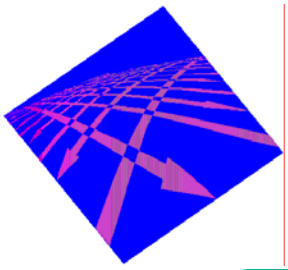


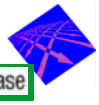
運用・管理・モニタリング

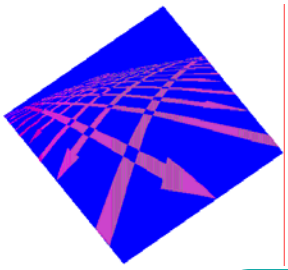
＜第2. 00版＞2006年01月

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。

内容

- 運用
  - Capture / Apply の起動・停止・操作
  - プルーニングのタイミング
  - 障害対応
  - フルリフレッシュ
  - レプリケーション環境の変更
  - キャプチャー済みアーカイブ・ログの判別
- 管理・モニタリング
  - 管理・モニタリングでよく使用する制御表
  - 日常監視する項目
  - モニタリングFAQ
  - エラーは発生していないか
  - 様々なトレース
  - 整合性の確認方法
  - レプリケーションセンターでのモニタリング
  - レプリケーション・アラート・モニター





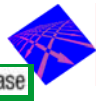
運用

＜第2. 00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。

内容

- Capture / Apply の起動・停止・操作
- プルーニングのタイミング
- 障害対応
- フルリフレッシュ
- レプリケーション環境の変更
- キャプチャー済みアーカイブ・ログの判別



Capture / Apply の起動・停止・操作

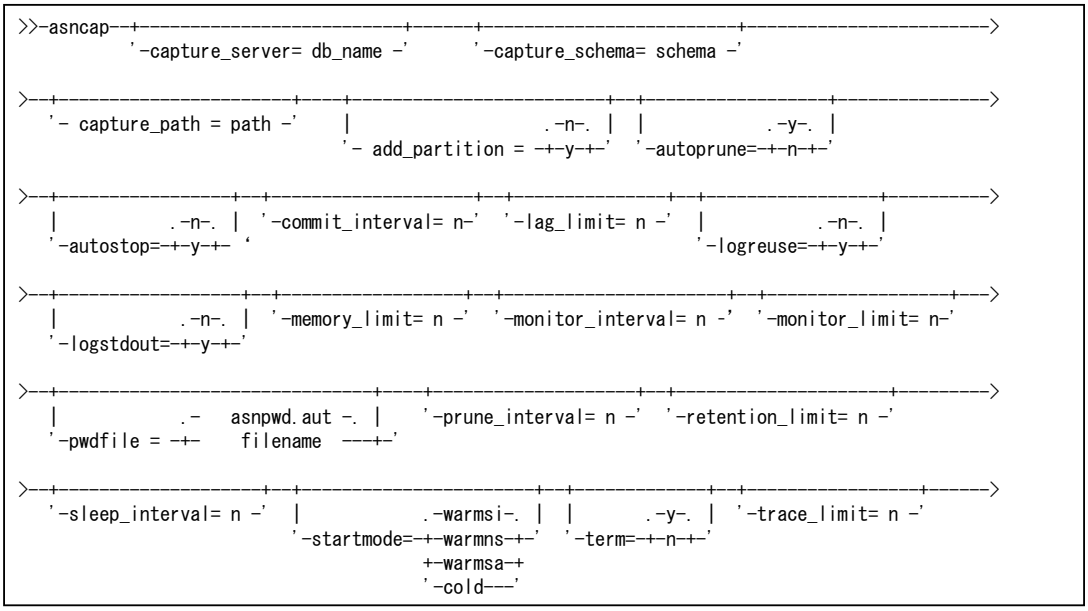
- Captureの起動・停止
  - コマンド/GUIを使用した起動
  - 状況のチェック
  - コマンド/ GUIを使用した停止
- Applyの起動・停止
  - コマンド/GUIを使用した起動
  - 状況のチェック
  - コマンド/ GUIを使用した停止
- Captureの操作
  - asnccmd コマンド
  - SIGNAL表の使用方法
  - Capture/Apply for z/OS ARM Support
- Applyの操作
  - asnacmd コマンド
  - EVENT表の使用方法

blank page

コマンドを使用したCaptureの起動

■ asncap コマンドを使用

- 起動例 ： asncap CAPTURE_WERVER（＋オプション）
- 環境 ： UNIX、Windows、および z/OS上のUSS



コマンドを使用したCaptureの起動

■ Capture起動時入力パラメーター

パラメーター	デフォルト値	説明
capture_server	Unix,Windows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のコントロールサーバーの名前
capture_schema	ASN	1-30文字のCAPTUREスキーマ名
capture_path	asncapが呼び出されたディレクトリー	CAPTUREが使用する作業ファイルのロケーション
add_partition	N	区画を追加時CaptureのWARM Start時に認識させるかどうか
autoprune	N	(CD)(UOW)(IBMSNAP_CAPMON)(IBMSNAP_CAPTRACE)表の自動プルーニングをするかどうか
autostop	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	CD,UOW表に書かれた内容をコミットする時間間隔
lag_limit	10080 (分)	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	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	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	32 (MB)	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。

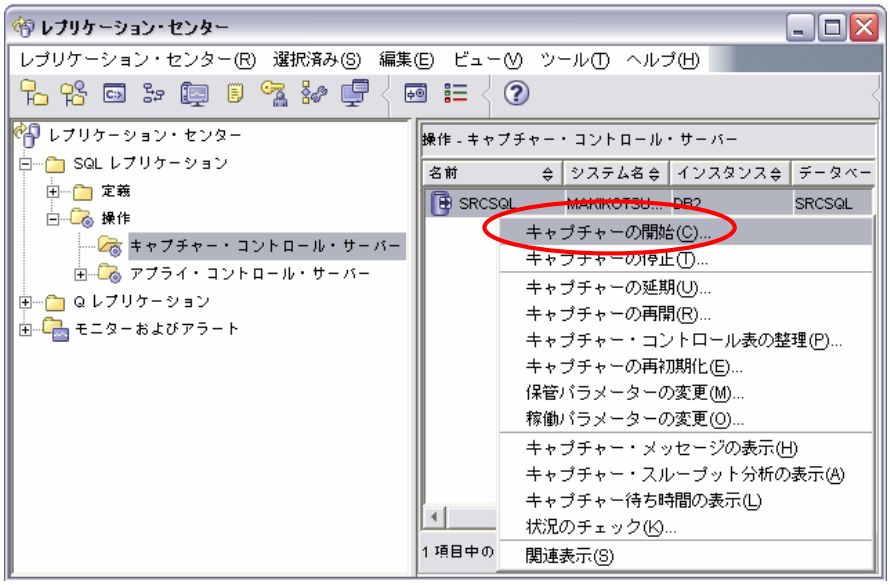
コマンドを使用したCaptureの起動

■ Capture起動時入力パラメーター

パラメーター	デフォルト値	説明
monitor_interval	5 (分)	CAPTUREがMONITOR表(IBMSNAP_CAPMON)表に行を挿入する間隔
monitor_limit	10080 (分)	MONITOR表の行がPRUNING対象になるかの分数を指定
pwdfile	asnpwd.out	パスワード・ファイルの名前を指定
prune_interval	300 (秒)	(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
retention_limit	10080 (分)	CD,UOW,SIGNAL表の行の最大保持期間
sleep_interval	5 (秒)	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
startmode	warmsi	CAPTUREのスタート時の処理タイプを指定 warmsi :差分収集から開始する、エラー時には終了する。初回定義時のみ、自動的にcoldスタートに切り替わる warmns : 差分収集から開始する、エラー時には終了する。warmスタートできなくてもcoldスタートには切り替わらない warmsa :差分収集から開始する、warmスタートできない場合coldスタートに切り替わる cold : 次回アプライ起動時に、フルリフレッシュを行う
term	Y	DB2が終了した場合にCAPTUREを終了するかどうか Y :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	10080 (分)	CAPTRACE表の行がPRUNING対象の適格になる分数を指定

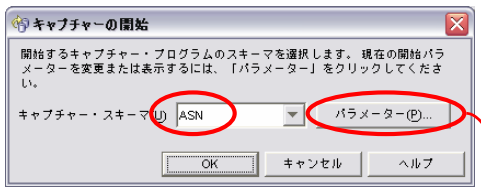
GUIを使用したCaptureの起動

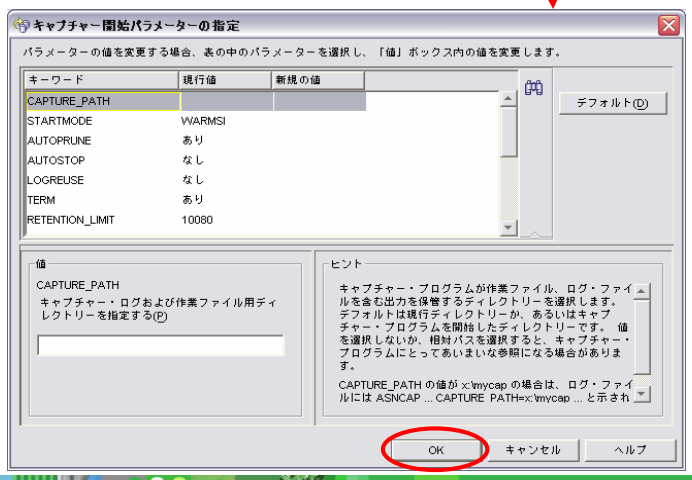
■ レプリケーションセンターを起動
■ 「SQLレプリケーション」→「操作」→「キャプチャー・コントロール・サーバー」→(サーバー上で右クリック)→「キャプチャーの開始」



GUIを使用したCaptureの起動

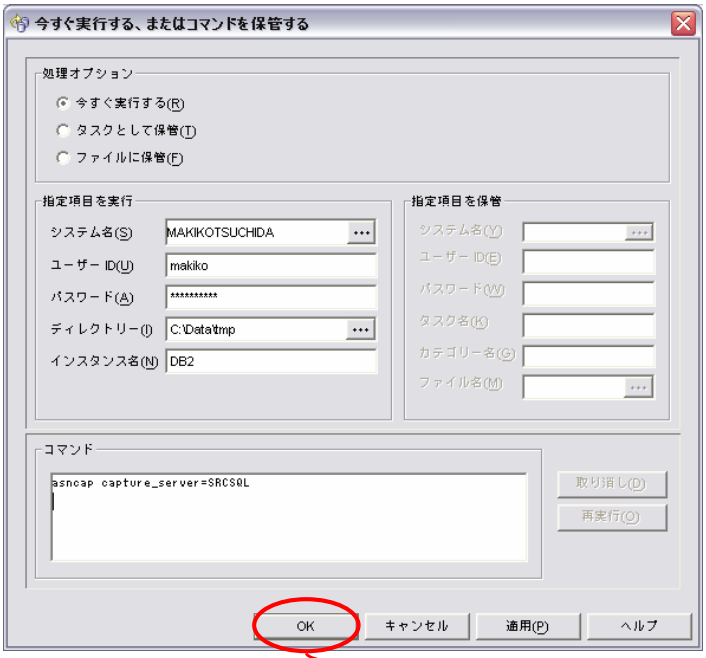
- キャプチャー・スキーマの指定（デフォルトはASN）





- 必要に応じてパラメーター値を変更
  何も指定しない場合、
  IBMSNAP_CAPPARMSの値が使用される
- 「OK」を押すとキャプチャー起動コマンドを今すぐ実行するかコマンドを保存するかを選択する画面に行く

GUIを使用したCaptureの起動



- 必要な情報を入力
- 「OK」を押すとキャプチャー起動コマンドが発行される
- スクリプトとして保管も可能

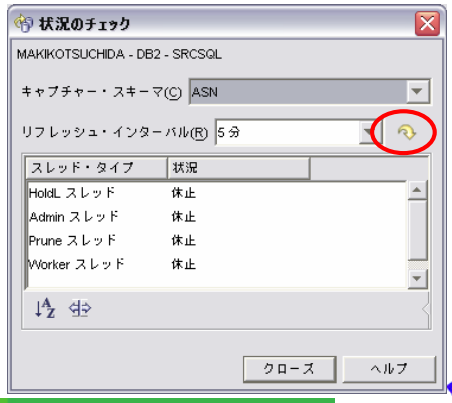
Captureの状況のチェック

■ コマンドの場合
- asnccmd CAPTURE_SERVER status

```
C:¥DpropR>asnccmd srcsql status
2005-07-04-10.31.01.460000 ASN0520I  "AsnCcmd" : "ASN" : "Initial" : STATUS コマンド応答: "HoldLThread" スレッドは "is resting" 状態にあります。
2005-07-04-10.31.01.470000 ASN0520I  "AsnCcmd" : "ASN" : "Initial" : STATUS コマンド応答: "AdminThread" スレッドは "is resting" 状態にあります。
2005-07-04-10.31.01.470000 ASN0520I  "AsnCcmd" : "ASN" : "Initial" : STATUS コマンド応答: "PruneThread" スレッドは "is resting" 状態にあります。
2005-07-04-10.31.01.480000 ASN0520I  "AsnCcmd" : "ASN" : "Initial" : STATUS コマンド応答: "WorkerThread" スレッドは "is resting" 状態にあります。
```

■ GUIの場合
- レプリケーションセンター
- 「SQLレプリケーション」→「操作」→「キャプチャー・コントロール・サーバー」→（サーバー上で右クリック）→「状況のチェック」
- 黄色矢印ボタンで更新する

（※ asnccmd コマンドシンタックスはCaptureno操作を参照）



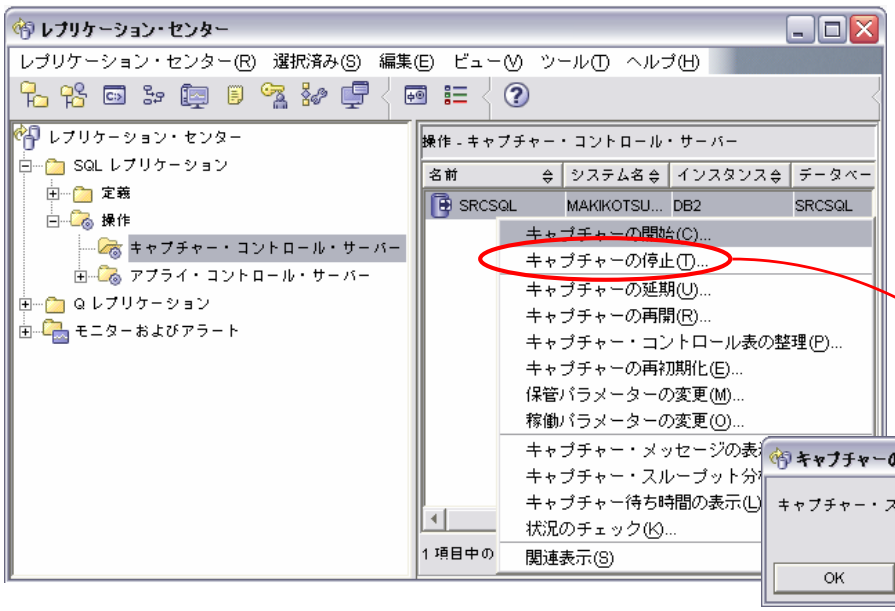
コマンドを使用したCaptureの停止

■ asnccmd コマンドを使用
- 停止例　：　asnccmd CAPTURE_SERVER stop

（※ asnccmd コマンドシンタックスはCaptureの操作を参照）

GUIを使用したCaptureの停止

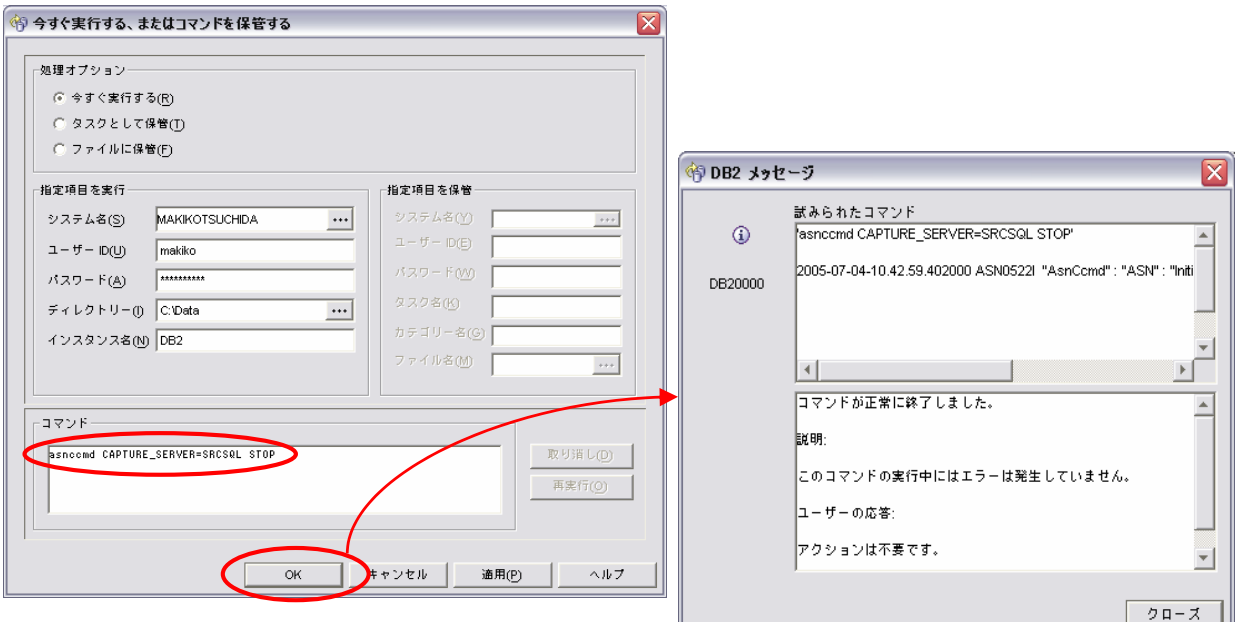
- レプリケーションセンターを起動
- 「SQLレプリケーション」→「操作」→「キャプチャー・コントロール・サーバー」→（サーバー上で右クリック）→「キャプチャーの停止」



