻
LASTSUCCESS =前回LASTSUCCESS + SLEEP_MINUTES

LASTRUN	LASTSUCCESS
1997-05-25-12.09.28.620000	1997-05-25-12.09.28.310000

②

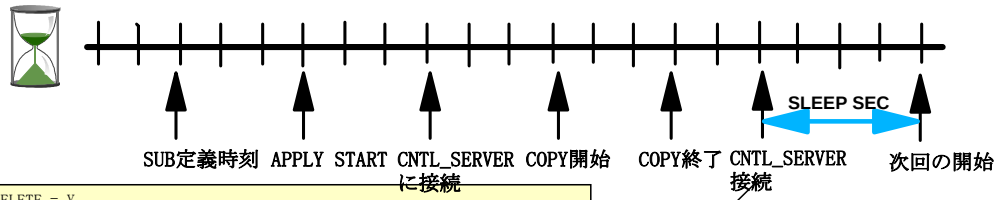
LASTRUNはCNTL_SERVER接続時刻
LASTSUCCESSは意図したSubscriptionの開始予定時刻になる

解説:

① サブスクリプション定義時はAPPLYの開始時刻（省略時はサブスクリプションを定義した現在時刻）を
SUBS_SET表のLASTRUN,LASTSUCCESSにセットする

② APPLYがスタートしフルリフレシュが完了すると
LASTRUN = CNTL_SERVER接続開始時刻
LASTSUCCESS = 前回LASTSUCCESS + SLEEP_MINUTES
に更新する

LASTRUN,LASTSUCCESSとSLEEP時間




```
SAT: MASS_DELETE = Y
  SAT: SET_INSERTED = 6
  SAT: SET_DELETED = 0
  SAT: SET_UPDATED = 0
  SAT: SET_REWORKED = 0
  SAT: SET_REJECTED_TRXS = 0
  SAT: STATUS = 0
  SAT: LASTRUN = 1997-05-25-12.09.28.620000
  SAT: LASTSUCCESS = 1997-05-25-12.09.28.310000
  SAT: SYNCHPOINT is null
  SAT: SYNCHTIME is 1997-05-25-12.09.31.500000
  SAT: SOURCE_ALIAS is DBA
  SAT: SOURCE_SERVER is DBA
  SAT: SOURCE_OWNER is
  SAT: SOURCE_TABLE is
  SAT: TARGET_ALIAS is DBA
  SAT: TARGET_SERVER is DBA
  SAT: TARGET_OWNER is
  SAT: TARGET_TABLE is
   CS: Stop command has not been issued
--- Process next subscription (3) ---
 CEXPC: connect to DBA
 GCST: Control server timestamp is 1997-05-25-12.09.32.840000
  R1NES: No eligible named event subscription at this moment
  R1RTS: No eligible relative timer driven subscription at this moment
 CS: Stop command has not been issued
  CS: Stop command has not been issued
 TTOS: slpDay is 0; slpHour is 0, slpMin is 0, slpSec is 55, slpMSec is 470000
 TTOS1: sleep_time = 56
 TTOS: delay_seconds = 56
 Delay_seconds = 56 seconds
  REST: Sleep time = 56 seconds.
    MSGF: MsgNumber is 1044.
   The NLS msg is (I) The Apply program will become inactive for 0 minutes and 56
seconds.
```

LASTSUCCESS+SLEPP_MINUTESが次回のSubscription開始時刻になる。

CNTL_SERVERの接続時刻が次回の開始時刻を越えていなければその時間sleepする。

解説:

■APLY START TIMEの計算ロジック
上記の例ではCNTL_SERVER接続時刻12:09.32.840000は次回の開始時刻、
LASTSUCCESS(12:09.28.310000) + SLEEP_MINUTES(1 Min)を越えていないためその差分56秒SLEEPする。

もしCOPY終了後、CNTL_SERVERの接続時刻がLASTSUCCESS + SLEEP_MINUTESを越えた場合はLASTSUCCESSの更新は前回のLASTSUCCESS + SLEEP_MINUTESの一番近いBoundary値に修正されます。

例)
前回のLASTSUCCESS=12:09.28.31
SLEEP_MINUTES=1
COPY終了後のCNTL_SERVER接続時刻=13:58.04.78
であった場合はLASTSUCCESS=13:57.28.31になる

LASTSUCCESSが更新されるのはCOPY終了後STATUS=0（正常終了）またはSTATUS=2の場合のみ,異常終了の場合(STATUS=-1)の場合には更新されないのでERRORの回復処置後ただちに実行可

Replicationに要した時間の求め方(エンドツーエンド待ち時間)

- レプリケーション・トランザクションのエンドツーエンド待ち時間は、キャプチャー・プログラムおよびアプライ・プログラムがサブスクリプション・セット内部であらゆるトランザクションを差分レプリケーションで複製するためにかかった時間。
  - Applytrail表の値から算出が可能
- (*ENDTIME* - LASTRUN) + (*SOURCE_CONN_TIME* - SYNCHTIME)
  - ENDTIME Applyがそのsetを終了した時刻
  - LASTRUN Applyがそのsetを開始した時刻
  - SOURCE_CONN_TIME Applyがデータをfetchする為にCaptureコントロールサーバーに接続したソースサーバー時刻
  - SYNCHTIME CaptureがCD表へ書き出した変更データのコミット時刻

パラメーター	列の値
ENDTIME	2001-10-20.10.01.00
LASTRUN	2001-10-20.10.00.30
SOURCE_CONN_TIME	2001-10-20.10.00.32
SYNCHTIME	2001-10-20.10.00.00

  - 上記の例では(10.01.00-10.00.30) + (10.00.32-10.00.00) = 30 + 32 = 64

解説:

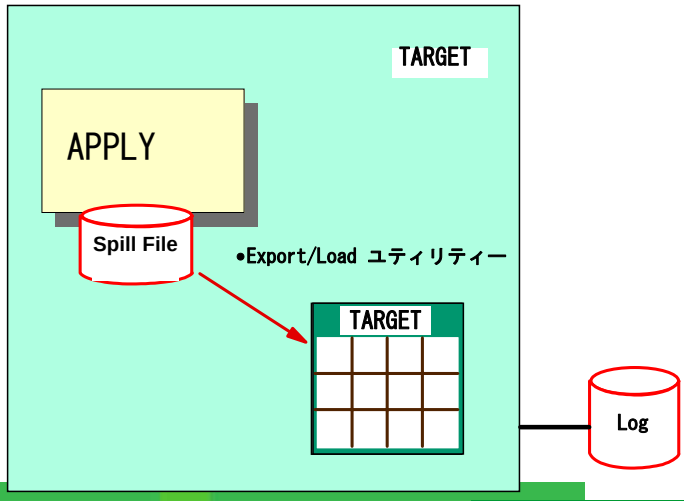
■エンドツーエンド待ち時間
　レプリケーション・トランザクションのエンドツーエンド待ち時間は、キャプチャー・プログラムおよびアプライ・プログラムがサブスクリプション・セット内部であらゆるトランザクションを差分レプリケーションで複製するためにかかった時間です。
　差分レプリケーションで発生したイベントのシーケンスの最初で、ソースとして使用されている登録済み表にトランザクションが コミットされます。 キャプチャー・プログラムはトランザクションのためのレコードを読み取り、そのレコードをソースのための CD 表に入れます。 キャプチャー・プログラムが CD 表でデータをコミットすると、登録表 (IBMSNAP_REGISTER) にあるグローバル・レコードの SYNCHTIME 列を更新します。 アプライ・プログラムが開始すると、キャプチャー・コントロール・サーバーに接続し、SYNCHTIME のための値を取得し、CD 表を読み取ります。 サブスクリプション・セットの処理を完了すると、アプライ・トレール (IBMSNAP_APPLYTRAIL) 表に行を挿入します。 値の中から、セットの処理を開始した時間 (LASTRUN) および完了した時間 (ENDTIME) に渡す値をこの行に入れます。

■エンドツーエンド待ち時間の計算に使用する値は次のとおりです。
■SYNCHTIME
　キャプチャー・プログラムによる、CD 表へのデータの最新コミットの時間
■LASTRUN
　アプライ・プログラムがサブスクリプション・セットの処理を開始する時間
■SOURCE_CONN_TIME
　アプライ・プログラムがキャプチャー・コントロール・サーバーに接続する時間
■ENDTIME
　アプライ・プログラムがサブスクリプション・セットの処理を完了する時間
■エンドツーエンド待ち時間を計算する公式は次のとおりです。
　(ENDTIME - LASTRUN) + (SOURCE_CONN_TIME - SYNCHTIME)