- キャプチャー・スキーマの指定（デフォルトはASN）

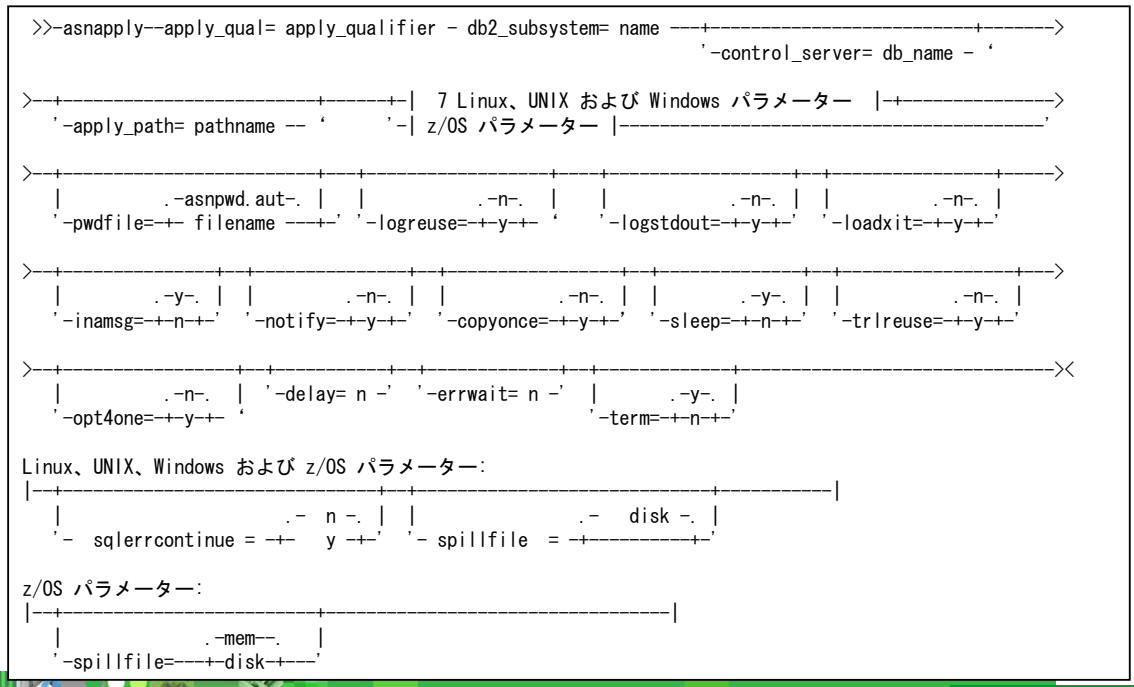
GUIを使用したCaptureの停止

- 「OK」を押すとキャプチャー起動コマンドが発行される
- スクリプトとして保存することも可能



コマンドを使用したApplyの起動

■ asnapply コマンドを使用
- 起動例 ： asnapply QUALFIER APPLY_SERVER（＋オプション）



コマンドを使用したApplyの起動

■ Capture起動時入力パラメーター

パラメーター	デフォルト値	説明
apply_qual	なし(指定必須)	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。アプライ修飾子名は最大18文字
db2_subsystem	なし(z/OSでは指定必須)	z/OSの場合のみ:DB2サブシステムの名前を指定。入力するDB2サブシステム名は最大4文字
control_server	UNIX,Windows : DB2DBDFT環境変数 z/OS :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	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
apply_path	asnapplyが呼び出されたディレクトリー	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
pwdfile	asnpwd.aut	パスワード・ファイルの名前を指定
logreuse	N	アプライ・プログラムが、ログ・ファイルを再利用するかを指定します。
logstdout	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 N :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する 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)の両方に送信する
loadxit	N	アプライ・プログラムがASNLOADを呼び出すかどうかを指定
inamsg	Y	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
notify	N	アプライ・プログラムがASNDONEを呼び出すかどうかを指定
copyonce	N	アプライ・プログラムがサブスクリプション・セットごとに、コピー・サイクルを1回だけ実行し終了するかどうかを指定

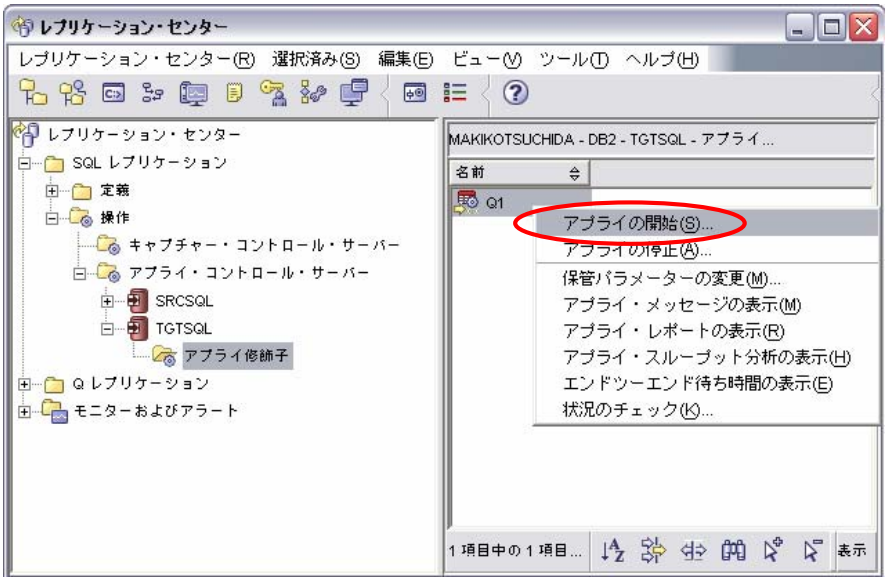
コマンドを使用したApplyの起動

■ Capture起動時入力パラメーター

パラメーター	デフォルト値	説明
sleep	Y	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する Y : アプライ・プログラムはスリープ状態に入る N : アプライ・プログラムは停止する
trlreuse	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IBMSNAP_APPLYTRAIL表を空にするかどうかを指定
opt4one	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する。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する。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する
delay	6	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（0～6）
errwait	300	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
term	Y	DB2の状況が、アプライ・プログラムが終了するかDB2の始動を待つかを指定
Linix, UNIX, Windows および z/OS パラメーター		
sqlerrcontinue	N	アプライ・プログラムが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
spillfile	disk	フェッチした応答セットをどこに保管するかを指定 （disk）
z/OS パラメーター		
spillfile	mem	フェッチした応答セットをどこに保管するかを指定 (mem/disk)

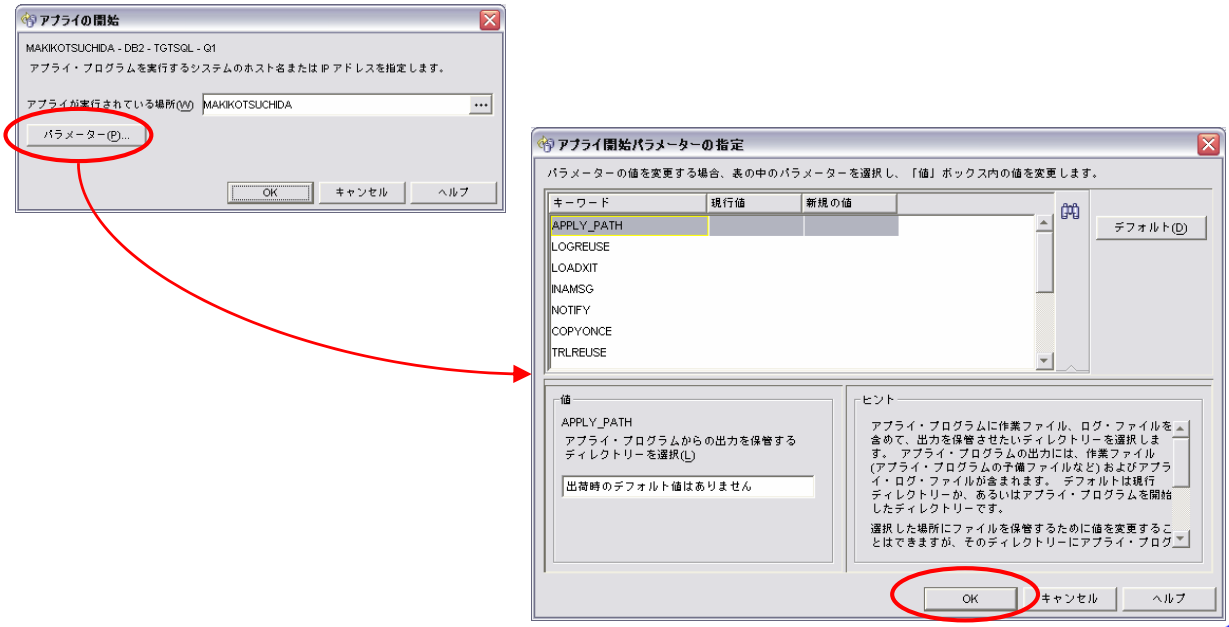
GUIを使用したApplyの起動

■ レプリケーションセンターを起動
■ 「SQLレプリケーション」→「操作」→「アプライ・コントロール・サーバー」→「アプライ修飾子」→（アプライ修飾子名を右クリック）→「アプライの開始」



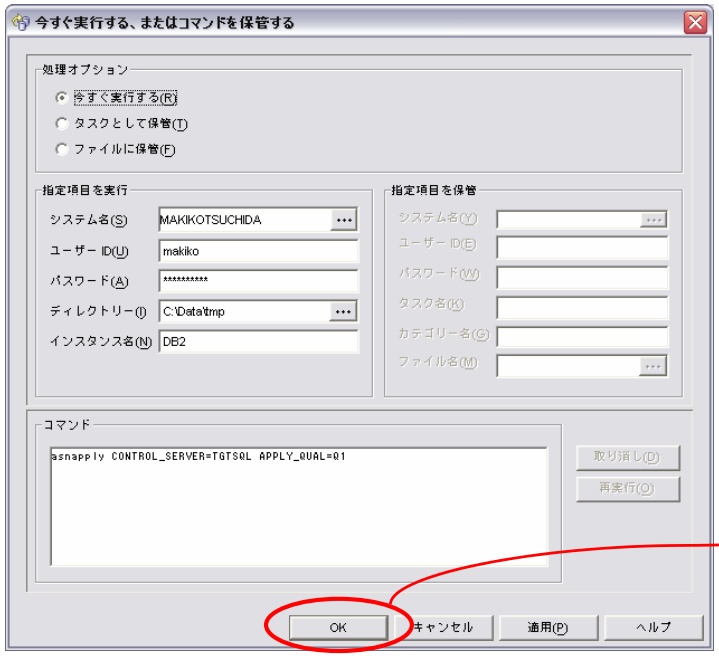
GUIを使用したApplyの起動

- アプライ・プログラムを実行するシステムを選択し、必要に応じてパラメーターを変更

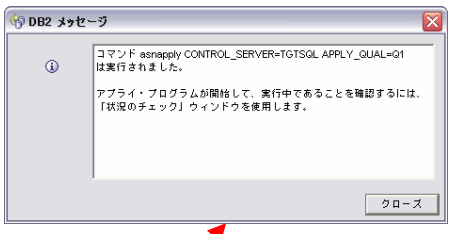


GUIを使用したApplyの起動

- 必要に応じてパラメーターを変更



- 必要な情報を入力
- 「OK」を押すとアプライ起動コマンドが発行される
- スクリプトとして保管も可能



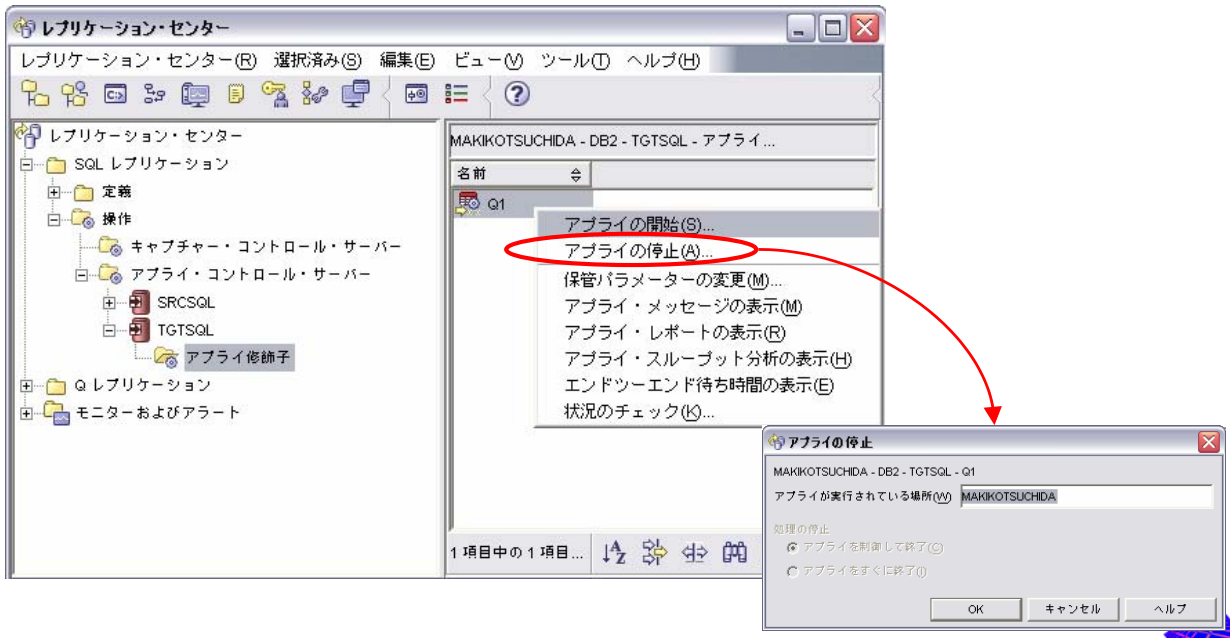
コマンドを使用したApplyの停止

■asnacmd コマンドを使用

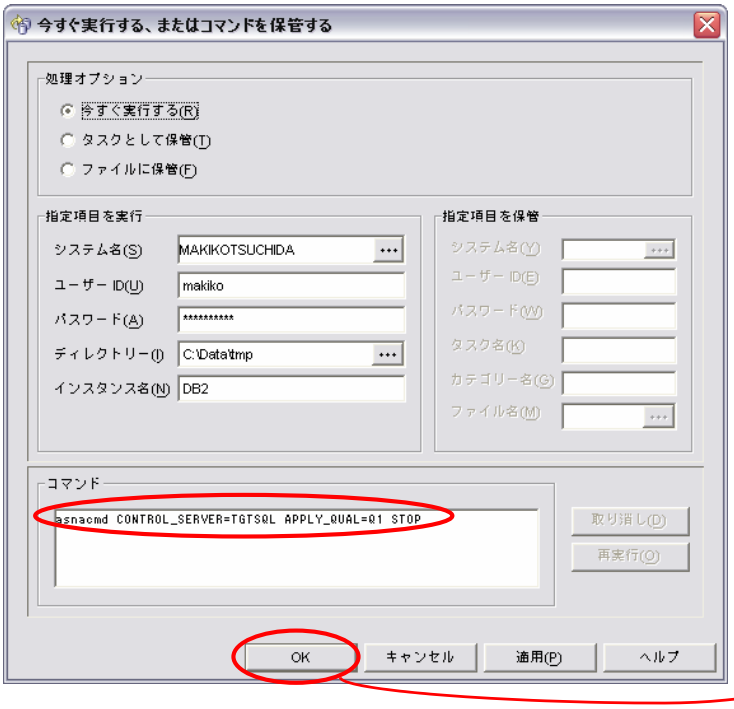
- 停止例 ： asnacmd APPLY_SERVER stop
  （※ asnacmd コマンドシンタックスはApplyの操作を参照）

GUIを使用したApplyの停止

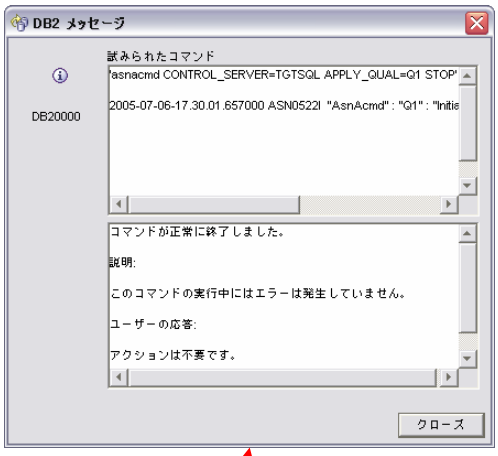
- レプリケーションセンターを起動
- 「SQLレプリケーション」→「操作」→「アプライ・コントロール・サーバー」→「アプライ修飾子」→（アプライ修飾子名を右クリック）→「アプライの停止」



GUIを使用したApplyの停止



- 「OK」を押すとキャプチャー起動コマンドが発行される
- スクリプトとして保存することも可能



Captureの操作

asnccmd コマンドによって稼動中のCaptureを操作する

- 実行可能な操作
  - Capture起動中のパラメーター値の変更
    - 一部のパラメータのみ、次ページ参照
  - プルーニング
    - CD表から、アプライ済みの不要データをDELETEする
    - デフォルトはインターバル間隔でのプルーニング(デフォルト間隔=300秒)だが、コマンドの発行によるプルーニングも可能
  - 現行パラメーター値の確認
  - 制御表再読み込み
    - Capture起動中に追加した定義を認識させるためなどに使用する
  - キャプチャーの中断/再開
  - キャプチャーの状況チェック
    - 各スレッドの状況を確認する
  - キャプチャーの停止

asnccmd コマンド

■ コマンドシンタックス

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

>--+-------------------------+--+-chgparms--| パラメーター |-+--->< 
   '-capture_schema= schema -'   +-prune----------------------+
                                 +-qryparms-------------------+
                                 +-reinit---------------------+
                                 +-resume---------------------+
                                 +-status---------------------+
                                 +-stop-----------------------+
                                 '-suspend
```

パラメーター	説明
chgparms	現行キャプチャーの使用パラメーター値を変更（次ページ参照）
prune	現時点でCD表からプルーニング対象となっている行を除去
qryparms	キャプチャーが現在使用しているパラメーター値を確認
reinit	キャプチャープログラム実行中に既存の登録情報を変更した場合に変更内容をキャプチャーに認識させるために発行する
resume	キャプチャーの再開
status	キャプチャーの状況のチェック。各スレッドの状況が表示される
stop	キャプチャーの停止
suspend	キャプチャーの中断

asnccmd コマンド

● コマンドシンタックス（chgparmsのパラメーター）

```
|--+-------------------+--+-----------------+----------------------->
   |            .-y-.  |  |           .-n-.  |
   '-autoprune=-+-n-+-'  '-autostop=-+-y-+-'

>--+---------------------+--+-----------------+---------------------->
   '-commit_interval= n -'  |           .-n-.  |
--                          '-logreuse=-+-y-+-'

>--+-------------------+--+-------------------+---------------------->
   |            .-n-.  |  '-memory_limit= n -'
   '-logstdout=-+-y-+-'

>--+----------------------+--+--------------------+------------------->
   '-monitor_interval= n -'  '-monitor_limit= n -'

>--+---------------------+--+----------------------+------------------->
   '-prune_interval= n -'  '-retention_limit= n -'

>--+---------------------+--+-------------+--+-------------------+-----|
   '-sleep_interval= n -'  |       .-y-.  |  '-trace_limit= n -'
                           '-term=-+-n-+-'
```

※各パラメーターの意味については、6,7ページ「Captureの起動」を参照

IBMSNAP_SIGNAL表

- SIGNAL表を使用する目的
  - キャプチャーに対して特定のアクションを指示するために使用する
- Capture-ApplyのHandshaking
  - V7のPC表に変わりV8からAPPLYは変更開始シグナルを挿入
- Captureへの制御
  - 特定の表に対する変更収集を開始する/変更収集を停止する(使用例①)
  - Captureを停止する
  - Update Anywhere実行時のデータループ防止
  - 特定の変更までレプリケーションする為にApplyに正確なログ番号を提供(使用例②)
    - EVENT表のEND_SYNCHPOINTを利用する
- USERシグナル、CMDシグナル
  - USERシグナルは特定のログ番号を提供(V7 Capture "GETLSEQ"コマンドと同等)
    - SIGNAL_STATE-P(Pending)/R(Received)
  - CMDシグナル
    - CAPSTART/CAPSTOP
    - STOP
    - UPDANY
      - シグナルタイプがユーザーの場合はSIGNAL_SUBTYPEは認識されない

解説:

- IBMSNAP_SIGNAL表の列

列名	説明
SIGNAL_TIME	シグナルの処理が完了した時刻
SIGNAL_TYPE	通知されたシグナルのタイプ CMD:ユーザー、アプライ、システムコマンドなどから通知されたシグナル USER:ユーザーから通知されたシグナル
SIGNAL_SUBTYPE	SIGNAL_TYPE=CMDの時にキャプチャーが実行するアクション CAPSTART:特定のソース表の変更キャプチャーを開始する STOP:キャプチャー・プログラムは停止する CAPSTOP:特定のソース表の変更キャプチャーを停止する UPDANY:アプライ・プログラムが、UpdateAnyware構成であることをキャプチャーに通知する
SIGNAL_INPUT_IN	SIGNAL_TYPE=USERの時:ユーザー入力の内容が保持される SIGNAL_TYPE=CMDの時: CMD+CAPSTART :IBMSNAP_PRUNCNTL表に入力されているソース表のマッピングID CMD+UPDANY :UpdateAnyware構成でのアプライを識別するアプライ修飾子 CMD+CAPSTOP :変更キャプチャーを停止するソース表スキーマおよびソース表名
SIGNAL_STATE	そのシグナルの状況を示す値 P:シグナルがペンディング状態で、まだキャプチャーはシグナルを受け取っていない R:キャプチャーがシグナルを受け取った状態。キャプチャーは、SIGNAL_TYPE=USERであるか SIGNAL_TYPE=CMDかつSIGNAL_SUBTYPE=STOPである場合は、STATEをRにする。 C:キャプチャーがシグナルの処理を完了した状態。SIGNAL_TYPE=CMDであり、 SIGNAL_SUBTYPE=STOP以外の場合は、キャプチャーは処理を完了するとSTATEをCにする。
SIGNAL_LSN	コミットレコードのログ・シーケンス番号。キャプチャーにより入力される値が入る。

SIGNAL表の使用例①

■ 特定のソース表のキャプチャーの停止

- 一部のソース表からのキャプチャーを停止したい場合、CAPSTOPシグナルを利用する
- IBMSNAP_SIGNAL表へCAPSTOPシグナルをInsertする
- キャプチャーは、REGISTER表とPRUNCNTL表を更新し、差分収集を停止する
- この操作は、単純に差分収集を停止、開始できるものではないので注意。
  - 一度非活動化したソース表を再活動化するには、フルリフレッシュが必要となる
  - 再活動化に関しては、フルリフレッシュを参照

■ CAPSTOPシグナル

```
INSERT INTO CAPSCHEMA.IBMSNAP_SIGNAL
    (signal_type, signal_subtype, signal_input_in, signal_state)
    VALUES
    ('CMD', 'CAPSTOP', スキーマ名.ソース表名, 'P');
```

■ CAPSTOPシグナル発行時のキャプチャー実行ログ

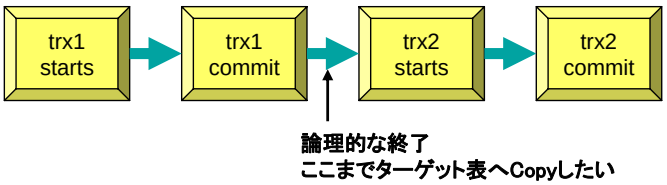
```
2005-10-24-14.13.04.542635 ASN0076I  CAPTURE "ASN" : "WorkerThread". Capture has stopped capturing
      changes for source table "MAKITV82"."TEST1" in response to a CAPSTOP signal.
2005-10-24-14.25.13.415894 ASN8999D  "Capture" : "ASN" : "WorkerThread" : trans::insertTx: row for
      CAPSTOPed table[tid=2,fid=2] not processed
2005-10-24-14.25.13.416041 ASN8999D  "Capture" : "ASN" : "WorkerThread" : trans::insertTx: row for
      CAPSTOPed table[tid=2,fid=2] not processed
2005-10-24-14.25.13.416080 ASN8999D  "Capture" : "ASN" : "WorkerThread" : trans::insertTx: row for
      CAPSTOPed table[tid=2,fid=2] not processed
```


解説:

- 登録済みオブジェクト(ソース表)の削除をしたい場合など、オブジェクトを非活動化する必要があります。また、このオブジェクトで一時的に変更のキャプチャーを停止しても、他の登録済みオブジェクトに対してはキャプチャー・プログラムを実行し続けておきたい場合も、登録済みオブジェクトを非活動化できます。
- キャプチャー・プログラムは、非活動化されたソース・オブジェクトについては変更のキャプチャーを停止します。しかし、これらのソース・オブジェクトに関連する変更データ (CD) 表、登録属性、およびサブスクリプション・セットは、システム上に残ります。
- 登録済みオブジェクトを非活動化する前に、この登録済みオブジェクトに関連付けられたすべてのサブスクリプション・セットを非活動化する必要があります。これにより、ユーザーがオブジェクトを削除する、または再度活動化する準備が整う前に、アプライ・プログラムがオブジェクトを自動的に再活動化し、非活動化処理に介入してくることを防止できます。
- オブジェクトが非活動化され、DB2 レプリケーションがそのオブジェクトに対する変更のキャプチャーを停止すると、登録済みオブジェクトに関連付けられたすべてのサブスクリプション・セットが影響を受けます。これらのサブスクリプション・セットの実行を続けたい場合は、この登録済みオブジェクトをソースとして使用するサブスクリプション・セット・メンバーを、非活動化されたサブスクリプション・セットから除去する必要があります。

SIGNAL表の使用例②

- 特定の変更までのレプリケーション
  - SIGNAL表を利用することで、Applyに正確なログ番号を提供し、特定の変更までレプリケーションすることが可能
  - EVENT表のEND_SYNCHPOINTを利用する



- ApplyをEVENT起動すると、IBMSNAP_SUBS_EVENT表のEND_SYNCHPOINTのログ・シーケンス番号に値するトランザクションまでを適用する
  - 事前に、EVENT起動のための設定をしておくことが必要
    - IBMSNAP_SUBS_SET表　REFRESH_TYPE='E' or 'B', EVENT_NAME='EVENT名'
- 手順
  - SIGNAL表にトリガーを作成
  - trx1のcommit後にSIGNAL表へInsertを実行
  - SIGNAL表のトリガーが起動し、EVENT表へEND_SYNCHPOINTをInsert

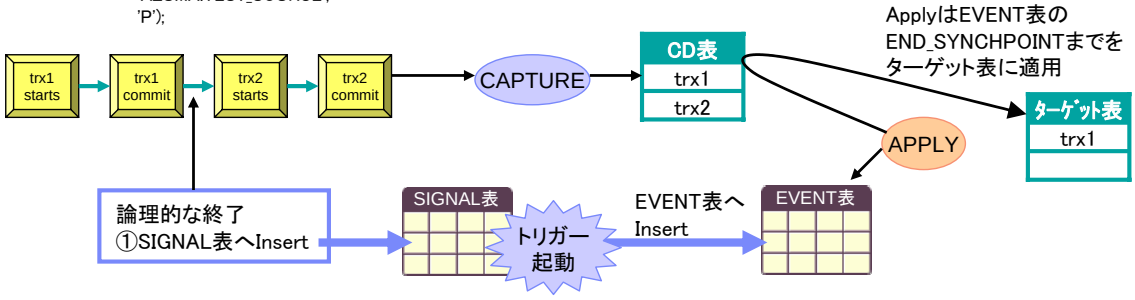
解説：

- EVENT起動　update ASN.IBMSNAP_SUBS_SET set REFERSH_TYPE='E' , EVENT_NAME='AZUMAS'
- SQL Insertステートメントを EVENT表へ実行するトリガーをSIGNAL表に作成

```
create trigger trig1
  after update on asn.ibmsnap_signal
  referencing new as n
  for each row mode db2sql
  when (n.signal_subtype='AZUMAS')
  insert into asn.ibmsnap_subs_event values
  ('EVENT1', CURRENT TIMESTAMP+1 MINUTES, N.SIGNAL_LSN,NULL)%
```

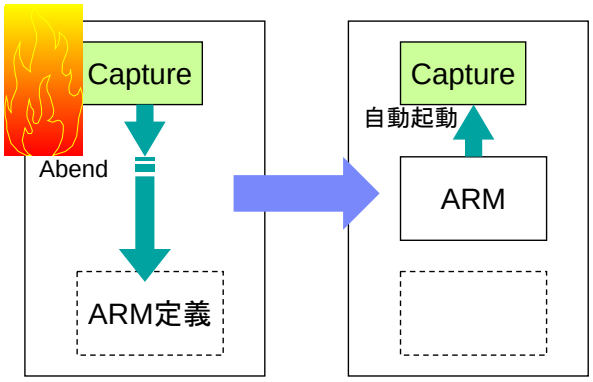
- 適用させたいトランザクションの終了後に、SQL Insertステートメントを SIGNAL表へ実行する（USERシグナル）

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



Capture / Apply for z/OS ARM Support

- ARM (Automatic Restart Manager)のサポート
  - Capture, ApplyがFailした場合に、それを自動検知し起動するサービス
    - IXCARM Macroを提供
    - ARM は、特定のバッチ・ジョブまたは開始タスクの可用性を改善するための MVS リカバー関数です。ジョブまたはタスクが失敗するか、ジョブやタスクを実行しているシステムに障害が発生した場合、ARM はオペレーターの介入なしに、ジョブまたはタスクを再始動できます
    - MVS ARM が使用可能な各アプリケーションは、自分自身についてユニークなエレメント名を生成し、ARM とのすべての連絡にこの名前を使用します。ARM はエレメント名をトラッキングし、エレメント名に対して再始動ポリシーを定義します。



解説:

- ARMポリシーの定義例

```
//IBMUSERP JOB ,,MSGCLASS=X,NOTIFY=IBMUSER
//*
//ST01    EXEC PGM=IXCMIAPU
//STEPLIB   DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
     DATA TYPE(ARM) REPORT(YES)
       DEFINE POLICY NAME(POLARM1)
         REPLACE(YES)

         RESTART_GROUP(*)
           RESTART_PACING(60)
           ELEMENT(*)
           RESTART_ATTEMPTS(1,300)
           TERMTYPE(ELEMTERM)
//
```

すべて対象
Captureエレメント名
　　ASNTCxxxxyyyy
Applyエレメント名
　　ASNAMxxxxyyyy
(PTF UQ79622 要)

xxxx DB2サブシステム名
yyyy データ共有メンバー名

解説：

■ ARM定義の確認

```
D XCF,ARMS,DETAIL
IXC392I  10.51.51  DISPLAY XCF 900
ARM RESTARTS ARE ENABLED
--------------- ELEMENT STATE SUMMARY ---------------  -TOTAL-  -MAX-
STARTING  AVAILABLE  FAILED  RESTARTING  RECOVERING
       0         15       0           0           0       15     20
RESTART GROUP:DEFAULT          PACING :   60    FREECSA:    0      0
ELEMENT NAME :ASNTCD71A        JOBNAME :CAPSTATC STATE   :AVAILABLE
  CURR SYS :ZOS1               JOBTYPE :JOB       ASID    :002B
  INIT SYS :ZOS1              JESGROUP:EPLEX    TERMTYPE:ELEMTERM
  EVENTEXIT:*NONE*              ELEMTYPE:*NONE*   LEVEL   :    2
  TOTAL RESTARTS :        0    INITIAL START:09/18/2003 10:51:38
  RESTART THRESH :   0 OF 1    FIRST RESTART:*NONE*
  RESTART TIMEOUT:      300    LAST  RESTART:*NONE*
RESTART GROUP:DEFAULT          PACING :   60    FREECSA:    0      0
```


解説：

■ ARM定義の確認

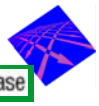
```
CANCEL CAPSTATC,ARMRESTART