ASNLOAD Exit

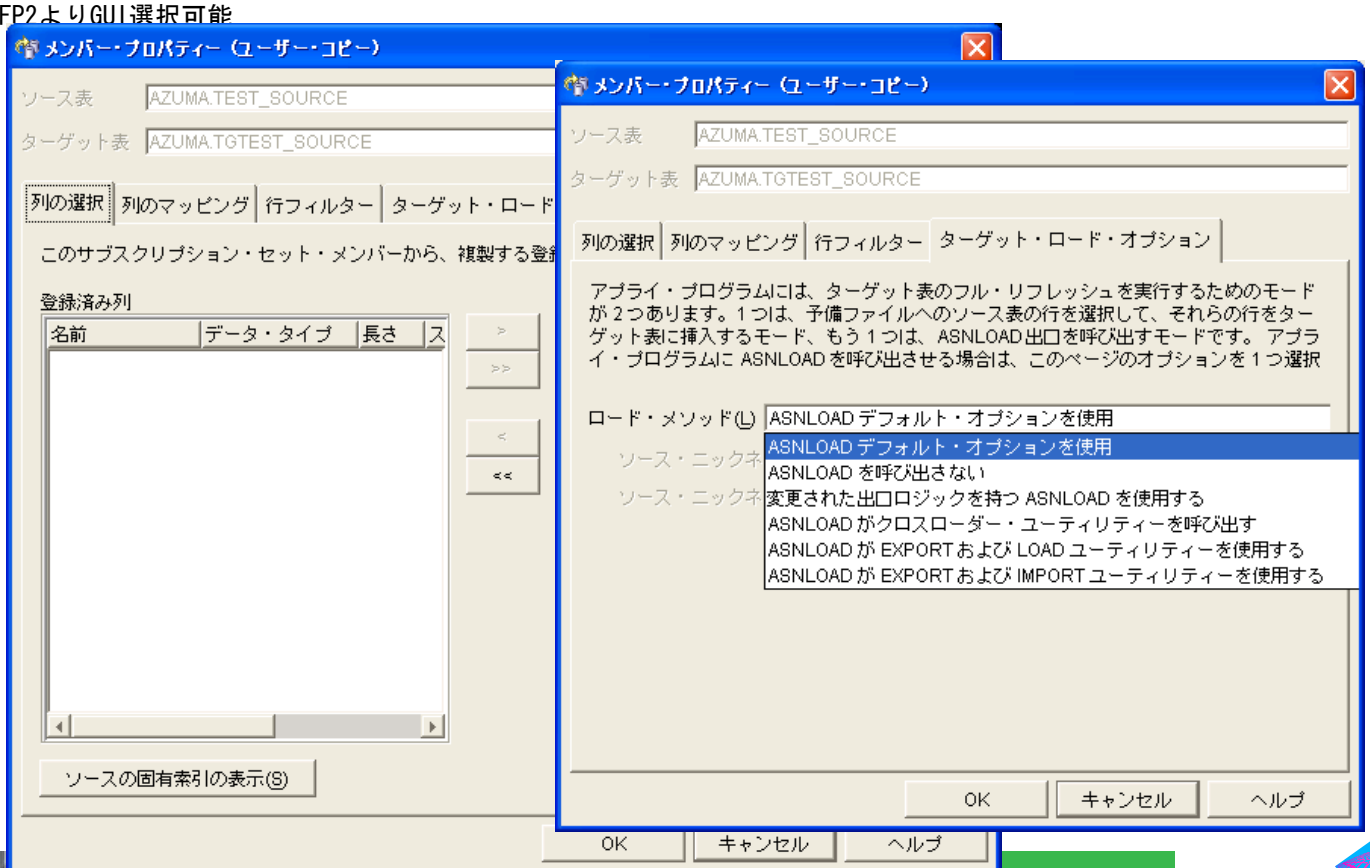
■ASNLOADとは

- 変更適用プログラム(APPLY)は、ターゲット表の全最新表示を実行するときは、そのつど ASNLOAD プログラムを呼び出すことができます。 LOADX パラメーターを指定して、変更適用プログラムがこのルーチンを呼び出すようにします。
- 差分COPYには使用できない
- Target-Source表のカラム順序は一致
- Target-DBのLOG量を軽減可能
  - ➤Active LOG Full(SQL0954N)回避



解説:

■FP2よりGUI選択可能



解説：

■省略時ではAPPLYはリフレッシュを実行するときASNLOADを使用しない。APPLYはINSERTステートメントを使用してSPILLファイルからターゲット表をフルリフレッシュする。ソース表が大きい場合にはASNLOADを使用して効率を上げることができる。

■ASNLOAD Configuration File(ASNLOAD.INI)

ASNLOADのカストマイズが可能。USERID,PASSWORDの指定が可能

```
;---asnload.ini sample file---
;Comments start with a semicolon. There is a section labeled [COMMON], followed by
;zero to n database specific sections. Each database specific section begins with the
;database alias in square brackets: [database alias].
;The database alias refers either to a source server, apply control server or a
;target server alias.

;If a database is not found in the inifile, the [COMMON] values would default for
;that database and if there is no applicable [COMMON] parameter in the inifile, then
;the default values in the asnload code for these keywords are in effect.

;The parameters are set below the section headers by specifying a keyword with its
;associated keyword value:
;KEYWORDNAME=keywordvalue
;NOTE: keywords are not case sensitive, however their values can be, so keyword values
;must be in quotes if mixed case, special characters or embedded blanks are required.

[COMMON]
;this is a comment line there are no common entries
[MYDBALIAS1]
copy=y
copyto=/twab/u/foo/copypath
lobfile=basefilename1, "baseFILEname2",
        basefilename3, basefilename4
;Note: In the case that multiple values are possible always leave a comma at the end
;of the line when you want to continue with more values in the next line (one
;commandline allows up to 1000 characters).
lobpath=/twab/u/foo/copypath1, "/twab/u/foo/copypath2"
[MYDBALIAS2]
copy=y
maxlobs=500000
;--- POSSIBLE KEYWORDS:---
;COPY  {Y/N}
;COPYTO {Path of the Copyimage}
;LOBFILE {List of Lobfilesbasenames}
;LOBPATH {List of Lobpaths}
;MAXLOBS {max number of lobfiles that can be created with a given list of Lobfilenames}
;UID     {only if compiled sample is used}
;PWD     {only if compiled sample is used}
;DATA_BUFFER_SIZE
;DISK_PARALLELISM
;CPU_PARALLELISM

;always leave a final comment line at the end of the inifile
```


解説：

■SUBS_MEMBR表のNEW COLUMN

■LOADX_TYPE　　SMALLINT

null - use predefined defaults　(省略時は4)
1 - do not call asnload for this member
2 - user defined, modified asnload exit code to be driven
3 - *crossloader*
4 - export/load
5 - export/import

■ LOADX_SRC_N_OWNER　VARCHAR(30)
LOADX_TYPE=3の場合に指定するTARGET_SERVER上に作成されているNICKNAMEのスキーマ名