IEA989I SLIP TRAP ID=X222 MATCHED.  JOBNAME=CAPSTATC, ASID=002B.
ASN8004D  "Capture" : "ASN" : Thread "Initial" received "Handled" signal
 "SIGABND".
ASN8008D  "Capture" : "ASN" : "Destroyed" IPC queue with key(s)
"1a072d48".
IEE301I CAPSTATC          CANCEL COMMAND ACCEPTED
.........
IEF450I CAPSTATC ASNCAP - ABEND=S222 U0000 REASON=00000000 943
        TIME=10.56.26
$HASP395 CAPSTATC ENDED
$HASP309 INIT 3    INACTIVE ******** C=ABC
IXC812I JOBNAME CAPSTATC, ELEMENT ASNTCD71A FAILED. 946
THE ELEMENT WAS RESTARTED WITH PERSISTENT JCL.
ICH70001I AZUMA    LAST ACCESS AT 10:51:21 ON THURSDAY, SEPTEMBER 18, 2003
$HASP373 CAPSTATC STARTED - INIT 3    - CLASS A - SYS ZOS1
IEF403I CAPSTATC - STARTED - TIME=10.56.26
ASN0546W  "Capture" : "ASN" : The program call issued to the 950
Automatic Restart Manager failed. The invoked IXCARM macro is
"READY", the return code is "0", and the reason code is "0".
```

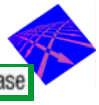
Applyの操作

- asnacmd コマンドによって稼動中のApplyを操作する
  - 実行可能な操作
    - キャプチャーの状況チェック
    - アプライの停止
  - コマンドシンタックス

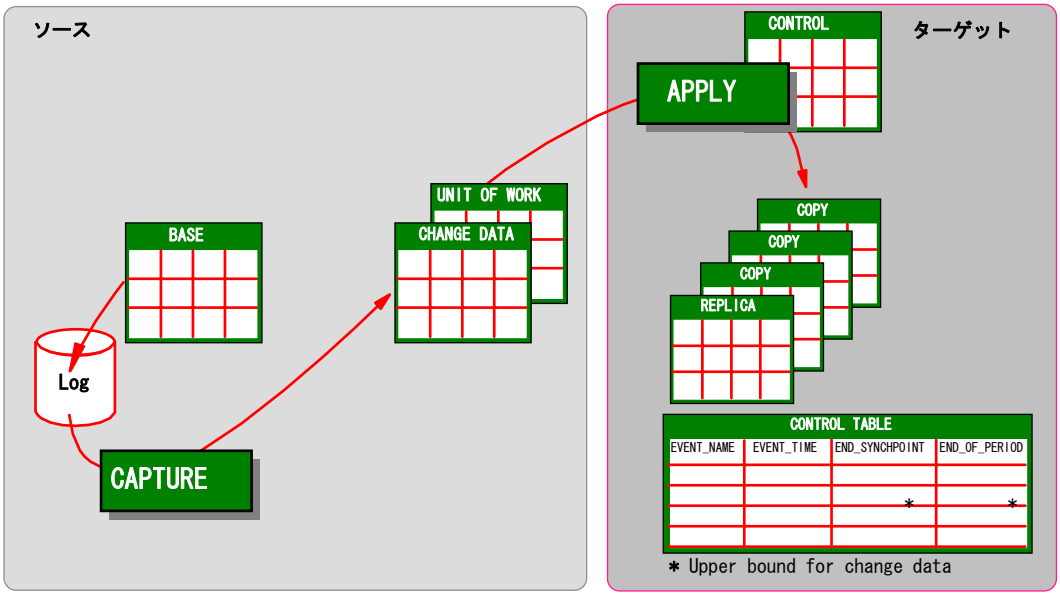
```
>>-asnacmd--apply_qual= apply_qualifier ------------------------->

>--+-------------------------+--+-status-+---------------------><
   '-control_server= db_name -'  '-stop---'
```

パラメーター	説明
status	アプライの状況のチェック。各スレッドの状況が表示される
stop	アプライの停止





EVENT表



Applyを7/17午後20:00にStartさせ、16:00までのデータを反映させたい場合

```
INSERT INTO ASN.IBMSNAP_SUBS_EVENT
 (EVENT_NAME, EVENT_TIME, END_OF_PERIOD) VALUES
 ('EVENT1', '2005-07-17-20:00', '2005-07-17-16:00' ) ;
```


解説:

■ イベント・タイミングとは

イベント・タイミングとは、イベントによって起動されるコピーのことです。
コントロール・センターでサブスクリプションを定義するときにイベント名を指定し、その後、アプリケーションまたはユーザーに、サブスクリプション・イベント表にそのイベント名のタイム・スタンプを挿入させます。Applyプログラムがイベント値を検出すると、複製サブスクリプションの複製を開始します。同じサブスクリプションについてイベント・ベース・タイミングと間隔タイミングを使用することができます。 サブスクリプション・イベント表 ASN.IBMSNAP_SUBS_EVENT には、上記で示されているように、3つの列があります。

EVENT_NAME	EVENT_TIME	END_OF_PERIOD	END_SYNCHPOINT
END_OF_DAY	2005-07-01-17:00:00.000000	2005-07-01-15:00:00.000000	

- EVENT_NAME は、複製サブスクリプションの定義時に指定するイベントです。コントロール・センターは、複製サブスクリプションが定義された後、この列をサブスクリプション・セット表に挿入します。
- EVENT_TIME は、Apply プログラムが複製サブスクリプションの処理を開始する時刻を示すタイム・スタンプです。
- END_OF_PERIOD は、ここに指定した時刻より後のトランザクションのコピーが据え置かれることを指定するオプション値です。EVENT_TIME は、アプライ・コントロール・サーバーのクロックにより設定されますが、END_OF_PERIOD はソース・サーバーのクロックにより設定されます。これらの 2つのデータベースは、時間帯の異なるサーバーに置かれている可能性があるため、この区別は重要です。
  上記例では、'END_OF_DAY' は複製サブスクリプションの定義時に指定したイベント名です。タイム・スタンプ値 2005-07-01-17:00:00.000000 は、Apply プログラムが複製サブスクリプションの処理を開始する時刻です。タイム・スタンプ値 2005-07-01-15:00:00.000000 は、変更が複製されなくなるトランザクション時刻です。
- END_SYNCHPOINTはそのログシーケンス番号まで反映したい値を指定し、END_OF_PERIODと両方指定された場合はEND_SYNCHPOINTが優先されます。

プルーニングのタイミング

- インターバルによるプルーニング
  - デフォルト（300秒）
    - AUTOPRUNE=Y（デフォルト）
      - パラメーターAUTPPRUNE=Yでキャプチャーが起動されているときに有効。
      - AUTOPRUNE値は、IBMSNAP_CAPPARMS表のAUTOPRUNE列もしくは、キャプチャーの起動パラメーターで指定する
    - インターバル間隔
      - IBMSNAP_CAPPARMS表のPRUNE_INTERVALもしくは、キャプチャーの起動パラメーターで指定する（単位：秒）
  - 定期的にCD表、UOW表、IBMSNAP_SIGNAL表、IBMSNAP_CAPMON表、IBMSNAP_CAPTRACE表の不要な行をDELETEする
    - メリット
      - 自動的に行われるため、コマンド発行のための開発などが不要
      - 数分間隔でプルーニングすることでCD表、UOW表に大量にデータを溜め込まずに済むため、スペースの節約になる
    - デメリット
      - 業務時間中もインターバル起動するような場合、プルーニングの分だけ処理コストが大きくなり、DBに与える負荷が大きくなる

解説：

- プルーニング、整理 (pruning)レプリケーションにおいて、キャプチャー・プログラム、Q キャプチャー・プログラム、アプライ・プログラム、または Q アプライ・プログラムが使用するレプリケーション・コントロール表またはログ・ファイルから、古いデータを除去するタスク。

- キャプチャーコントロール表のうちプルーニング対象となる表
  - CD表
  - UOW表
  - IBMSNAP_SIGNAL表
  - IBMSNAP_CAPMON表
  - IBMSNAP_CAPTRACE表

- AUTOPRUNE
  - キャプチャー・プログラムが、不必要になった行を CD 表、UOW 表、IBMSNAP_SIGNAL表、IBMSNAP_CAPTRACE表、およびIBMSNAP_CAPMON表から自動的に除去するかどうかを示すフラグ。
    Y = 自動プルーニングはオン（デフォルト）
    N = 自動プルーニングはオフ

- PRUNE_INTERVAL
  - IBMSNAP_CAPPARMS表もしくは、起動パラメーターで指定する。起動パラメーターで指定された場合は、CAPPARMS表の設定値がオーバーライドされる。
  - 意味
    - キャプチャー・プログラムが、不必要になった CD 表、UOW 表、IBMSNAP_SIGNAL表、IBMSNAP_CAPTRACE表、およびIBMSNAP_CAPMONの行を自動的に除去する（AUTOPRUNE = Y の場合のみ）頻度（秒単位）。除去するインターバルを短くするとスペースは節約できますが、処理コストが増加します。除去するインターバルを長くすると CD 表および UOW 表のスペースはより多く必要になりますが、処理コストは減少します。

プルーニングのタイミング

■コマンドによる任意のタイミングでのプルーニング

- コマンドを発行したタイミングでプルーニングを行う
- asnccmd *capture_server* prune で実行する

➢ メリット
- プルーニングを行うタイミングを任意で指定できる（例えば、夜間の業務時間帯以外など）
- 業務外時間に行うなどすることで、DB上で実行される業務アプリケーションに負荷を与えずに済む

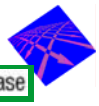
➢ デメリット
- プルーニングコマンドを発行するための仕組みを作成する必要がある
- プルーニング間隔を大きくした場合は、CD表やUOW表に、その間隔分データがたまるため、表スペース容量に考慮が必要

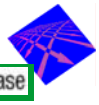
解説：

■ コマンドによって、プルーニングするようasnccmdコマンドを発行した場合、CD表、UOW表、IBMSNAP_CAPMON表、IBMSNAP_CAPTRACE表、および IBMSNAP_SIGNAL 表の整理を 1 回だけ行います。

フルリフレッシュの操作

- ASNLOAD
  - FP10新機能 Nickname Target
- マニュアル・フルリフレッシュ
- 強制フルリフレッシュ

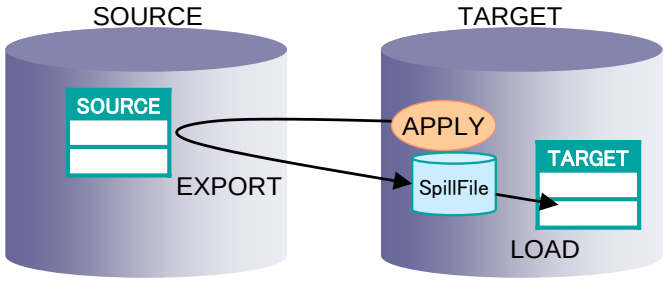




ASNLOAD

■ 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（ソース表からの全件フルリフレッシュ）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。APPLY起動時にLOADXIT 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。
  - asnapply {アプライ修飾子名} {コントロールサーバ名}LOADXIT
- 差分COPYには使用できない
- Target-DBのLOG量を軽減可能
- Active LOG Full(SQL0964)回避



解説：

■ 省略時ではAPPLYはリフレッシュを実行するときASNLOADを使用しません。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します。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ます。

■ ASNLOADの構成情報は、Configuration File(ASNLOAD.INI)で設定可能です。

■ ASNLOADのカストマイズ、USERID,PASSWORDの指定が可能です。

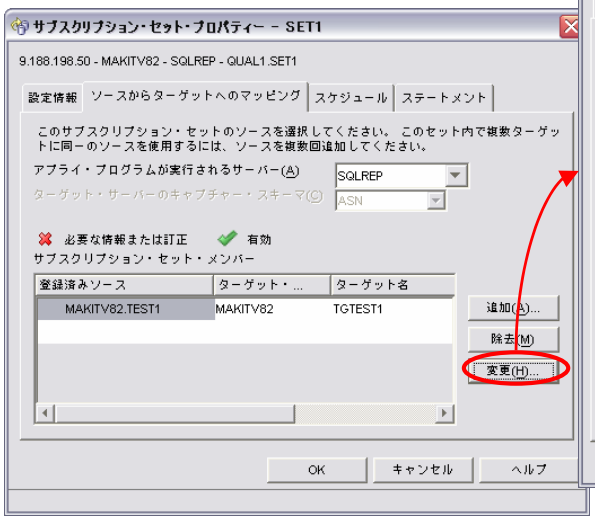
■ SUBS_MEMBR表のLOADX_TYPEの設定値によって呼び出されるユーティリティが換わります。

- NULL
  - 適切なユーティリティーが選択されます
- 1
  - ASNLOADを呼び出しません
- 2
  - ユーザーが提供するロジックを呼び出します
- 3
  - カーソルを使用したロードを行います（クロスローダー）
- 4
  - EXPORTとLOADユーティリティーを使用します
- 5
  - EXPORTとIMPORTユーティリティーを使用します

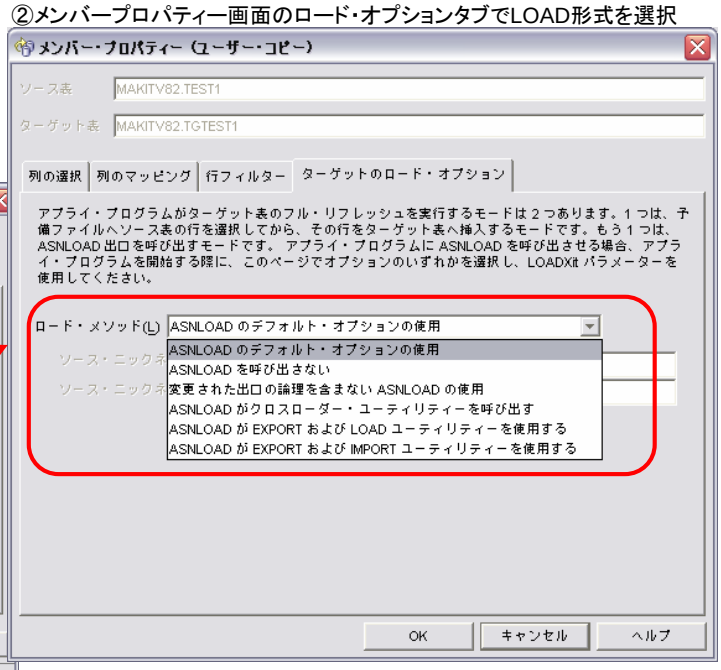
ASNLOADの設定変更（GUI）

- FP2よりGUI選択可能
- 定義情報で、ASNLLOADの使用を指定

①サブスクリプション・セットの設定外面から、メンバーを選択し「変更」

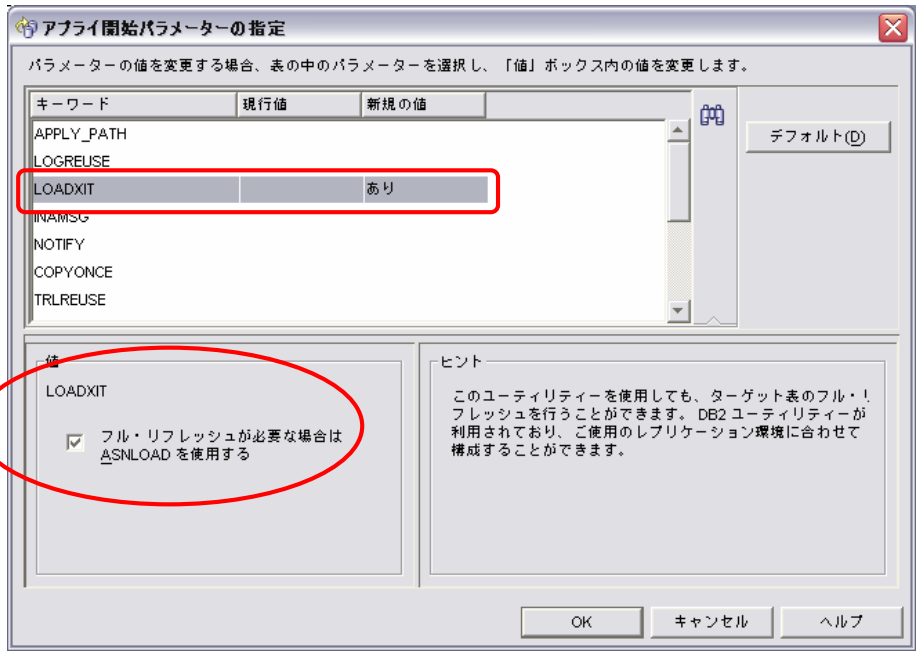


②メンバープロパティー画面のロード・オプションタブでLOAD形式を選択



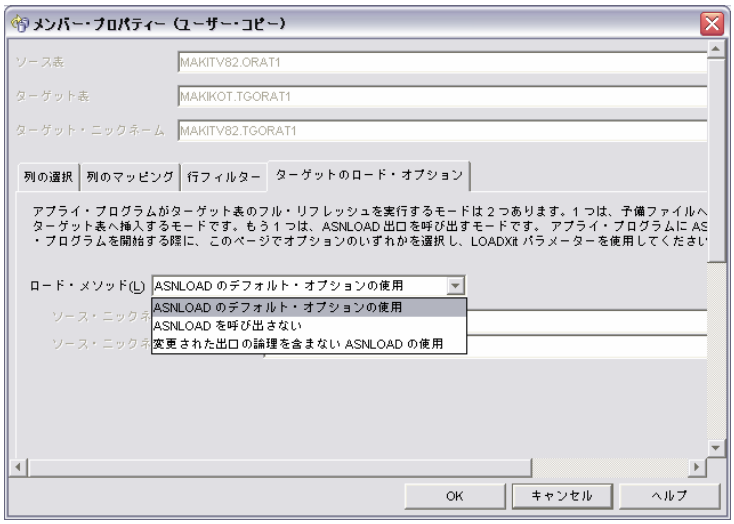
ASNLOAD実行（GUI）

- FP2よりGUI選択可能
- GUIからのアプライの開始時にLOADXITの使用を指定



異種DB使用時のASNLOAD

- ASNLOAD
  - DB2のユーティリティを使用したフルリフレッシュが実施可能
  - 前提
    - ターゲット表の列は、ソース表の順序 と データ・タイプと一致する。
    - ターゲット表は、レプリケーション・マッピングの一部である列のみを含む。
    - ターゲット表は空でなければいけない
      - Import Replaceが使用できないため



解説:

- ターゲット表がニックネームの場合は、Export/Importが使用可能
  ただしGUIからはタイプを選べないため、SUBS_MEMBR表のLOADX_TYPE の手動変更が必要
  - NULL :適切なユーティリティーが選択されます
  - 1 :ASNLOADを呼び出しません
  - 2 :ユーザーが提供するロジックを呼び出します
  - 3 :カーソルを使用したロードを行います(クロスローダー)
  - 4 :EXPORTとLOADユーティリティーを使用します
  - 5 :EXPORTとIMPORTユーティリティーを使用します

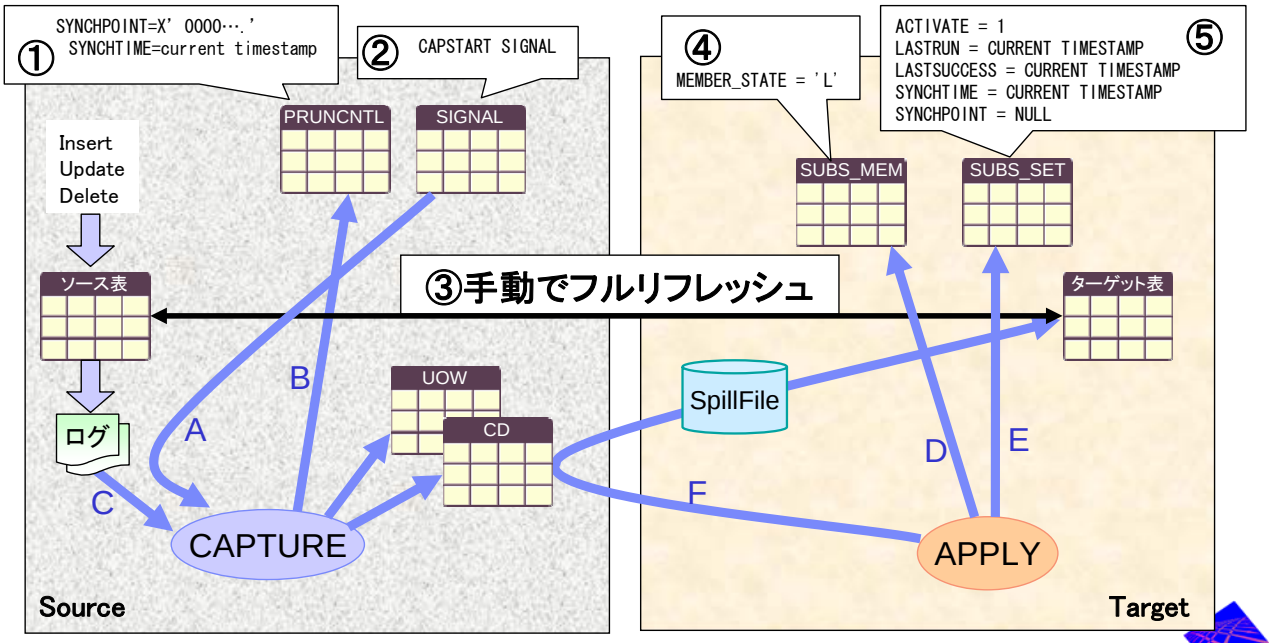
  - Fixpak9まで
    - OracleからDB2へのレプリケーションの場合には1，2，3のみの使用が可能です。
      4，5の場合ASNLOADの戻りコードASNLOAD_INVALID_LOADXTYPE(111)で失敗します
    - DB2からOracleへのレプリケーションの場合には1，2のみの使用が可能です。
      ターゲットがニックネームのため3，4，5はASNLOADの戻りコード
      ASNLOAD_TARGETTAB_INCOMPATIBLE_LOADXTYPE_4(113)
      ASNLOAD_TARGETTAB_INCOMPATIBLE_LOADXTYPE_5(114)で失敗します

  - Fixpak10
    - OracleからDB2へのレプリケーションの場合にはすべての呼び出しが可能です。
      4、5の設定はレプリケーション・センターから出来ません。IBMSNAP_SUBS_MEMBRのLOADX_TYPE列を手動で更新してください。
      GG04615　DPROPR APPLY ASNLOAD TO SUPPORT EXPORT FROM NICKNAMES

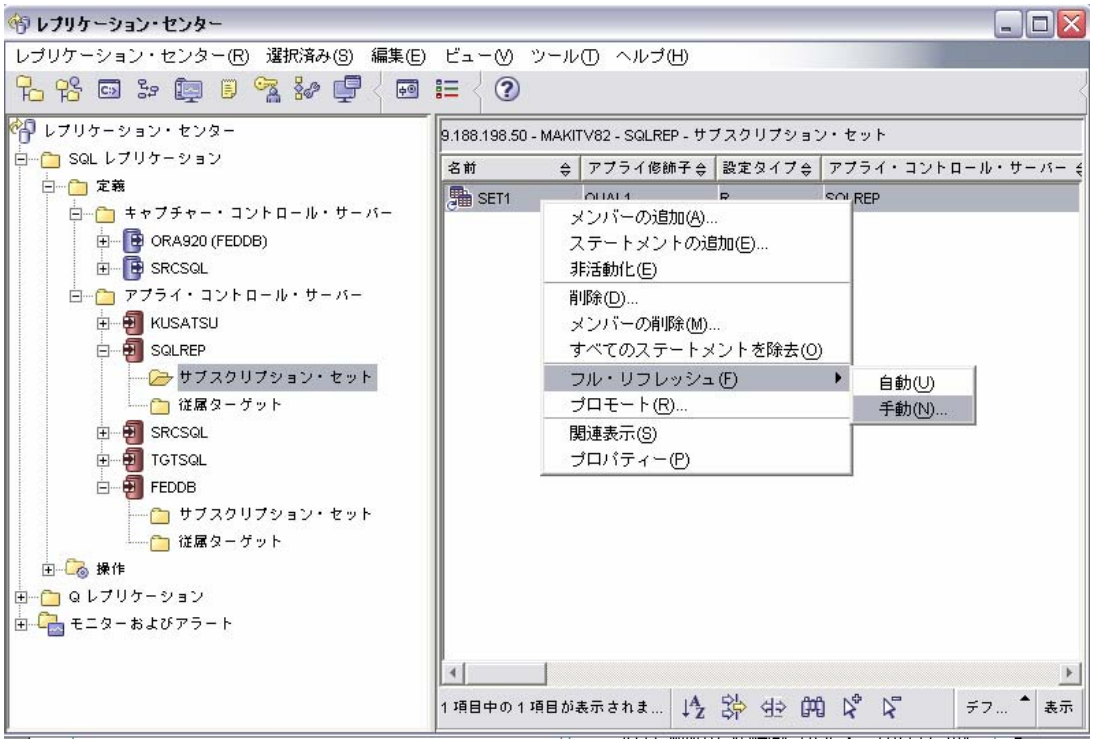
    DB2からOracleへのレプリケーションの場合には1，2、5の使用が可能です。
    ターゲットがニックネームのため3，4はASNLOADの戻りコード
    ASNLOAD_TARGETTAB_INCOMPATIBLE_LOADXTYPE_4(113)で失敗します

マニュアル・フルリフレッシュ（手動フルリフレッシュ）

■ APPLYにフルリフレッシュを行わせず、差分収集からキャプチャー、アプライをスタートする方法(p57,58の説明を参照)



GUIからフルリフレッシュスクリプトを生成、実行



解説：

レプリケーション・センターから生成される手動フルリフレッシュのスクリプト（左）と解説（右）（1/2）
キャプチャーコントロールサーバ側で実行するもの