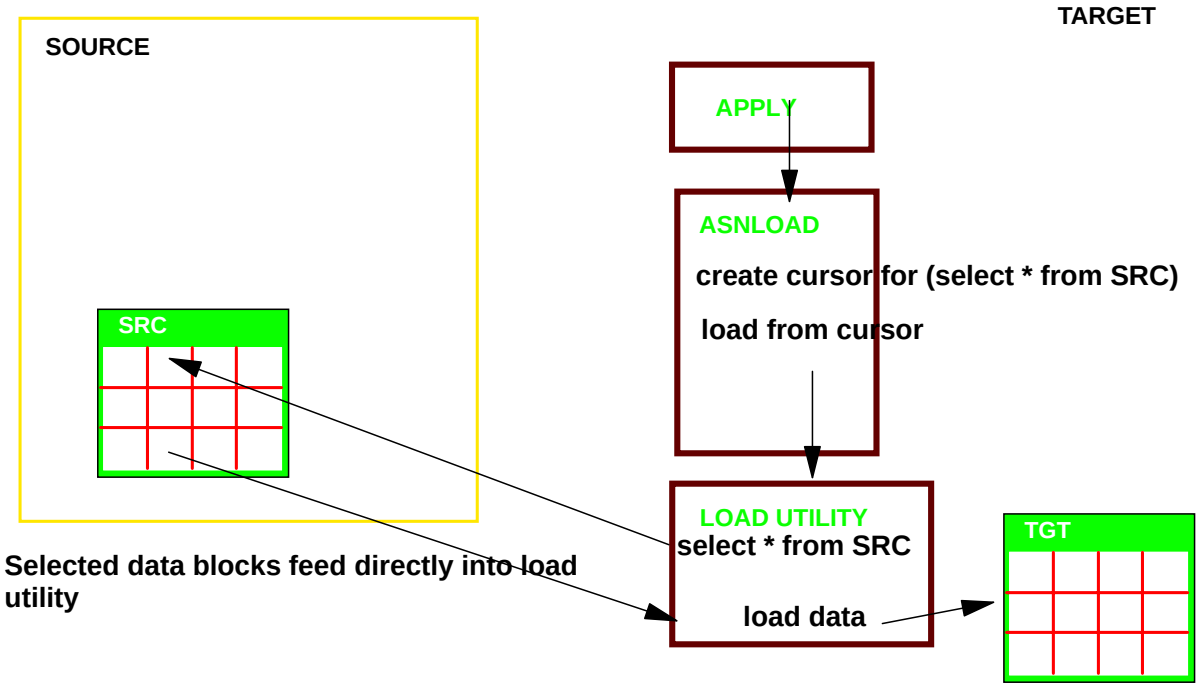
■ LOADX_SRC_N_TABLE　　VARCHAR(128)
LOADX_TYPE=3の場合に指定するTARGET_SERVER上に作成されているNICKNAMEの表名

■ 現行のレプリケーションセンターでは上記の３つに値を指定することがサポートされていない為、手動で行う必要

```
uncatalog node local;
catalog local node local instance db2;
drop wrapper DRDA;
create wrapper DRDA;
create server  FEDDB  type DB2/NT VERSION 7.1 WRAPPER DRDA AUTHORIZATION "azuma" PASSWORD
"azuma" options (Node 'LOCAL',DBNAME 'V8DB',FOLD_ID 'L',FOLD_PW 'L');
create user mapping for USER server FEDDB
options (Remote_authid 'azuma',Remote_password 'azuma');
create nickname NICK For FEDDB.AZUMA.TEST_SOURCE;
-- Update the subscription members table to tell Apply to use the sample Cross Loader ASNLOAD exit PGM
update ASN.IBMSNAP_SUBS_MEMBR set loadx_src_n_owner='AZUMA', loadx_src_n_table='NICK',loadx_type=3
where apply_qual='QUALN';
```

解説：

■CROSSLOADER (Unix,Windows)
UDB V8から使用可能になったWORK FILEなしでカーサーからLOAD可能なUtility
"Load from that cursor"とも呼ばれるリモートからLOADする場合はNICKNAMEを使用しなければならない。



解説：

■ASNLOADの実行例
==> asnapply APPLY_QUAL=QUAL1 control_server=V8DB LOADX=Y
copyonce=Y trcflow

```
Apply program compiled at 03:28:51 on Aug  8 2002 (Level 80116m)
Apply qualifier is QUAL1.
Control srvr name is V8DB.
Issue Sleep msg.
Invoke ASNLOAD.
Will not invoke ASNDONE.
COPYONCE is specified
NOLOGREUSE is in use
NOLOGSTDOUT is in use
NOTRLREUSE is in use
SQLERRCONTINUE is not in use
COMMIT(n) is not specified
APP_PATH is /home/db2v8/dpr/
Instance name is db2v8.
  ABIND: sqlca.sqlcode is 0
 CIMPC: currtSrvrType=SQL. currtSrvrVersion=08010.
 CIMPC: Userid is DB2V8
 CIMPC: cntl server is 8
2002-09-01-23.24.58.692517 ASN1045I  APPLY "QUAL1". The Apply program was started
 using database "V8DB".
  The NLS msg is ASN1045I  APPLY "QUAL1". The Apply program was started using database "V8DB".
--- Process next subscription (1) ---
CPGCST: Control server timestamp is 2002-09-01-23.24.58.721608
CPFES: beginTS is 2002-09-01-23.24.58.721608
Compiled(P) at 03:29:19 on Aug  8 2002 (Level 80116m)
  CPGCI: numKeys is 1
 GMI: set type is READ_ONLY1


 ---------------------------
 CLOS: setRepeatCopy is 0
 CLOS: activate = 1
 CLOS: status   = 0
 CLOS: lastrun = 2002-09-01-23.24.58.721608
 CLOS: lastsuccess = 2002-09-01-23.25.00.000000
 CLOS: synchpoint is null
 CLOS: synchtime = 2002-09-01-23.24.58.772753
 CLOS: apply_qual = QUAL1
 CLOS: set_name = SET1
 CLOS: sWhosOnFirst = S
```

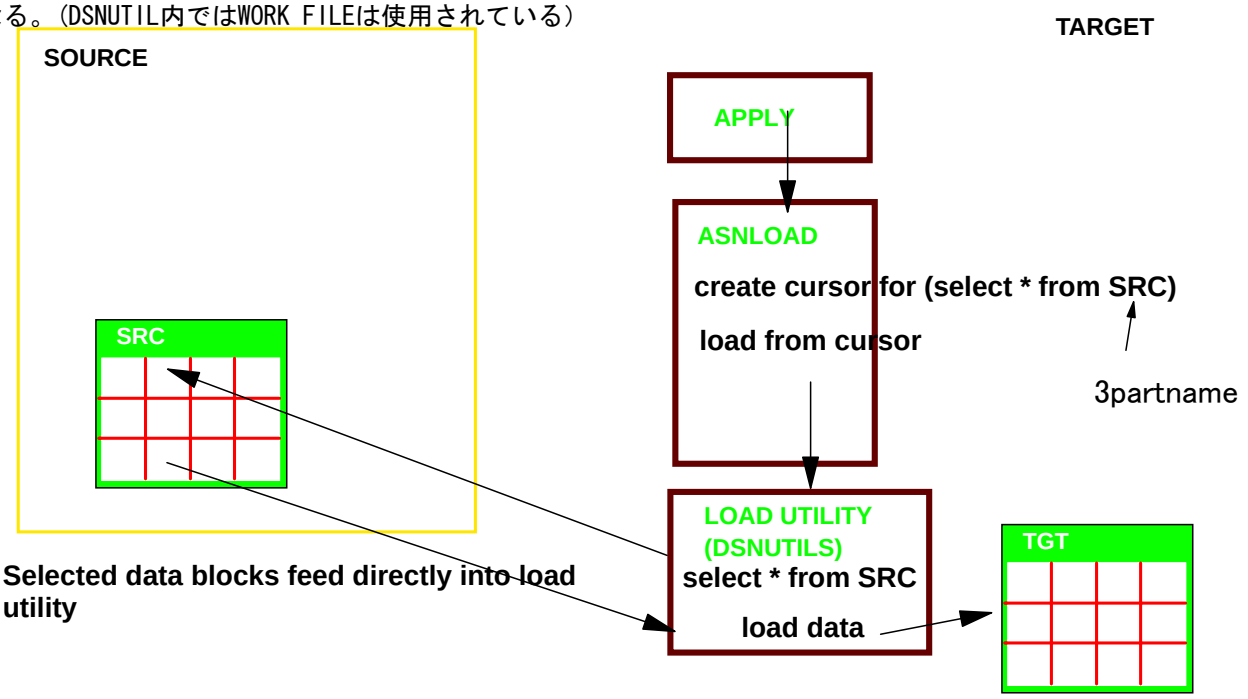
```
PSET: currtSrvrType is SQL, currtSrvrVersion is 08
PSET: connect to V8DB; rc = 1
 SAT: ASNLOAD = Y, EFFECT_MEMBERS = 1
 SAT: FULL_REFRESH = Y
 SAT: SET_INSERTED = 0
 SAT: SET_DELETED = 0
 SAT: SET_UPDATED = 0
 SAT: SET_REWORKED = 0
 SAT: SET_REJECTED_TRXS = 0
 SAT: STATUS = 0
 SAT: LASTRUN = 2002-09-01-23.24.58.721608
 SAT: LASTSUCCESS = 2002-09-01-23.25.00.000000
 SAT: SYNCHPOINT is null
 SAT: SYNCHTIME is 2002-09-01-23.24.58.772753
 SAT: SOURCE_ALIAS is V8DB
 SAT: SOURCE_SERVER is V8DB
 SAT: SOURCE_OWNER is
 SAT: SOURCE_TABLE is
 SAT: TARGET_ALIAS is V8DB
 SAT: TARGET_SERVER is V8DB
 SAT: TARGET_OWNER is
 SAT: TARGET_TABLE is
 SAT: SQLSTATE is null
 SAT: SQLERRM is null
 SAT: SQLCODE is null
 SAT: SQLERRP is null
 SAT: APPERRM is null
--- Process next subscription (2) ---
CPGCST: Control server timestamp is 2002-09-01-23.25.00.842997
 R1SE: No eligible subscription at this moment
ApplyWorker: files_allocated is -1;  lf_allocated is -1;  dlf_allocated is -1
ApplyWorker: totalrc is 0. totalcf is 0.
2002-09-01-23.25.07.744957 ASN1097I  APPLY "QUAL1". The Apply
program stopped.
```

ゼロ

解説：

■CROSSLOADER (z/OS)

Windows,Unix環境と同様にサポートされるがストアードプロシージャーDSNUTILSのセットアップが必要。
このストアードプロシージャーはWLM管理下で実行される必要がある。またLOADX=Yを指定してAPPLYから実行前にASNLOAD,APPLYおよびDSNUTIS自身のPACKAGEをソース側にバインドしておく必要がある。z/OSでは3part Nameを使用してRemoteのソース表をアクセスする。したがってDRDAを使用してUDB上のORACLEなどからもReplicationが可能になる。(DSNUTIL内ではWORK FILEは使用されている)



解説：

■z/OSでのApplyのリモートバインド

```
//BNDUDB    JOB CLASS=A,TIME=1440,
//          MSGCLASS=H,MSGLEVEL=(1,1),REGION=4M,NOTIFY=AZUMA
//*************************************************************
//* Free Capture, Apply and Monitor packages in DSN6 if
//* previous versions of packages are not needed
//* Local bind of Capture, Apply and Monitor packages in DSN6
//*************************************************************
//ASNBIND  EXEC PGM=IKJEFT01
//STEPLIB  DD DSN=DSN71G.SDSNEXIT,DISP=SHR
//         DD DSN=DSN71G.SDSNLOAD,DISP=SHR
//         DD DSN=DSN71G.RUNLIB.LOAD,DISP=SHR
//DBRMLIB  DD DSN=ASN810.DPROPR.V810.DBRMLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(D71A)

 BIND PACKAGE(DB2W.ASNCOMMON) MEMBER(ASNDBCON) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNCOMMON) MEMBER(ASNMSGT) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNCOMMON) MEMBER(ASNSQLCF) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNCOMMON) MEMBER(ASNSQLCZ) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
BIND PACKAGE(ASNAPPLY) MEMBER(ASNAAPP) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNAWPN) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNACMP) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNAFET) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(CS) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNAISO) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNAMAN) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNAPPWK) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNAPRS) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
 BIND PACKAGE(DB2W.ASNAPPLY) MEMBER(ASNLOAD) -
 ACTION(REP) -
 DEGREE(ANY)  -
 VALIDATE(RUN) -
 ISOLATION(UR) -
 KEEPDYNAMIC(YES)
BIND PLAN(ASNTA810) -
 PKLIST(*.ASNCOMMON.*,*.ASNAPPLY.*) -
   ACQUIRE(USE) ACTION(REPLACE) DEGREE(ANY)  -
   RELEASE(COMMIT) ISOLATION(UR) -
  KEEPDYNAMIC(YES)
/*
```

解説：

■z/OSでのLOADX=Y指定結果
MVS to MVS replication

```
---> Detail  =    DSN07011
--> Calling fetch1() with SQL :
    SELECT COUNT(*) FROM "AZUMA"."TEST_SOURCE"
--> on server = DB2W
--> RET = 0
--> indexes  == 1
--> for_keys == 0
--> num_rows   == 6
--> getSortKeysTotal() == 6

ASNLOAD input values passed by APPLY:
   Target Server  = KUSATSU
   Target Owner   = AZUMA
   Target Table.  = TGTEST_SOURCE
   Trace          = yes
   Trace Filename = /u/AZUMA/QUAL1.LOADTRC
   Source Server  = DB2W
   Control Server  = KUSATSU
   App. Qualifer  = QUAL1
   SQL. Stmt      = SELECT  "ROW_NUMBER", "TEST_DATA", "DESCRIPTION"
   FROM  "AZUMA"."TEST_SOURCE" A  OPTIMIZE FOR 50000 ROWS

Crossloader Stmt:
   SQL. Stmt      = TEMPLATE UNLDE DSN AZUMA.UNLDE TEMPLATE WORKDSN1
                    DSN AZUMA.UNLDEI
                    TEMPLATE WORKDSN2 DSN AZUMA.UNLDEO EXEC SQL
                    DECLARE C1 CURSOR FOR SELECT  "ROW_NUMBER", "TEST_DATA",
                    "DESCRIPTION"
                    FROM "DB2W"."AZUMA"."TEST_SOURCE" A  OPTIMIZE FOR 50000
                    ROWS
                    ENDEXEC LOAD DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS
                    6 INTO
                    TABLE "AZUMA"."TGTEST_SOURCE" STATISTICS
   Apply path     = /u/AZUMA/
   Loadx Type     = NULL

--------> Running LOAD...

ASNLOAD Running under user = AZUMA
WorkDSN1 = AZUMA.UNLDEI
WorkDSN2 = AZUMA.UNLDEO
```

```
------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN AZUMA.UNLDE TEMPLATE WORKDSN1 DSN AZUMA.UNLDEI TEMPLATE
WORKDSN2 DSN AZUMA.UNLDEO EXEC SQL DECLARE C1 CURS
OR FOR SELECT  "ROW_NUMBER", "TEST_DATA", "DESCRIPTION" FROM
"DB2W"."AZUMA"."TEST_SOURCE" A  OPTIMIZE FOR 50000 ROWS ENDEXEC LOAD
DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 6 INTO TABLE "AZUMA"."TGTEST_SOURCE" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = LOAD
ODSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN AZUMA.UNLDE
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN AZUMA.UNLDEI
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN AZUMA.UNLDEO
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ODSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR
 SELECT "ROW_NUMBER","TEST_DATA","DESCRIPTION" FROM
 "DB2W"."AZUMA"."TEST_SOURCE" A OPTIMIZE FOR 50000 ROWS ENDEXEC
ODSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 6
 DSNU650I  -D71A DSNURWI -  INTO TABLE "AZUMA"."TGTEST_SOURCE" STATISTICS
 DSNU350I  -D71A DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU610I  -D71A DSNUSUTP -   SYSTABLEPART CATALOG UPDATE FOR DSNDB04.TGTESTRS SUCCESSFUL
 DSNU610I  -D71A DSNUSUTB -     SYSTABLES CATALOG UPDATE FOR
 AZUMA.TGTEST_SOURCE SUCCESSFUL
 DSNU610I  -D71A DSNUSUTS -   SYSTABLESPACE CATALOG UPDATE FOR DSNDB04.TGTESTRS SUCCESSFUL
 DSNU620I  -D71A DSNURDRT - RUNSTATS CATALOG TIMESTAMP = 2002-09-06-16.11.08.372645
 DSNU304I  -D71A DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=6 FOR
 TABLE AZUMA.TGTEST_SOURCE
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=6
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:06
 DSNU042I    DSNUGSOR - SORT PHASE STATISTICS -
                        NUMBER OF RECORDS=6
                        ELAPSED TIME=00:00:00
 DSNU349I  -D71A DSNURBXA - BUILD PHASE STATISTICS - NUMBER OF KEYS=6 FOR
 INDEX AZUMA.IXTGTEST_SOURCE
 DSNU258I    DSNURBXD - BUILD PHASE STATISTICS - NUMBER OF INDEXES=1
 DSNU259I    DSNURBXD - BUILD PHASE COMPLETE, ELAPSED TIME=00:00:00
 DSNU381I  -D71A DSNUGSRX - TABLESPACE DSNDB04.TGTESTRS IS IN COPY PENDING
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
*** Calling REPAIR ...
------- INITIATING CALL -------
SQL: REPAIR SET TABLESPACE DSNDB04.TGTESTRS NOCOPYPEND

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = REPAIR
ODSNU050I    DSNUGUTC -  REPAIR
 DSNU650I  -D71A DSNUCBRS -   SET TABLESPACE DSNDB04.TGTESTRS NOCOPYPEND
 DSNU651I  -D71A DSNUCBRS - SET NOCOPYPEND OPERATION SUCCESSFUL
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
--------> LOAD ended with CODE = 0

Exiting ASNLOAD
```