```
UPDATE ASN.IBMSNAP_REGISTER SET DISABLE_REFRESH = 1
    WHERE  SOURCE_OWNER = 'MAKITV82' AND
    SOURCE_TABLE = 'TEST1'  AND SOURCE_VIEW_QUAL = 0;

UPDATE ASN.IBMSNAP_PRUNCNTL SET SYNCHPOINT = NULL,
    SYNCHTIME = NULL WHERE SOURCE_OWNER = 'MAKITV82'
    AND SOURCE_TABLE = 'TEST1' AND SOURCE_VIEW_QUAL =
    0 AND APPLY_QUAL = 'QUAL1' AND SET_NAME = 'SET1' AND
    TARGET_OWNER = 'MAKITV82' AND TARGET_TABLE =
    'TGTEST1' AND TARGET_SERVER = 'SQLREP';
```

①
```
UPDATE ASN.IBMSNAP_PRUNCNTL SET SYNCHPOINT =
  X'00000000000000000000', SYNCHTIME = CURRENT
  TIMESTAMP WHERE SOURCE_OWNER = 'MAKITV82' AND
  SOURCE_TABLE = 'TEST1' AND SOURCE_VIEW_QUAL = 0
  AND APPLY_QUAL = 'QUAL1' AND SET_NAME = 'SET1' AND
  TARGET_OWNER = 'MAKITV82' AND TARGET_TABLE =
  'TGTEST1' AND TARGET_SERVER = 'SQLREP';
```

②
```
INSERT INTO ASN.IBMSNAP_SIGNAL (SIGNAL_TIME,
    SIGNAL_TYPE, SIGNAL_SUBTYPE, SIGNAL_INPUT_IN,
    SIGNAL_STATE) VALUES(CURRENT TIMESTAMP, 'CMD',
    'CAPSTART', '0', 'P');
```

- REGISTER表のDISABLE_REFRESHを1にセット
  この値をセットしたソース表は、自動フルリフレッシュができなくなる
- PRUNCNTL表のSYNCHPOINTとSYNCHTIMEをNULLにセット
- PRUNCNTL表のSYNCHPOINTにX'0000….'、SYNCHTIMEに現在時刻をセット

A SIGNAL表にCAPSTARTをInsert。SIGNAL表にはDATA CAPTURE CHANGES属性がついているので、キャプチャーはログから、CAPSTARTシグナルを収集する

B このときPRUNCNTL表のSYNCHPOINTには、差分収集スタート時のLSNをUPDATEし

C CAPSTARTシグナルがINSERTされた時刻からの差分収集を開始する

解説：

レプリケーション・センターから生成される手動フルリフレッシュのスクリプト（左）と解説（右）（2/2）
アプライコントロールサーバ側で実行するもの　※必ずCAPSTARTシグナル発行後に実行すること

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

④
```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
        SET MEMBER_STATE = 'L'
        WHERE SET_NAME = 'SET1' AND APPLY_QUAL =
        'QUAL1' AND WHOS_ON_FIRST = 'S';
```

⑤
```
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'
        AND WHOS_ON_FIRST = 'S';
```

- サブスクリプションセットのACTIVATEを0にセットして、サブスクリプションを非活動化しておく。アプライが停止している場合は必須ではない

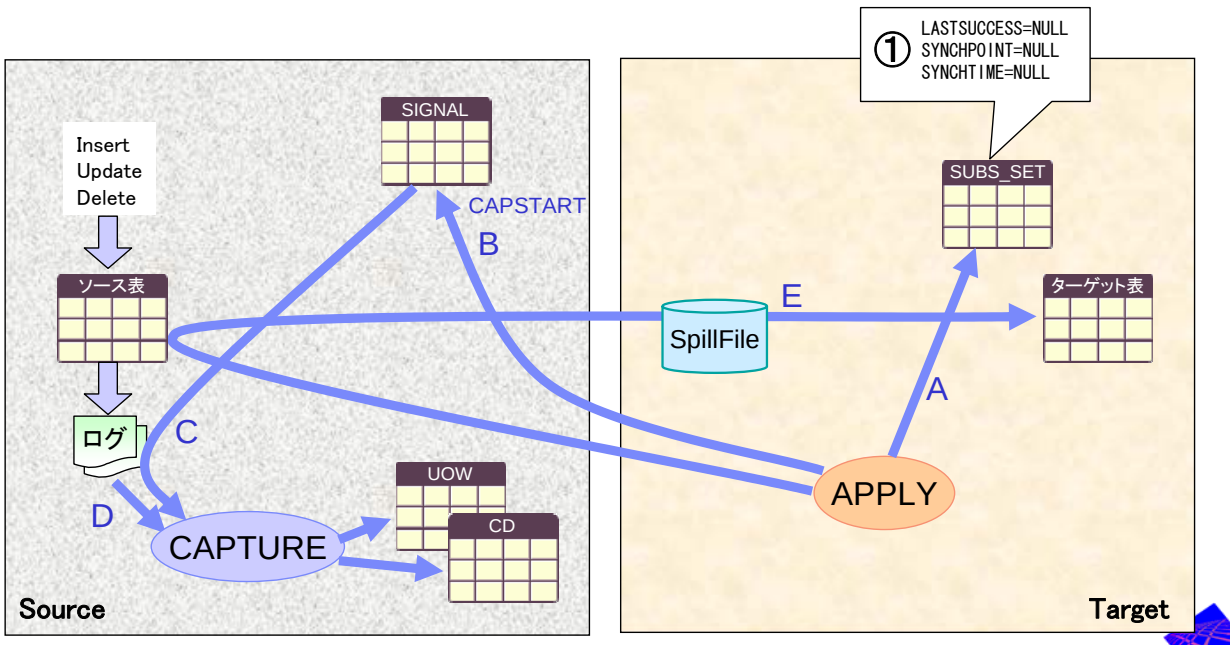
D SUBS_MEMBR表のMEMBER_STATEを 'L'（Loaded）にセット。このことで、アプライは、フルリフレッシュ（ターゲット表へのLOAD）直後だと認識する

E SUBS_SET表をUPDATEすることで、アプライがCD表から差分をターゲット表に反映できる形にする。サブスクリプションもACTIVATE=1にUPDATEし、再活動化している

F アプライはこの後、差分をターゲットに反映する。COPYONCE起動の場合は起動時に、インターバル起動の場合は④で設定されたLASTSUCCESS＋SLEEP_MINUTESの時間になるとターゲットへ差分をアプライする

強制自動フルリフレッシュ

■ 任意のサブスクリプションセットだけを強制的にフルリフレッシュする方法(p60の説明を参照)



解説:

レプリケーション・センターから生成される自動フルリフレッシュのスクリプト(左)と解説(右) (2/2)
アプライコントロールサーバ側で実行するもの ※必ずCAPSTARTシグナル発行後に実行すること

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

③

```
UPDATE ASN.IBMSNAP_SUBS_SET
      SET LASTSUCCESS=NULL,
      SYNCHPOINT=NULL,
      SYNCHTIME=NULL
      WHERE APPLY_QUAL='QUAL1'
      AND SET_NAME='SET1' AND WHOS_ON_FIRST='S';
```

```
UPDATE ASN.IBMSNAP_SUBS_
      SET SET ACTIVATE=1
      WHERE SET_NAME='SET1'
      AND APPLY_QUAL='QUAL1'
      AND WHOS_ON_FIRST='S';
```

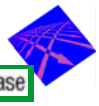
■ 強制フルリフレッシュするサブスクリプションセットのACTIVATEを0にセットして、サブスクリプションを非活動化しておく。アプライが停止している場合は必須ではない

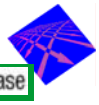
■ SUBS_SET表のSYNCHPOINTとSYNCHTIMEをNULLにするとアプライに強制的にフルリフレッシュを行わせることができる

A サブスクリプションセットを再活動化。SUBS_SET表のSYNCHPOINTとSYNCHTIMEのNULL値を見て、アプライは、
B まず、キャプチャーに対してCAPSTARTを発行する
C SIGNAL表にはDATA CAPTURE CHANGES属性がついているので、キャプチャーはCAPSTARTシグナルをログから読み取る。
D キャプチャーの差分収集開始。
E (フルリフレッシュ形式がEXPORT/LOADの場合)アプライはソース表を分離レベルCSでフェッチし、APPLY_PATHにSpillFileを作成してターゲット表にLOADを行う

障害対応

- 障害時に考慮するケースのフロー
- キャプチャー
  - キャプチャーエラー時の動作
  - キャプチャーのエラーで考慮すべきこと
  - CD表挿入時のエラーとRegistrationの再活動化
- アプライ
  - アプライエラー時の動作
- データベース単位のクラッシュ時
  - ターゲット・データベースクラッシュ時
  - ソース・データベースクラッシュ時
- HACMP使用時の考慮点

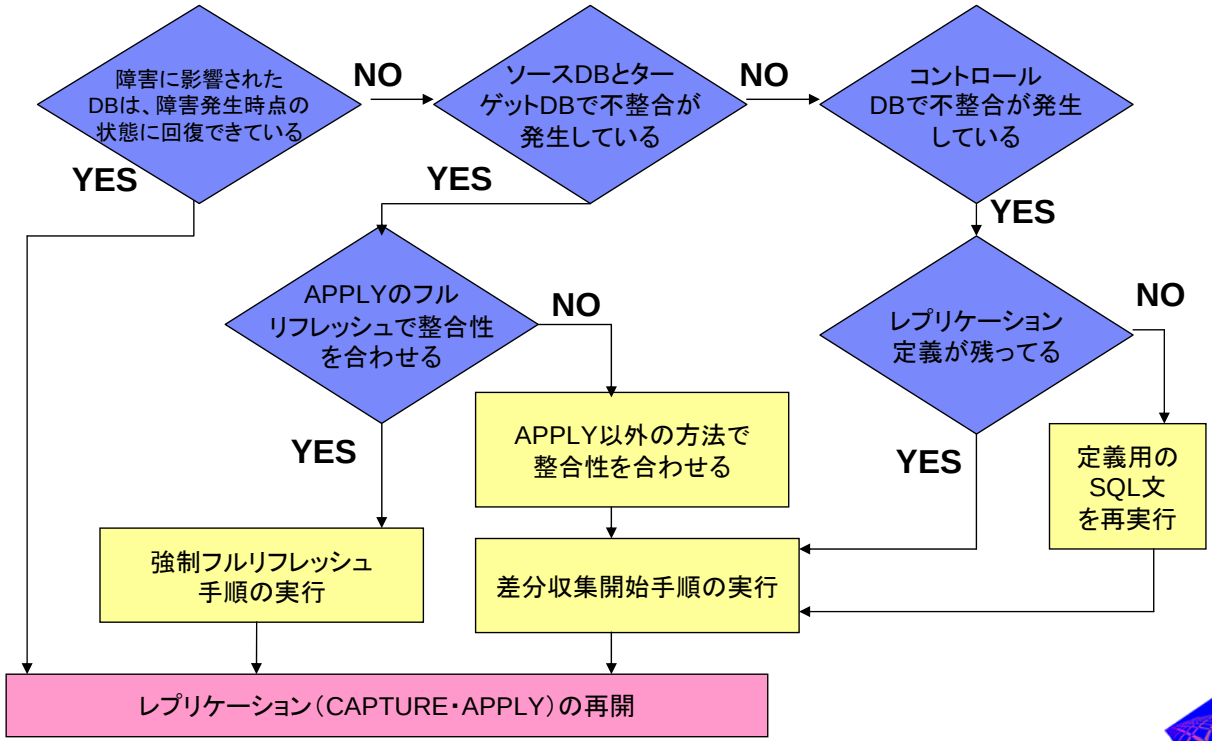




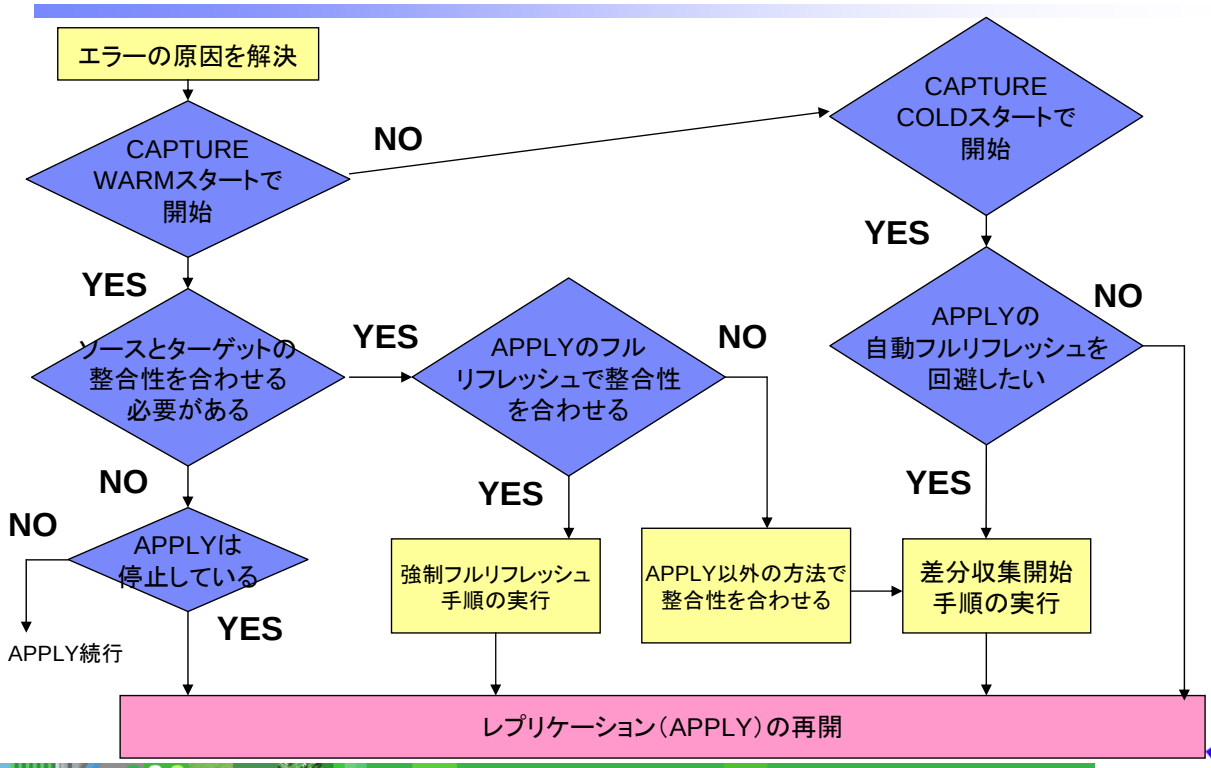
障害対応

- システム障害、ソースDB、ターゲットDB、コントロールDBいずれかの障害
  これらの障害が発生しても、データベースの観点で、障害発生時点までロールフォワード・リカバリーができていればレプリケーションはそのまま継続可能。
  - ソースDB、ターゲットDBのどちらかの表が、ある過去の状態にしか回復できず、ソース表とターゲット表の内容に不整合が発生してしまった場合は、以下の選択がある。
    - –フルリフレッシュからやり直す
    - –ユーザー責任でソース表とターゲット表の整合を合わせ（例えばUNLOAD・LOADなどで）、レプリケーションは差分収集から開始する
  - またソース表、ターゲット表は整合性が保たれているが、レプリケーションの制御表のみ壊れてしまった、というような場合はレプリケーション定義が残っていれば、上記の差分収集から開始する方法で制御表を更新すれば、レプリケーションを再開することが可能。レプリケーション定義が残っていなくても、最初に定義したときのスクリプト（SQL文）を保管しておけば、それを実行後、同じように差分収集から開始可能
- CAPTUREの障害時
  CAPTUREが起動しない、途中で停止した場合
  - その原因を調査し、解決できれば、そのまま差分収集を継続することが可能
  - ただし、CAPTUREをCOLDスタートした場合、その後に処理するAPPLYは、FULLREFRESHを行おうとする。よって、APPLYを起動する前に差分収集から開始するためのフルリフレッシュ回避の処理を行う必要がある
  - またCAPTUREがとまっていた時間が長く、CAPTUREが再開してもレプリケーションが追いつくまでに非常に時間がかかるというような場合は、上記の「ソース表とターゲット表の内容に不整合が発生してしまった場合」と同様の処理が必要になる
- APPLYの障害時
  APPLYが起動しない、途中で停止した場合。
  - その原因を調査し、解決できれば、そのまま差分収集を継続することが可能
  - ただしAPPLYがとまっていた時間が長く、APPLYが再開してもレプリケーションが追いつくまでに非常に時間がかかるというような場合は、上記の「ソース表とターゲット表の内容に不整合が発生してしまった場合」と同様の処理が必要
  - またこの間CAPTUREが稼動していれば、CD表にはデータが溜まってゆくため、CD表のスペースにも注意が必要
- 障害時の考え方のフローの例は次ページ

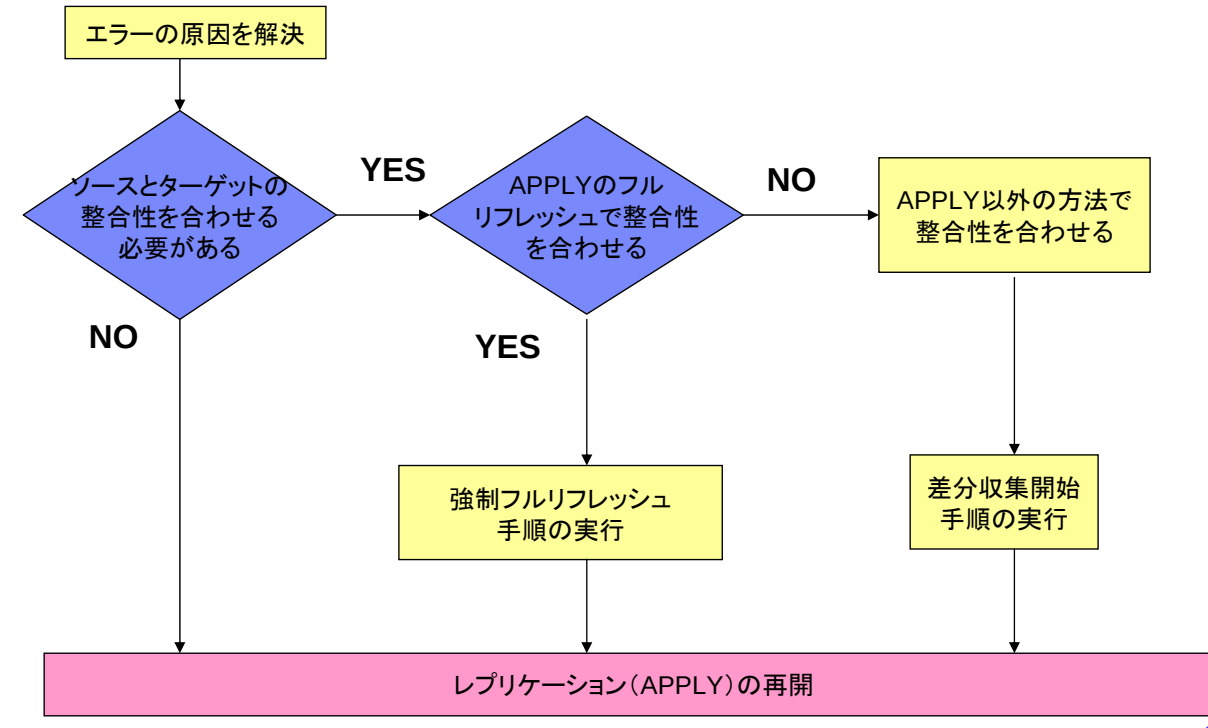
システム障害　DB障害発生



キャプチャー障害



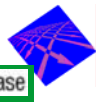
アプライ障害



Captureのエラー

- キャプチャーエラー時の動作
  - CD表、UOW表へのInsertに失敗した場合
    - 作業単位(UOW)をロールバックし、キャプチャーは停止する
  - 必要なログが読み込めない場合
    - キャプチャーは停止する。完了できない作業単位はロールバックする
  - IBMSNAP_CAPTRACE表のInsertに失敗した場合
    - IBMSNAP_TRACE表への書き込みは不可能だが、差分収集およびCD表、UOW表へのInsertは継続する
  - キャプチャー・プロセスを強制終了した場合
    - 作業単位(UOW)をロールバックし、キャプチャーは停止する

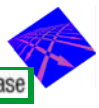
※いずれの場合もIBMSNAP_RESTART表に、キャプチャー終了時点のログ・シーケンス番号および時間を挿入して終了する。このため、再起動時にはキャプチャー終了時点の続きからキャプチャーを開始することが可能



Captureのエラーで考慮すべきこと

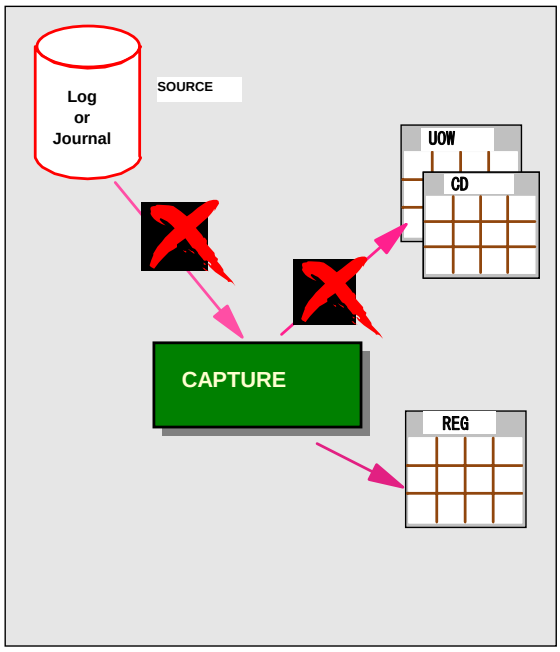
- LAG_LIMIT(IBMSNAP_CAPPAMRS表で設定)
  デフォルト:10080分(1週間)
  - ログ・レコードの処理時にキャプチャー・プログラムがシャットダウンせずにログ読み取りを遅らせることができる分数。
    キャプチャーが現在読んでいるログと、現在書き込まれているログの差(LAG)が1週間以上になると、フル・リフレッシュの方が経済的であるため、フルリフレッシュが行われる

- RETENTION_LIMIT(IBMSNAP_CAPPAMRS表で設定)
  デフォルト:10080分(1週間)
  - プルーニング対象でないデータが、CD 表、UOW 表、およびシグナル表の中に留まる時間の長さ。通常、CD 表および UOW表のデータは、ターゲットに適用された後でプルーニングされる。アプライされていないデータが、このRETENTION_LIMITにより削除された場合は、ソース表とターゲット表間にデータの不整合があるとみなされ、アプライはフルリフレッシュを行おうとする



CD表挿入時のエラーとRegistrationの再活動化

STOP_ON_ERRORとSTATE



- STOP_ON_ERROR
  - REGISTRATIONの属性のひとつ
  - Captureが始動、開始、再開始またはCD表へ挿入時にエラーを検出した場合のCaptureの処理を指示するフラグ

Y... (デフォルト)エラーを検出するとCaptureは停止
N... エラーを検出すると停止せず、STATEをS(STOP)にし、エラー情報をSTATE_INFORに書き出しCaptureは停止しない

STATEをSからAに変更するにはCOLDスタートではRESETされないので手動で変更する必要がある

- STATE
  - S...STOPPED(停止)
  - A...ACTIVE
  - I...非ACTIVE

解説:

- 予期しないエラーの為、CAPTUREが登録情報を非活動化した場合は登録を再活動する特別の処理を行う必要がある。
  - 予期しないエラーが発生するとSTOP_ON_ERROR列の値がNに設定されている場合、REGISTER情報のSTATE列の値をSに設定する。このSTATE列はCAPTUREがこの登録の処理を停止したことと登録の修復が必要である意味を持ち、APPLYは停止状態の登録に対してはCAPSTARTを発行することはありえない。
  - エラーの登録情報をノーマルへ戻す為には下記のSQLを発行し、CAPTUREのREINITあるいは停止、再始動を行う必要がある