解説：

■z/OS ASNLOAD がFailした例

Apply実行結果

```
 ------------------------
  Compiled(A) at 15:37:17 on Aug 28 2002 (Level 80116y)
   USRX: return from ASNLOAD, rc = 98
2002-09-11-07.32.15.522449 ASN1053E  APPLY "QUAL1".  The execution of the ASNLOAD exit routine failed.
The return code is "98".
     The NLS msg is ASN1053E  APPLY "QUAL1".  The execution of the ASNLOAD exit routine failed.  The return code is "98".
   USRX: ASNLOAD failed errcode is 725308.
PSET: ROLLBACK
PSET: success is 0
PSET: currtSrvrType is SQL, currtSrvrVersion is 07
PSET: connect to DB2W; rc = 1
PSET: currtSrvrType is SQL, currtSrvrVersion is 07
PSET: connect to KUSATSU          ; rc = 1
 CLOS: setRepeatCopy is 0
 CPCLOS: success is 0. retcode is 0.
 CLOS: activate = 1
 CLOS: status   = -1
 CLOS: lastrun = 2002-09-11-16.28.46.234474
 CLOS: lastsuccess is null
 CLOS: synchpoint is null
 CLOS: synchtime is null
 CLOS: apply_qual = QUAL1
 CLOS: set_name = SET1
 CLOS: sWhosOnFirst = S
 SAT: ASNLOAD is null
 SAT: FULL_REFRESH is null
 SAT: SET_INSERTED = 0
 SAT: SET_DELETED = 0
 SAT: SET_UPDATED = 0
 SAT: SET_REWORKED = 0
 SAT: SET_REJECTED_TRXS = 0
 SAT: STATUS = -1
```

解説：

■QUAL1.LOADTRCの内容

```
ASNLOAD input values passed by APPLY:
   Target Server  = KUSATSU
   Target Owner   = AZUMA
   Target Table.  = TGTEST_SOURCE
   Trace          = yes
   Trace Filename = /u/AZUMA/QUAL1.LOADTRC
   Source Server  = DB2W
   Control Server  = KUSATSU
   App. Qualifer  = QUAL1
   SQL. Stmt      = SELECT  "ROW_NUMBER", "TEST_DATA", "DESCRIPTION" FROM  "AZUMA"."TEST_SOURCE" A  OPTIMIZE FOR 50000 ROWS
Crossloader Stmt:
   SQL. Stmt      = TEMPLATE UNLDE DSN AZUMA.UNLDE TEMPLATE WORKDSN1 DSN AZUMA.UNLDE1 TEMPLATE WORKDSN2 DSN AZUMA.UNLDE0 EXEC SQL
DECLARE C1 CURSOR FOR SELECT  "ROW_NUMBER", "TEST_DATA", "DESCRIPTION" FROM "DB2W"."AZUMA"."TEST_SOURCE" A  OPTIMIZE FOR 50000 ROW
S ENDEXEC LOAD DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 5 INTO TABLE "AZUMA"."TGTEST_SOURCE" STATISTICS
   Apply path     = /u/AZUMA/
   Loadx Type     = NULL
---------> Running LOAD...
ASNLOAD Running under user = AZUMA
WorkDSN1 = AZUMA.UNLDE1
WorkDSN2 = AZUMA.UNLDE0
------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN AZUMA.UNLDE TEMPLATE WORKDSN1 DSN AZUMA.UNLDE1 TEMPLATE WORKDSN2 DSN AZUMA.UNLDE0 EXEC SQL DECLARE C1 CURS
OR FOR SELECT  "ROW_NUMBER", "TEST_DATA", "DESCRIPTION" FROM "DB2W"."AZUMA"."TEST_SOURCE" A  OPTIMIZE FOR 50000 ROWS ENDEXEC LOAD
DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 5 INTO TABLE "AZUMA"."TGTEST_SOURCE" STATISTICS
** Error: SqlException caught by ASNLDX2 main.
**   SQL Stmt: CALL SYSPROC.DSNUTILS
**   Method..: DsnUtils::callDsnUtils
***   DSNT408I SQLCODE = -471, ERROR:  INVOCATION OF FUNCTION OR PROCEDURE SYSPROC   →・¥
***            .DSNUTILS FAILED DUE TO REASON 00E7900C                              →・¥
***   DSNT418I SQLSTATE   = 55023 SQLSTATE RETURN CODE                              →・¥
***   DSNT415I SQLERRP    = DSNX9WCA SQL PROCEDURE DETECTING ERROR                  →・¥
***   DSNT416I SQLERRD    = 0  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION            →・¥
***   DSNT416I SQLERRD    = X'00000000'  X'00000000'  X'00000000'  X'FFFFFFFF'      →・¥
***            X'00000000'  X'00000000' SQL DIAGNOSTIC INFORMATION                  →・¥
***                                                                                 →・¥
***                                                                                 →・¥
***                                                                                 →・¥
```


解説：

■z/OS上でのWLM状況

```
D WLM,APPLENV=*
IWM029I  17.20.20  WLM DISPLAY 839
  APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
  BBOASR1                          AVAILABLE
  BBOASR2                          AVAILABLE
  CBINTFRP                         AVAILABLE
  CBNAMING                         AVAILABLE
  CBSYSMGT                         AVAILABLE
  WLMENV1                          STOPPED

VARY WLM,APPLENV=WLMENV1,RESUME
IWM032I VARY RESUME FOR WLMENV1 COMPLETED

D WLM,APPLENV=*
IWM029I  17.21.42  WLM DISPLAY 843
  APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
  BBOASR1                          AVAILABLE
  BBOASR2                          AVAILABLE
  CBINTFRP                         AVAILABLE
  CBNAMING                         AVAILABLE
  CBSYSMGT                         AVAILABLE
  WLMENV1                          AVAILABLE

S D71AWLM
$HASP100 D71AWLM  ON STCINRDR
IEF695I START D71AWLM  WITH JOBNAME D71AWLM  IS ASSIGNED TO USER IBMUSER  , GROUP SYS1
$HASP373 D71AWLM  STARTED
IEF403I D71AWLM - STARTED - TIME=17.22.27
```

解説:

■z/OSでのLOADX=Y指定結果
UDB to MVS replication

```
ASNLOAD input values passed by APPLY:
   Target Server  = KUSATSU
   Target Owner   = AZUMA
   Target Table.  = TGSOURCE
   Trace          = yes
   Trace Filename = /u/AZUMA/QUAL1.LOADTRC
   Source Server  = V8DB
   Control Server  = KUSATSU
   App. Qualifer  = QUAL1
   SQL. Stmt      = SELECT  "C1", "C2", "C3" FROM  "DB2V8"."SOURCE" A

Crossloader Stmt:
   SQL. Stmt      = TEMPLATE UNLDE DSN .UNLDE TEMPLATE WORKDSN1
DSN .UNLDEI TEMPLATE WORKDSN2 DSN .UNLDEO EXEC SQL DECLARE C1 CURSOR
FOR SELECT  "C1", "C2", "C3" FROM "V8DB"."DB2V8"."SOURCE" A
ENDEXEC LOAD DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 0 INTO TAB
LE "AZUMA"."TGSOURCE" STATISTICS
   Apply path     = /u/AZUMA/
   Loadx Type     = NULL

--------> Running LOAD...

ASNLOAD Running under user = AZUMA
WorkDSN1 = AZUMA.UNLDEI
WorkDSN2 = AZUMA.UNLDEO

------- INITIATING CALL -------
SQL: TEMPLATE UNLDE DSN .UNLDE TEMPLATE WORKDSN1
 DSN .UNLDEI TEMPLATE WORKDSN2 DSN .UNLDEO EXEC SQL DECLARE C1 CURSOR FOR
SELECT
"C1", "C2", "C3" FROM "V8DB"."DB2V8"."SOURCE" A  ENDEXEC LOAD DATA
INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 0 INTO TABLE "AZUMA"."TGS
OURCE" STATISTICS

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = LOAD
0DSNU050I    DSNUGUTC -  TEMPLATE UNLDE DSN .UNLDE
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN1 DSN .UNLDEI
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE WORKDSN2 DSN .UNLDEO
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  EXEC SQL DECLARE C1 CURSOR FOR SELECT "C1","C2","C3"
FROM "V8DB"."DB2V8"."SOURCE" A ENDEXEC
0DSNU050I    DSNUGUTC -  LOAD DATA INCURSOR(C1) REPLACE LOG(NO) SORTKEYS 0
 DSNU650I   -D71A DSNURWI -  INTO TABLE "AZUMA"."TGSOURCE" STATISTICS
 DSNU350I   -D71A DSNURRST - EXISTING RECORDS DELETED FROM TABLESPACE
 DSNU610I   -D71A DSNUSUTP -   SYSTABLEPART CATALOG UPDATE FOR DSNDB04.TGSOURCE SUCCESSFUL
 DSNU610I   -D71A DSNUSUTB -     SYSTABLES CATALOG UPDATE FOR AZUMA.TGSOURCE SUCCESSFUL
 DSNU610I   -D71A DSNUSUTS -   SYSTABLESPACE CATALOG UPDATE FOR DSNDB04.TGSOURCE SUCCESSFUL
 DSNU620I   -D71A DSNURDRT - RUNSTATS CATALOG TIMESTAMP = 2003-05-04-20.18.30.859545
 DSNU304I   -D71A DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=1 FOR
TABLE AZUMA.TGSOURCE
 DSNU302I    DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=1
 DSNU300I    DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:16
 DSNU042I    DSNUGSOR - SORT PHASE STATISTICS -
                        NUMBER OF RECORDS=1
                        ELAPSED TIME=00:00:00
 DSNU349I   -D71A DSNURBXA - BUILD PHASE STATISTICS - NUMBER OF KEYS=1 FOR
INDEX AZUMA.IXTGSOURCE
 DSNU258I    DSNURBXD - BUILD PHASE STATISTICS - NUMBER OF INDEXES=1
 DSNU259I    DSNURBXD - BUILD PHASE COMPLETE, ELAPSED TIME=00:00:01
 DSNU381I   -D71A DSNUGSRX - TABLESPACE DSNDB04.TGSOURCE IS IN COPY PENDING
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
*** Calling REPAIR ...
------- INITIATING CALL -------
SQL: REPAIR SET TABLESPACE DSNDB04.TGSOURCE NOCOPYPEND

1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = REPAIR
0DSNU050I    DSNUGUTC -  REPAIR
 DSNU650I   -D71A DSNUCBRS -   SET TABLESPACE DSNDB04.TGSOURCE NOCOPYPEND
 DSNU651I   -D71A DSNUCBRS - SET NOCOPYPEND OPERATION SUCCESSFUL
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
--------> LOAD ended with CODE = 0

Exiting ASNLOAD
```


ASNDONE Exit

■ASNDONEとは

- 変更適用プログラムは、サブスクリプション処理が成功または失敗のどちらであるかに関係なく、その処理後にオプションとして ASNDONE 出口ルーチンを呼び出すことができます。 必要に応じて、このルーチンを変更することができます。 たとえば、このルーチンは、UOW 表を検査して、拒否されたトランザクションを検出したら、メッセージの発行やアラートの生成などの事後アクションを始動することができます。 この出口ルーチンの別の使用法は、障害が発生した (状況 = -1) サブスクリプション・セットを非活動化して、その障害が修正されるまで変更適用プログラムが再試行しないようにすることです。 この出口ルーチンを修正する方法について詳しくは、¥sqllib¥samples¥replディレクトリーにあるサンプル・プログラム (ASNDONE.SMP) の prolog 部分をご覧ください。
- NOTIFY=Yで呼び出す

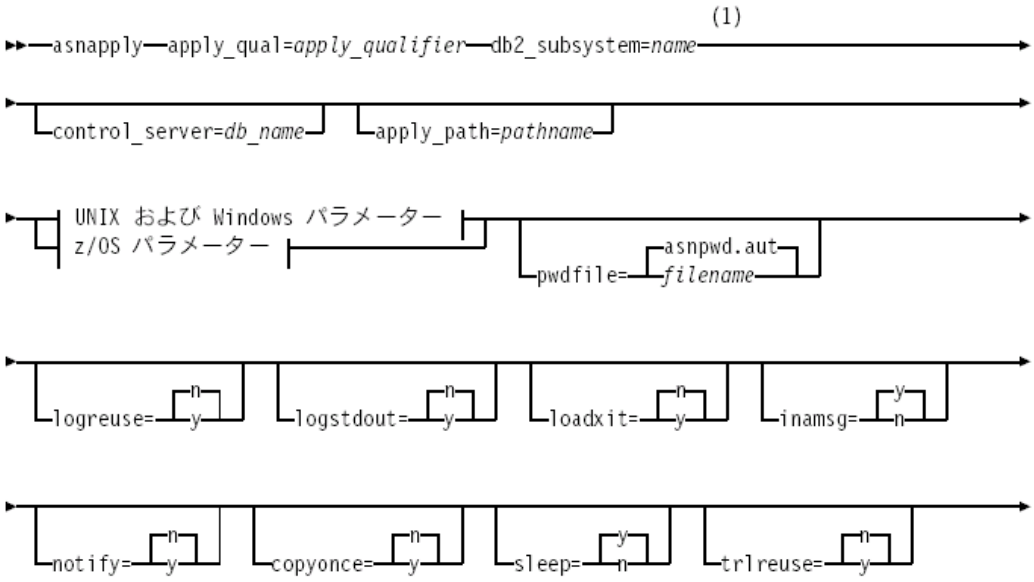
APPLYの開始パラメーター

- スタートパラメーター
  - ***apply_path***
  - apply_qual
  - control_server
  - copyonce
  - delay
  - db2_subsystem (z/OS only)
  - errwait
  - inamsg
  - loadxit
  - logreuse
  - logstdout
  - notify
  - ***opt4one***
  - ***pwdfile*** (UNIX and Windows only)
  - sleep
  - spillfile (choice for z/OS only)
  - ***sqlerrcontinue***
  - ***term***
  - trlreuse
  - *Highlighted* parameters(new function)

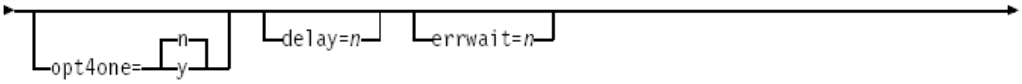
解説:

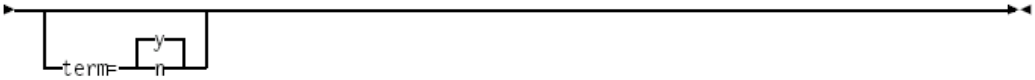
■APPLY Parameter

asnapply コマンドを使用してアプライ・プログラムを始動するための構文

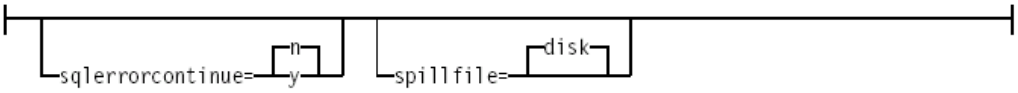


解説:

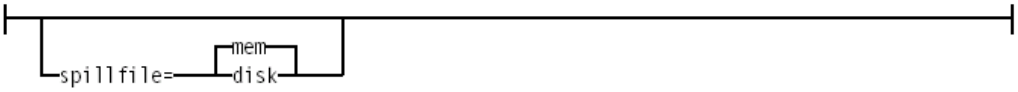




UNIX および Windows パラメーター:



z/OS パラメーター:



解説:

パラメーター	定義
apply_qual=*apply_qualifier*	アプライ・プログラムが、処理されるサブスクリプション・セットの識別に使用するアプライ修飾子を指定します。このパラメーターは必須です。 入力する値は、サブスクリプション・セット(IBMSNAP_SUBS_SET)表のAPPLY_QUAL列の値と一致する必要があります。アプライ修飾子名には大文字小文字の区別があり、最大18文字です。
db2_subsystem=*name*	**z/OSの場合のみ**:DB2サブシステムの名前を指定します。入力するDB2サブシステム名は最大4文字です。このパラメーターにはデフォルトはありません。このパラメーターは必須です。
control_server=*db_name*	サブスクリプション定義とアプライ・プログラム・コントロール表が存在する、アプライ・コントロール・サーバーの名前を指定します。 **UNIXおよびWindowsの場合**:アプライ・コントロール・サーバーを指定しないと、このパラメーターはデフォルトでDB2DBDFT環境変数の値になります。 **z/OSの場合**:コントロール・サーバー・パラメーターは、コントロール・サーバーに接続するデータベース・サーバーの名前です。

解説:

パラメーター	定義
apply_path=*pathname*	アプライ・プログラムが使用する作業ファイルのロケーションを指定します。デフォルトは、asnapplyコマンドが呼び出されたディレクトリーです。
pwdfile=*filename*	パスワード・ファイルの名前を指定します。パスワード・ファイルを指定しない場合、デフォルトはasnpwd.autです。 このコマンドは、apply_pathパラメーターで指定されたディレクトリー内でパスワード・ファイルを探します。apply_pathパラメーターを指定しないと、このコマンドは、コマンドを呼び出したディレクトリー内でパスワード・ファイルを探します。
logreuse=*y/n*	アプライ・プログラムが、ログ・ファイル(db2instance.control_server.apply_qualifier.APP.log)を再利用するか、またはメッセージを付加するかを指定します。 n(デフォルト) アプライ・プログラムは、アプライ・プログラムの再始動後であっても、ログ・ファイルにメッセージを付加します。 y アプライ・プログラムは、ログ・ファイルを削除し、アプライ・プログラムの再始動時にそれを再作成することにより、ログ・ファイルを再利用します。 z/OSの場合:ログ・ファイルにはDB2インスタンス名は含まれません(control_server.apply_qualifier.APP.log)。

解説:

logstdout=*y/n*	アプライ・プログラムがメッセージをどこに送信するかを指定します。 n(デフォルト) アプライ・プログラムはログ・ファイルにのみメッセージを送信します。 y アプライ・プログラムは、メッセージをログ・ファイルと標準出力(stdout)の両方に送信します。
loadxit=*y/n*	アプライ・プログラムがASNLOADを呼び出すかどうかを指定します。ASNLOADはIBM提供の出口ルーチンであり、エクスポートおよびロード・ユーティリティーを使用して、ターゲット表をリフレッシュします。 n(デフォルト) アプライ・プログラムはASNLOADを呼び出しません。 y アプライ・プログラムはASNLOADを呼び出します。

解説:

パラメーター	定義
inamsg=*y/n*	アプライ・プログラムを非アクティブにしたとき、アプライ・プログラムからメッセージを出すかどうかを指定します。 y(デフォルト) アプライ・プログラムは非アクティブ時にメッセージを出します。 n アプライ・プログラムは非アクティブ時にメッセージを出しません。
notify=*y/n*	アプライ・プログラムがASNDONEを呼び出すかどうかを指定します。ASNDONEは、アプライ・プログラムがサブスクリプション・セットのコピーを終了した時に、ユーザーにコントロールを戻すための出口ルーチンです。 n(デフォルト) アプライ・プログラムはASNDONEを呼び出しません。 y アプライ・プログラムはASNDONEを呼び出します。

解説:

copyonce=*y/n*	アプライ・プログラムが呼び出された時点で適格と見なされたサブスクリプション・セットごとに、アプライ・プログラムがコピー・サイクルを1回実行するかどうかを指定します。その後、アプライ・プログラムは終了します。適格と見なされるサブスクリプション・セットとは、以下の基準を満たすものです。 ● サブスクリプション・セット(IBMSNAP_SUBS_SET)表の中で(ACTIVATE>O)。ACTIVATE列の値がゼロより大きい場合、そのサブスクリプション・セットは無期限にアクティブであるか、または1回のみのサブスクリプション処理に使用されています。 ● (REFRESH_TYPE=RまたはB)または(REFRESH_TYPE＝Eであり、指定されたイベントが発生)。REFRESH_TYPE列の値はIBMSNAP_SUBS_SET表に保管されます。 サブスクリプション・セット表のMAX_SYNCH_MINUTES限度および、サブスクリプション・イベント(IBMSNAP_SUBS_EVENT)表のEND_OF_PERIODタイム・スタンプが指定されている場合は、これに従います。 n(デフォルト) アプライ・プログラムは、適格なサブスクリプション・セットごとにコピーサイクルを1回実行しません。 y アプライ・プログラムは、適格なサブスクリプション・セットごとにコピー・サイクルを1回実行します。

解説:

パラメーター	定義
sleep＝*y/n*	処理の対象として適格となる新しいサブスクリプションがない場合に、アプライ・プログラムがどうするかを指定します。 y(デフォルト) アプライ・プログラムはスリープ状態に入ります。 n　アプライ・プログラムは停止します。
trlreuse＝*y/n*	アプライ・プログラムの始動時に、アプライ・プログラムがアプライ・トレール(IBMSNAP_APPLYTRAIL)表を空にするかどうかを指定します。 n(デフォルト) アプライ・プログラムは IBMSNAP_APPLYTRAIL 表にエントリーを付加します。アプライ・プログラムは表を空にしません。 y　アプライ・プログラムはプログラム始動時に IBMSNAP_APPLYTRAIL 表を空にします。

解説:

opt4one＝*y/n*	アプライ・プログラムに定義されているサブスクリプション・セットが1つだけの場合、アプライ・プログラムのパフォーマンスを最適化するかどうかを指定します。 n(デフォルト) サブスクリプション・セットが1つの場合、アプライ・プログラムのパフォーマンスを最適化しません。 y　サブスクリプション・セットが1つの場合、アプライ・プログラムのパフォーマンスを最適化します。 最適化をyに指定すると、アプライ・プログラムはサブスクリプション・セット・メンバーの情報をキャッシュに入れて再利用します。このようにサブスクリプション・セット・メンバーの情報を再利用すると、CPU使用率が減り、スループットが向上します。
delay＝*n*	連続レプリケーションを使用する場合に、それぞれのアプライ・サイクルが終了した後、何秒待つかを示す遅延時間(秒単位)を指定します。nは、0、1、2、3、4、5または6です。デフォルトは6です。
errwait＝*n*	アプライ・プログラムがエラー状態になった後、何秒待ってから再試行するかを示す秒数(1から300)を指定します。デフォルト値は300秒(5分)です。 **重要**:アプライ・プログラムはほとんど切れ目なく稼動しており、アプライ・トレール(IBMSNAP_APPLYTRAIL)表に多くの行を生成するので、ここにあまり小さい数を指定しないでください。

解説:

パラメーター	定義
term＝*y*/*n*	DB2の状況が、アプライ・プログラムの動作にどのように影響するかを指定します。 y（デフォルト） DB2が終了すると、アプライ・プログラムは終了します。 n DB2がアクティブでない場合、アプライ・プログラムはDB2の始動を待ちます。 **UNIX および Windows の場合**：DB2が静止し、アプライ・プログラムがアクティブの場合、アプライ・プログラムはアクティブのままであり、DB2が静止モードでなくなるまで再接続しません。 **z/OS の場合**：DB2が静止し、アプライ・プログラムがアクティブの場合、アプライ・プログラムはアクティブのままであり、DB2が再度開始されるまで再接続しません。

解説:

sqlerrorcontinue＝*y*/*n*	**UNIX および Windowsの場合のみ**：アプライ・プログラムがある種のSQLエラーを検出した場合、アプライ・プログラムが処理を継続するかどうかを指定します。 アプライ・プログラムは、失敗したSQLSTATEをSQLSTATEファイルに指定された値に照らしてチェックします。このSQLSTATEファイルは、アプライ・プログラムの実行前にユーザーが作成します。ファイルの内容と一致すれば、アプライ・プログラムは失敗した行についての情報をエラー・ファイル（*apply_qualifier*.ERR）に書き込み、処理を継続します。SQSTATEファイルには5バイト値を20個まで含めることができます。 n（デフォルト） アプライ・プログラムはSQLSTATEファイルをチェックしません。 y アプライ・プログラム処理中にSQLSTATEファイルをチェックします。

解説：

spillfile＝*filetype*	フェッチした応答セットをどこに保管するかを指定します。 **UNIX および Windowsの場合、**有効な値は次のとおりです。 **disk**（デフォルト） ディスク・ファイル。 **z/OSの場合、**有効な値は次のとおりです。 **mem**（デフォルト） メモリー・ファイル。応答セット用の十分なメモリーがない場合、アプライ・プログラムは失敗します。 **disk**　ディスク・ファイル。

解説：

■z/OSでのApplyの始動JCL

```
//APPLYAZ  JOB CLASS=A,MSGCLASS=H,REGION=0M,NOTIFY=AZUMA
//*        USER=DB2004,PASSWORD=DB2004
/*JOBPARM S=ZOS1
//QUAL1    EXEC PGM=ASNAPPLY,
//             PARM='/QUAL1 D71A KUSATSU LOGREUSE
//             DISK logstdout loadxit=y COPYONCE=Y TRCFLOW'
//* with copyonce parameter
//*            PARM='/QUAL1 D71A KUSATSU LOGREUSE DISK logstdout copyonce'
//*
//*****************************************************************
//* Example of running V8 Apply
//* This shows how to specify Apply_path
//* Apply qualifier AQREC2
//* Subsystem name D7DP
//* Control server location name D7DP
//* logreuse,logstdout,copyonce (all this can be specified
//*                              in asn.ibmsnap_appaparms table)
//*
//* APPLY_PATH=//JAYAV8 (can be specified in asn.ibmsnap_appparms)
//* Apply trc and log files will go to
//* JAYANTI.JAYAV8.D7DP.AQREC2..APP.LOG
//*********************************************************************
//***********  APPLY_PATH=//JAYAV8'
//STEPLIB  DD  DISP=SHR,DSN=ASN810.DPROPR.V810.LOADLIB
//         DD  DISP=SHR,DSN=CEE.SCEERUN
//         DD  DISP=SHR,DSN=DSN71G.SDSNLOAD
//CEEDUMP  DD  DUMMY
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//ASNASPL  DD  DSN=&&ASNASPL,DISP=(NEW,DELETE,DELETE),
//          UNIT=SYSDA,SPACE=(CYL,(11,7)),
//          DCB=(RECFM=VS,BLKSIZE=6404)
//
```

APPLY CAF=N

■Applyは通常RRSAFでなくCAFでDB2 z/OSへ接続

- ●もしRRSAFで接続したい場合はCAF=Nを指定する
- ●APAR PQ95770 でCAF=N/Yの指定が可能になった

■PARM='ENVAR("TZ=JST-9")/QUAL1 D71A KUSATSU LOGREUSE DISK logstdout loadxit=N TRCFLOW'

```
DSNV401I  -D71A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -D71A ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
DB2CALL  T       65 AZUMAA       AZUMA    ASNTA810 002A   229
DB2CALL  T        9 AZUMAA       AZUMA    ASNTA810 002A   230
DB2CALL  T       30 AZUMAA       AZUMA    ASNTA810 002A   231
DB2CALL  T       14 AZUMAA       AZUMA    ASNTA810 002A   232
```


解説:

■PARM='ENVAR("TZ=JST-9")/QUAL1 D71A KUSATSU LOGREUSE DISK logstdout loadxit=N TRCFLOW **CAF=N**'

```
DSNV401I  -D71A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -D71A ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    T       66 AZUMAA.Initi AZUMA    ASNTA810 002A   222
 V437-WORKSTATION=*, USERID=Initial,
      APPLICATION NAME=QUAL1
RRSAF    T       10 AZUMAA.HoldL AZUMA    ASNTA810 002A   223
 V437-WORKSTATION=*, USERID=HoldLThread,
      APPLICATION NAME=QUAL1
RRSAF    T      955 AZUMAA.Worke AZUMA    ASNTA810 002A   224
 V437-WORKSTATION=*, USERID=WorkerThread,
      APPLICATION NAME=QUAL1
RRSAF    T       45 AZUMAA.Admin AZUMA    ASNTA810 002A   225
 V437-WORKSTATION=*, USERID=AdminThread,
      APPLICATION NAME=QUAL1
```

ASN.IBMSNAP_APPPARMS表

■APPLY始動時に指定がない場合、使用される値

≺apply_qual	(no default)	≺logstdout	(N)
≺apply_path	(current directory)	≺notify	(N)
≺copyonce	(N)	≺opt4one	(N)
≺delay	(6 – seconds)	≺sleep	(Y)
≺errwait	(300 – seconds)	≺spillfile	(M z/OS, D other)
≺inamsg	(Y)	≺sqlerrcontinue	(N)
≺loadxit	(N)	≺term	(Y)
≺logreuse	(N)	≺trlreuse	(N)

- apply_qual, control_server, (for zSeries) db2_subsystem. これらのパラメーターは必須
- APPPARMS表の値はスタートアップ時の指定で上書きされる

解説:

■実行例

```
asnapply apply_qual=AQ1 control_server=dbx apply_path=/home/files/apply/  pwdfile=pass1.txt
```

APPLY操作コマンド

■Apply稼動中に実行可能なコマンド

- asnacmdコマンドを使用して以下の操作コマンドを入力可能:
  - STOP
  - STATUS
    - もしApplyが稼働していなかった場合にはコミュニケーションエラーを返す
- Captureへの操作コマンドは: asnccmd、Applyへはasnacmdとなり、モニターへの操作コマンドはasnmcmd.

解説:

■操作コマンド

- Apply command syntax:

Apply command syntax:

```
>>-asnacmd----+-------------------------------+------------>
              '-apply_qual-- = --apply qualifier--'

 >-----+-----------------------------+--------------------->
       '-control_server-- =  --db_name--'

 >-----+-status--------------------+----------------------><
       '-stop----------------------'
```

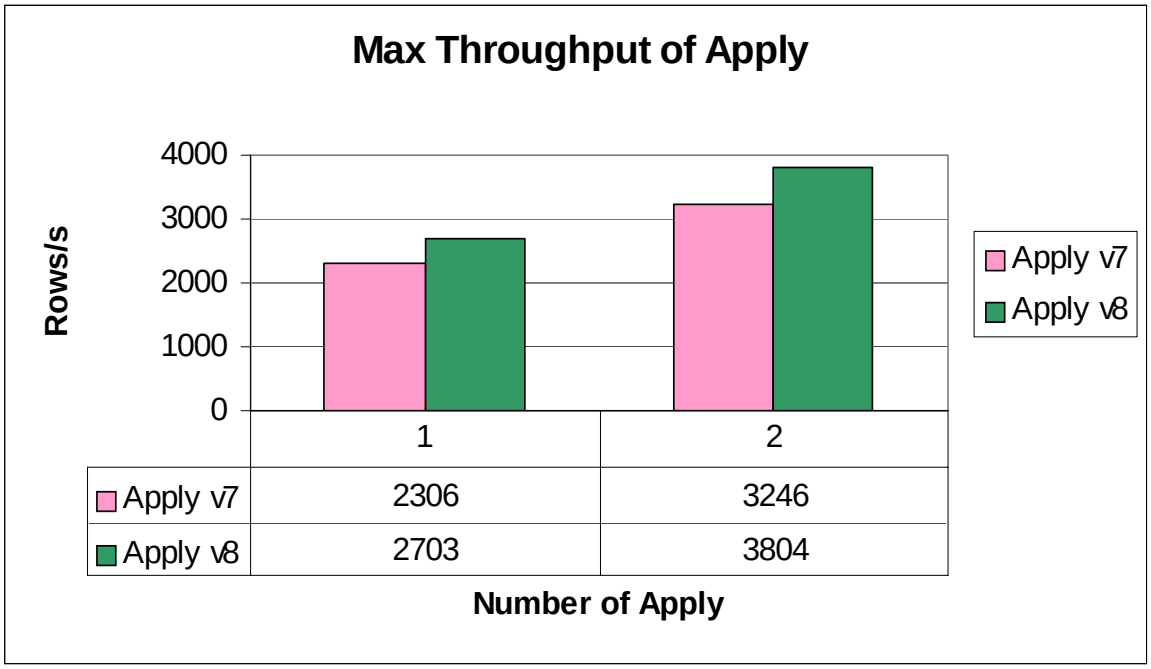
- 例(指定位置規則なし)

```
asnacmd status apply_qual=AQ1 control_server=trgdb
asnacmd apply_qual=AQ1 status control_server=trgdb
asnacmd apply_qual=AQ1 control_server=trgdb status
```

Apply パフォーマンス比較

Apply V8 vs Apply V7 on the Same Machine

- Apply V8:
  - ►17% improvement



Questions ?