```
update asn.ibmsnap_register set state='I'  where state='S' and source_table='TEST_SOURCE';
```

- STATEを'I'にUpdateすると、CAPTUREは差分収集をすることができない。その場合、次回APPLY起動時には、データの不整合を回復するために、フルリフレッシュが実行されるので注意
  - フルリフレッシュを避けたい場合
    - STOP_ON_ERRORはデフォルトの 'Y' に設定
    - または、手動でソース表とターゲット表の整合性をとる手動フルリフレッシュを行う

アプライのエラー

- アプライエラー時の動作
  - ターゲット表への適用に失敗した場合
    - サブスクリプション・セットごとに作業単位（UOW）をロールバックする
    - COPYONCE起動でない場合は、ERROR_WAITごとにリトライを行う
  - IBMSNAP_APPLYMON, IBMSNAP_APPLYTRAILへのInsertに失敗した場合
    - IBMSNAP_APPLYMON, IBMSNAP_APPLYTRAIL表へのInsertは行わないが、ターゲット表への反映は継続する
  - アプライ・パスにスピル・ファイル作成が作成できない場合
    - エラー終了する

※エラーの原因が除去されると、アプライはターゲット表への適用を再開することが可能
※COPYONCE起動の場合は、アプライ起動後にターゲット表へ反映が可能

データベース単位のクラッシュ時

■データベース単位でのクラッシュが発生した場合のSQLレプリケーションの復旧方法

■障害が発生した場合
- リストア、ロールフォワードによってクラッシュしたデータベースが最新に戻れば、キャプチャー、アプライの処理続行が可能
- ソース/ターゲットの整合性が崩れた場合、差分レプリケーションを開始する前に、ソースDBとターゲットDBのハンド・シェイキング（同期点の確立）が必要
  - 同期がとれたことをキャプチャー、アプライが認識しなければならない
  - 同期取得の方法は様々
    - リストア、ロールフォワードによりソース/ターゲットDBの同期をとる方法
    - ターゲットがクラッシュした場合、ソースDBからフルリフレッシュによって同期をとる方法、など

リカバリ方法一例

■ターゲット・データベース　クラッシュ時
- リストア/ロールフォワードによるリカバリ
- 自動フルリフレッシュによる全件リフレッシュ
- 手動フルリフレッシュによる全件リフレッシュ

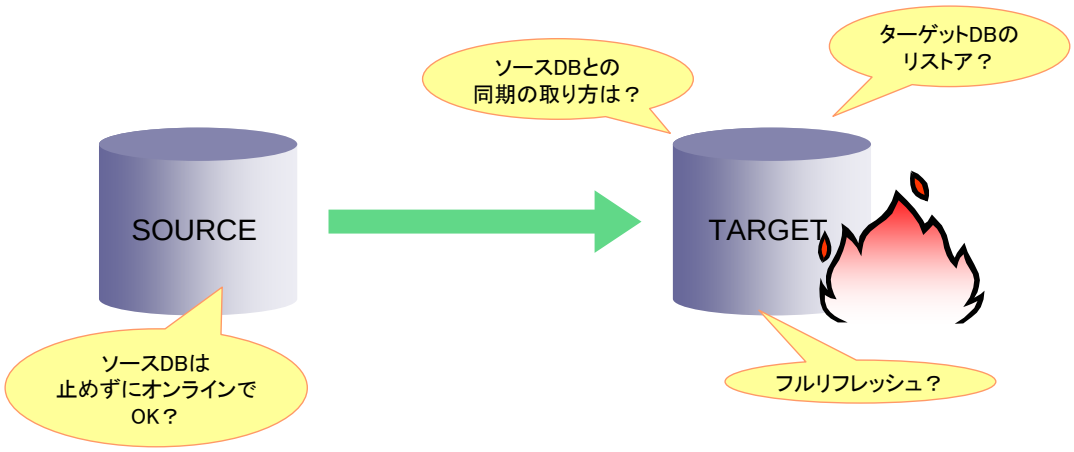
■ソース・データベース　クラッシュ時
- ソースDBリカバリ時のキャプチャーに関する注意点
- ターゲットDBのリカバリ方法

※注意
本資料に掲載した障害時のリカバリ方法は、SQLレプリケーションにおける障害回復の考え方と、限られたケースでのリカバリ例を挙げたもので、全ての環境でのデータベース、SQLレプリケーションシステムの回復を保障するものではありません。本番環境での回復方法については、各環境に合わせ検討した上で、十分なテストを行ってください。

ターゲット・データベース　クラッシュ時

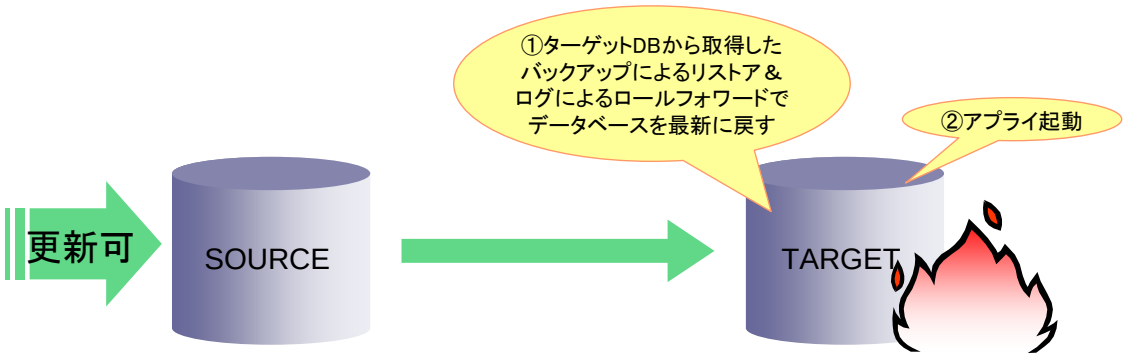
■リカバリ方法

- リストア/ロールフォワードによるリカバリ
- 自動フルリフレッシュによる全件リフレッシュ
- 手動フルリフレッシュによる全件リフレッシュ



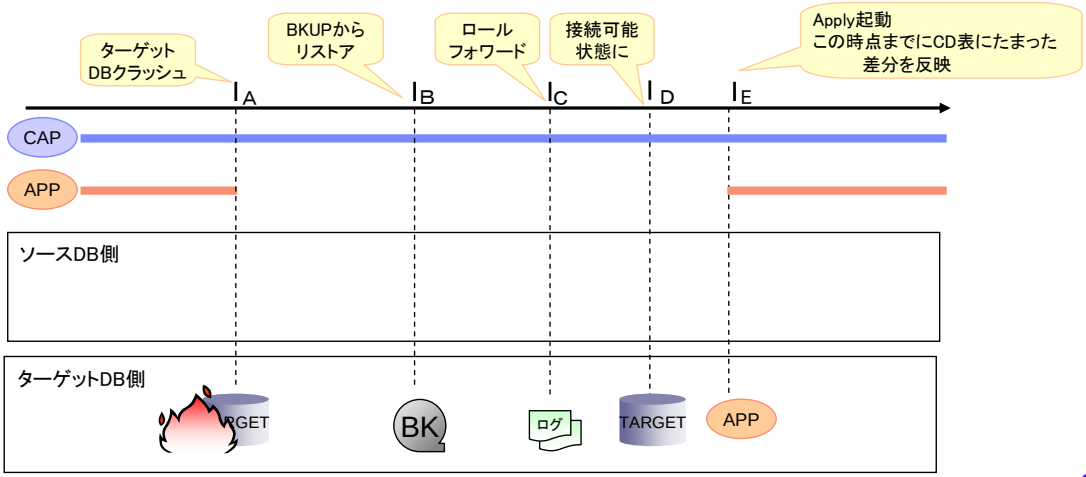
リストア/ロールフォワードによるリカバリ

- 概略
  - ターゲット・データベースのバックアップイメージからのリストア/ロールフォワード
- ソースDB
  - オンライン可能
- Capture
  - 稼動、停止　どちらも可能。
    - Captureを稼動させたままの場合
      - CD表にデータがたまるため、表スペース容量に注意が必要
    - Captureを停止する場合
      - 次回起動時までに必要なログを除去、移動してはならない（USEREXITによる移動であれば問題ない）



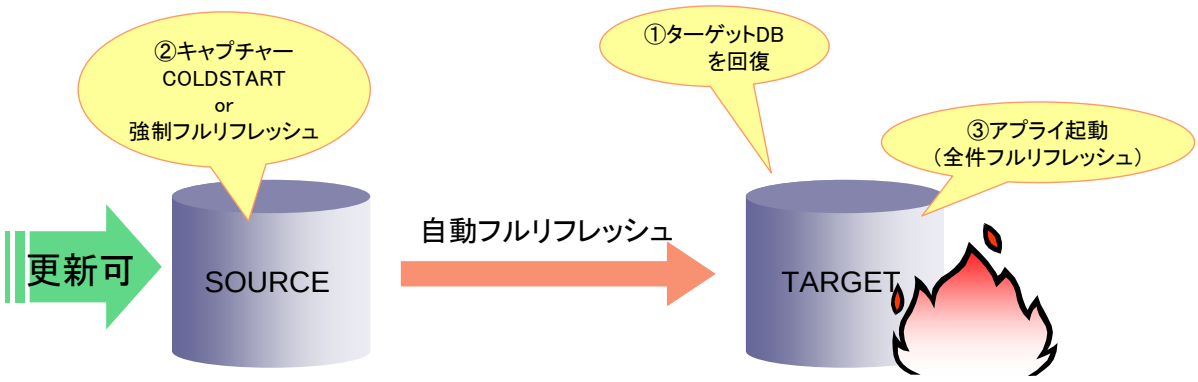
リストア/ロールフォワードによるリカバリ

- 概略
  - ターゲット・データベースのバックアップイメージからのリストア/ロールフォワード
- ソースDB
  - オンライン可能
- Capture
  - 稼動、停止　どちらも可能。
    - 稼動させたままの場合
      - CD表にデータがたまるため、表スペース容量に注意が必要
    - 停止する場合
      - 次回起動時までに必要なログを除去、移動してはならない（USEREXITによる移動であれば問題ない）



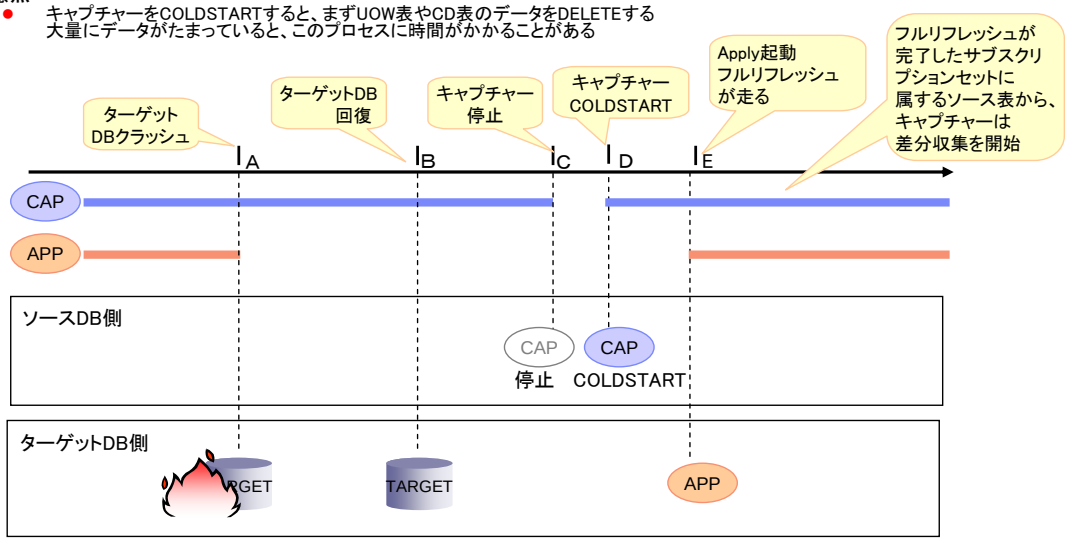
自動フルリフレッシュによる全件リフレッシュ

- 概略
  - ターゲットDBが最新まで戻らなかった場合や、ソースとターゲットの整合性が取れているかが不明な場合、ソースとターゲットの同期が必要
  - ターゲットDBを回復後、ソースDBから全件自動フルリフレッシュを行うことで同期を取得し、差分収集を開始可能
- ソースDB
  - オンライン可能
- Capture
  - 全ての登録ソース表フルリフレッシュしたい場合COLDSTARTでCaptureを再起動
  - 一部のサブスクリプションをフルリフレッシュしたい場合は強制フルリフレッシュ



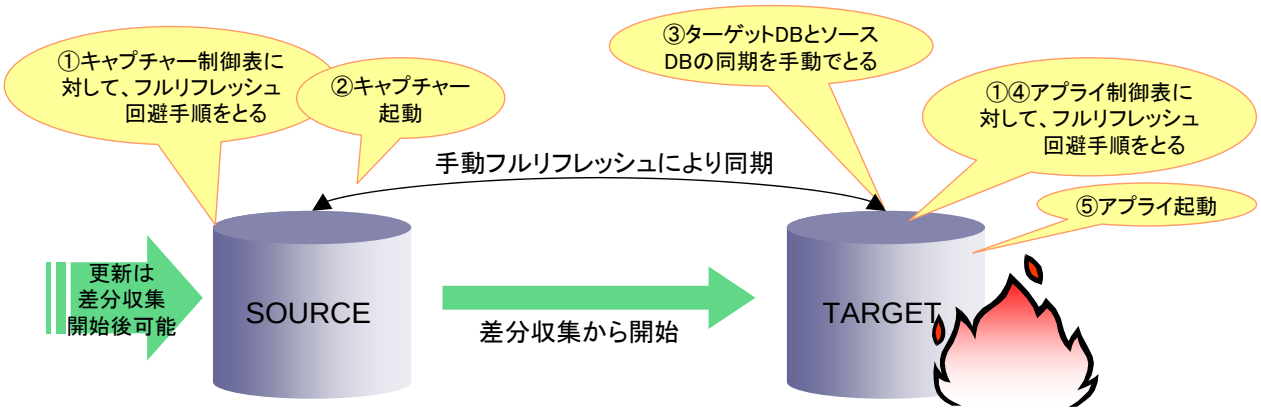
自動フルリフレッシュによる全件リフレッシュ

- 概略
  - ターゲットDBが最新まで戻らなかった場合や、ソースとターゲットの整合性が取れているかが不明な場合、ソースとターゲットの同期が必要
  - ターゲットDBを回復後、ソースDBから全件自動フルリフレッシュを行うことで同期を取得し、差分収集を開始可能
- ソースDB
  - オンライン可能
- Capture
  - COLDSTARTで再起動
- 考慮点
  - キャプチャーをCOLDSTARTすると、まずUOW表やCD表のデータをDELETEする
    大量にデータがたまっていると、このプロセスに時間がかかることがある



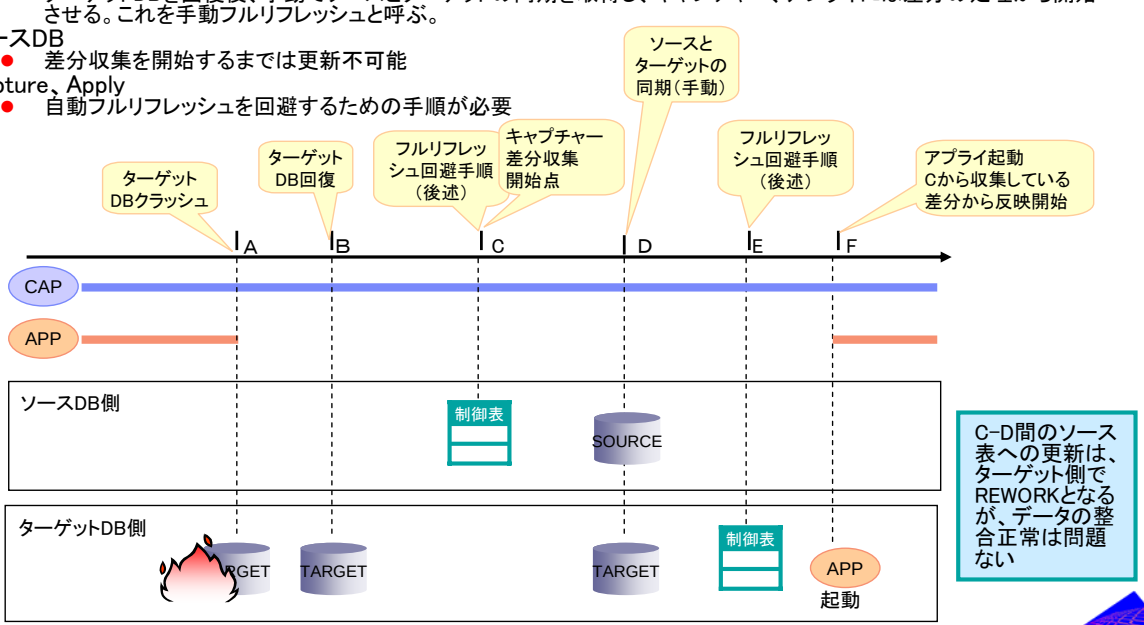
手動フルリフレッシュによる全件リフレッシュ

- 概略
  - ターゲットDBが最新まで戻らなかった場合や、ソースとターゲットの整合性が取れているかが不明な場合、ソースとターゲットの同期が必要
  - ターゲットDBを回復後、手動でソースとターゲットの同期を取得し、キャプチャー、アプライには差分の処理から開始させる。これを手動フルリフレッシュと呼ぶ。
- ソースDB
  - 差分収集を開始するまでは更新不可能
- Capture
  - 自動フルリフレッシュを回避するための手順が必要
  - 自動フルリフレッシュ回避手順については、次節で説明する



手動フルリフレッシュによる全件リフレッシュ

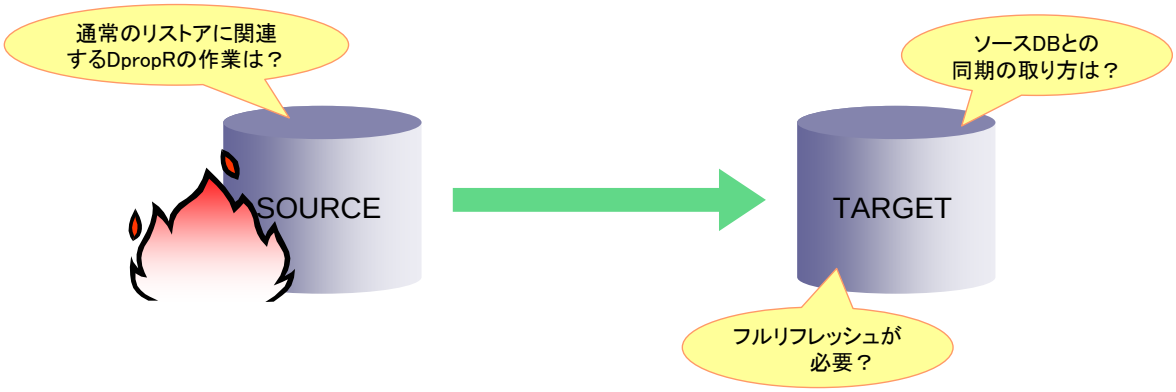
- 概略
  - ターゲットDBが最新まで戻らなかった場合や、ソースとターゲットの整合性が取れているかが不明な場合、ソースとターゲットの同期が必要
  - ターゲットDBを回復後、手動でソースとターゲットの同期を取得し、キャプチャー、アプライには差分の処理から開始させる。これを手動フルリフレッシュと呼ぶ。
- ソースDB
  - 差分収集を開始するまでは更新不可能
- Capture、Apply
  - 自動フルリフレッシュを回避するための手順が必要



ソース・データベース　クラッシュ時

■リカバリ方法

- ソースデータベースのリカバリ
- ターゲットデータベースのリカバリは、67ページの考慮点と、前述のターゲットデータベース・クラッシュ時のリカバリを参考



blank page

ソースデータベースのリカバリ

- 通常のリカバリ方法を実行
  - バックアップリストア＆ロールフォワードリカバリなど
- リストア後のキャプチャー起動時の注意点
  - ソースDBがリストアされた場合、次にキャプチャーをWARMスタートで起動すると、キャプチャーはリストアされたことを検知し、以下のASN0144Eを出力し、停止する
  - ただし、再度起動しなおすと、正常にWARM起動できる

ASN0144E CAPTURE capture_schema . プログラムは、ソース・データベース src_db_name がリストア、またはロールフォワードされたことを検出しました。整合性をリストアするために、コールド・スタートをお勧めします。

説明:

キャプチャー・プログラムは、WARMNS または WARMSI の開始モードで開始されました。キャプチャー・プログラムはウォーム・スタートを試行したとき、DB2 ログ読み取り API から、ソース・データベースがリストアまたはロールフォワードされ、ログ・シーケンス番号が再利用されたことを示す戻りコードを受信しました。ソース・データベースの状態とキャプチャーされたデータの状態の整合性が、もはや取れていません。キャプチャー・プログラムは終了され、コールド・スタートへの切り替えは自動的に行いません。

ユーザーの処置:

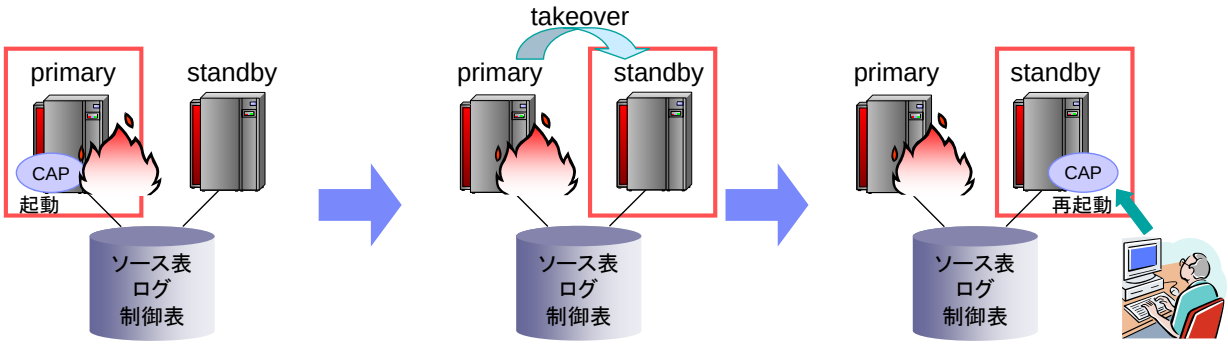
キャプチャー・プログラムのウォーム・スタートを実行しても安全だという確信がある場合は、キャプチャー・プログラムを再始動してください。2 度目の試行では終了されません。キャプチャー・プログラムのウォーム・スタート後、キャプチャーされたデータが整合するかどうかについて確信が持てない場合は、キャプチャー・プログラムのコールド・スタートを実行することをお勧めします。

ターゲットデータベースのリカバリ

- ソースデータベースがクラッシュしたが、最新まで戻った場合（整合性がとれた場合）
  - キャプチャー、アプライから見て整合性が取れた状態であるはずなので、キャプチャーやアプライをそのまま起動可能
- ソースデータベースとターゲットデータベースの整合性の保障が不明な場合
  - ソースとターゲットの整合性を取るためには、手動/自動フルリフレッシュが必要
- ソースデータベースが最新まで戻らなかった場合（ターゲットデータベースの方が進んでしまった場合）
  - そのまま、キャプチャーやアプライは起動できてしまうが、データの整合性が失われた状態
  - ターゲットデータベースの方が進んでいることから、ターゲットからの回復も考えら得るが、アプライされていなかった更新データなどは復旧できない

HACMP使用時の考慮点(1/2)

■ キャプチャー/アプライがHACMP環境で起動している場合
- 例)ソースサーバーがtakeoverする場合



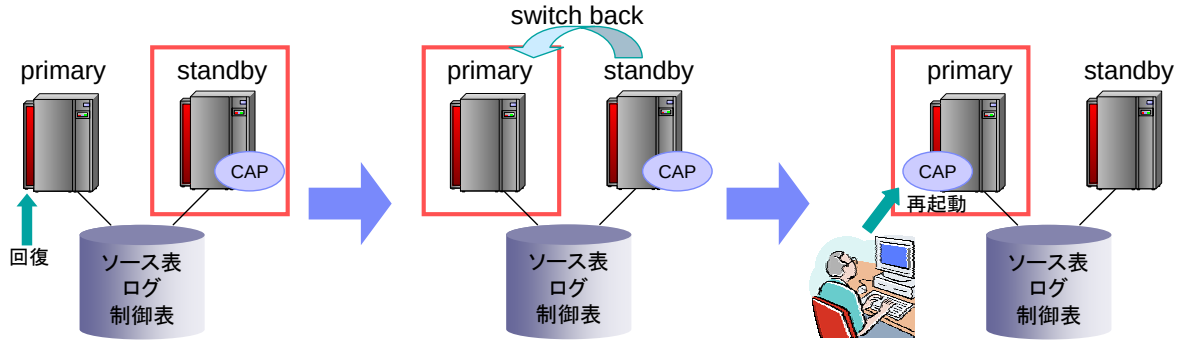
■ primaryのサーバーがダウン
キャプチャープログラムは、サーバー上で起動しているため、ダウンする
ソース表や制御表、CD表が共有ディスク上にあれば、キャプチャーからみる整合ポイントは保持される

■ HACMPによりtakeover
primaryサーバからstandbyサーバーへtakeoverされる

■ キャプチャー再起動
takeover完了後は、ユーザーによるキャプチャーの再起動が必要
(スクリプト実行やサービス登録などでよい)

HACMP使用時の考慮点(2/2)

■ キャプチャー/アプライがHACMP環境で起動している場合
- 例)ソースサーバーがswitchbackする場合(Switchback時)



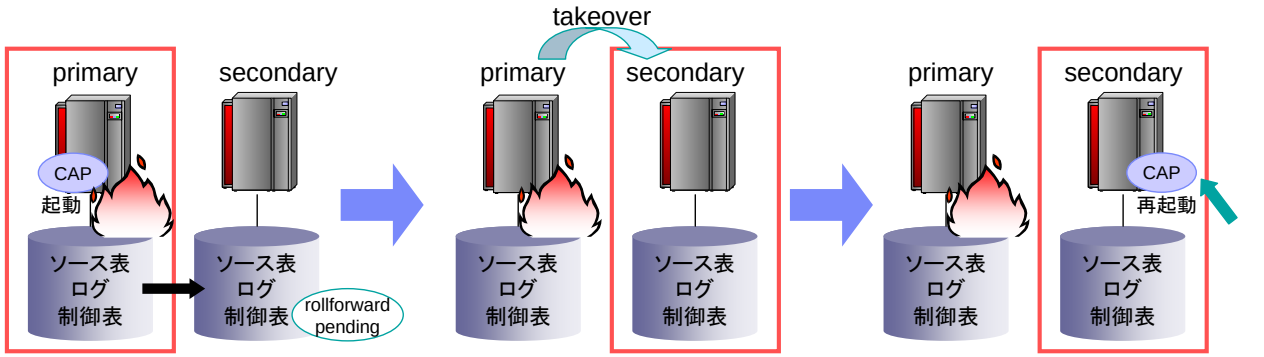
■ primaryのサーバーを回復

■ switchbackを行う

■ キャプチャー再起動
switchback完了後は、ユーザーによるキャプチャーの再起動が必要
(スクリプト実行やサービス登録などでよい)

HADR使用時の考慮点(1/2)

■ キャプチャー/アプライがHADR環境で起動している場合
- 例)ソースサーバーがtakeoverする場合



■ primaryのサーバーがダウン
キャプチャープログラムは、サーバー上で起動しているため、ダウンする
ソース表や制御表、CD表が共有ディスク上にあれば、キャプチャーからみる整合ポイントは保持される

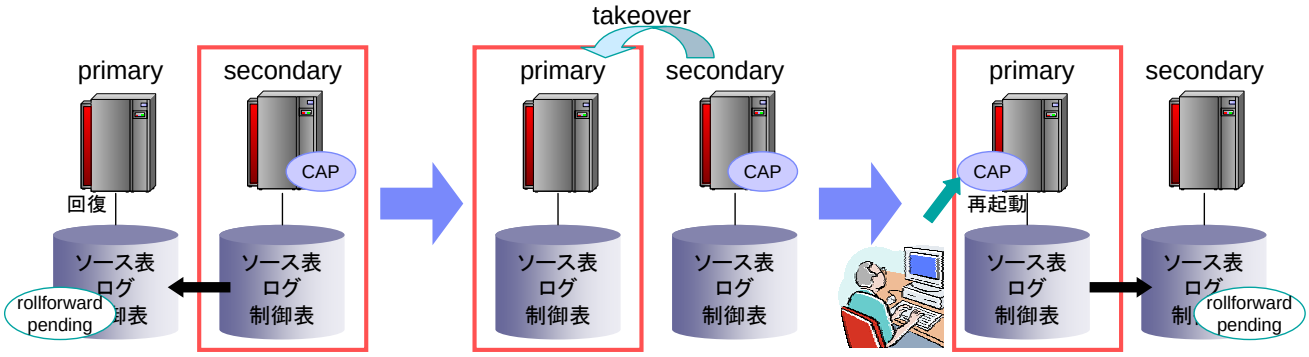
■ secondaryサーバーへtakeover

■ キャプチャー再起動
takeover完了後は、ユーザーによるキャプチャーの再起動が必要
(スクリプト実行やサービス登録などでよい)

このとき、キャプチャーがロールフォワードを検知し、ASN0144Eとなるため、2度起動する必要がある

HADR使用時の考慮点(2/2)

■ キャプチャー/アプライがHADR環境で起動している場合
- 例)ソースサーバーがtakeoverする場合



■ primaryのサーバーを回復
この後、primaryがsecondaryの更新をキャッチアップする

■ 再びprimaryサーバーへtakeover

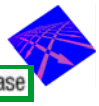
■ キャプチャー再起動
takeover完了後は、ユーザーによるキャプチャーの再起動が必要
(スクリプト実行やサービス登録などでよい)

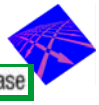
このとき、キャプチャーがロールフォワードを検知し、ASN0144Eとなるため、2度起動する必要がある

レプリケーション環境の変更

- サブスクリプション単位の非活動化 / 活動化
- メンバー単位の非活動化 / 活動化
- レプリケーション対象表の追加
- レプリケーション対象列の追加

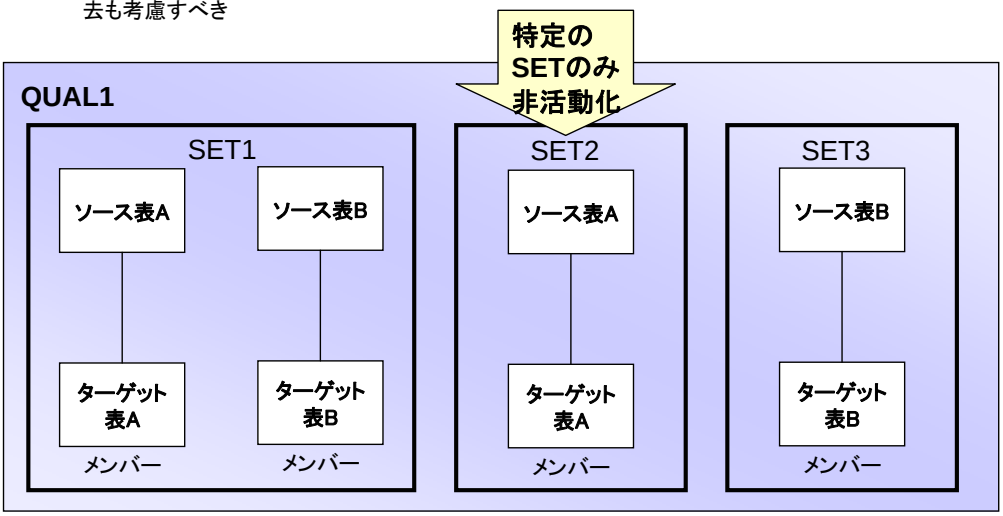
参考
マニュアルより「SQLレプリケーション環境の変更」
http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.ii.doc/admin/te0ch000.htm





サブスクリプション単位の非活動化 / 活動化

- 一部のサブスクリプション・セットだけを非活動化/活動化
  - 一部のサブスクリプション・セットのみターゲット表への反映を停止したい時、セットのサブスクリプション単位の非活動化を行いアプライさせないようにすることが可能
  - 非活動化されると、アプライはそのセットに対する現行の処理を終了後、サブスクリプションセットを非活動化する
  - 長期間非活動化する注意が必要
    - ソース表からの読み取りを継続する場合は、CD表、UOW表にデータが蓄積される
    - CD表に蓄積されたデータがRETENTION_LIMITを超えて保管されるとフルリフレッシュとなる
    - 長期間停止したい場合は、Captureによる読み取りの中止や、メンバー、ソース表のレジストレーションの除去も考慮すべき



解説:

- サブスクリプション・セットは、除去することなく、非活動化できます。サブスクリプション・セットを非活動化すると、アプライ・プログラムは、現在の処理サイクルを完了させてから、サブスクリプション・セットの処理を停止します。サブスクリプション・セットを長期間にわたり非活動化させる場合は、非活動化したサブスクリプション・セットに関して注意が必要になります。

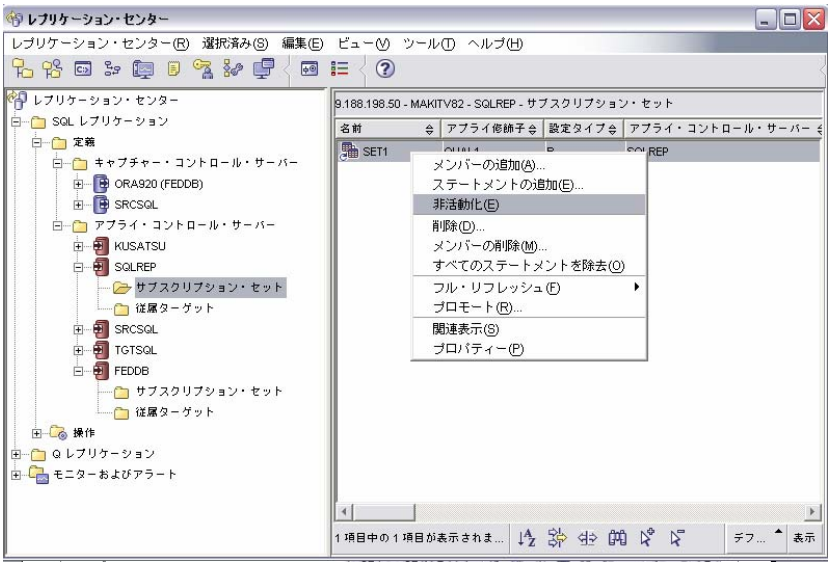
- 長期間の非活動化
  - 現在は必要ないが、将来使用する可能性のあるサブスクリプション・セットは非活動化しておくことができます。しかし、このサブスクリプション・セットを長期間にわたって非活動化しておく必要がある場合には、その間にCD表に累積した変更データによってキャプチャー・プログラムおよびアプライ・プログラムのパフォーマンスが影響を受ける可能性があるため、追加の処理が必要になります。
  - キャプチャー・プログラムは、プルーニング時にはアプライ・プログラムによってターゲットに反映されたデータのみをプルーニングします。長い時間にわたりアプライ・プログラムが非アクティブになるか、サブスクリプション・セットが非アクティブ化されると、CD表、UOW表に未アプライデータが蓄積されてゆきます。CD表、UOW表に大量データが蓄積されることは、ディスクスペースを逼迫するだけでなく、プルーニングのパフォーマンスにも影響します。
  - UOW 表および CD 表は、最終的にはキャプチャー・プログラムのRETENTION_LIMIT (デフォルト値は 7 日) に基づいてプルーニングされます。しかし、RETENTION_LIMITによってアプライされていないプルーニングがされてしまうと、次回サブスクリプション活動化時に、ソース表とターゲット表の不整合を解消しようとフルリフレッシュが実行されます。

  - このようなプルーニングの問題を防止するためには、長期間にわたり非活動化しておく必要のあるサブスクリプション・セットについては、プルーニング情報をリセットすることができます。

```
UPDATE ASN.IBMSNAP_PRUNE_SET
        SET SYNCHPOINT=x'00000000000000000000', SYNCHTIME=NULL
        WHERE APPLY_QUAL= 'QUAL1' AND SET_NAME ='SET1'

UPDATE ASB.IBMSNAP_PRUNCNTL
        SET SYNCHPOINT= NULL,SYNCHTIME= NULL
        WHERE APPLY_QUAL= 'QUAL1' AND SET_NAME= 'SET1'
```

サブスクリプション単位の非活動化

- レプリケーション・センターから

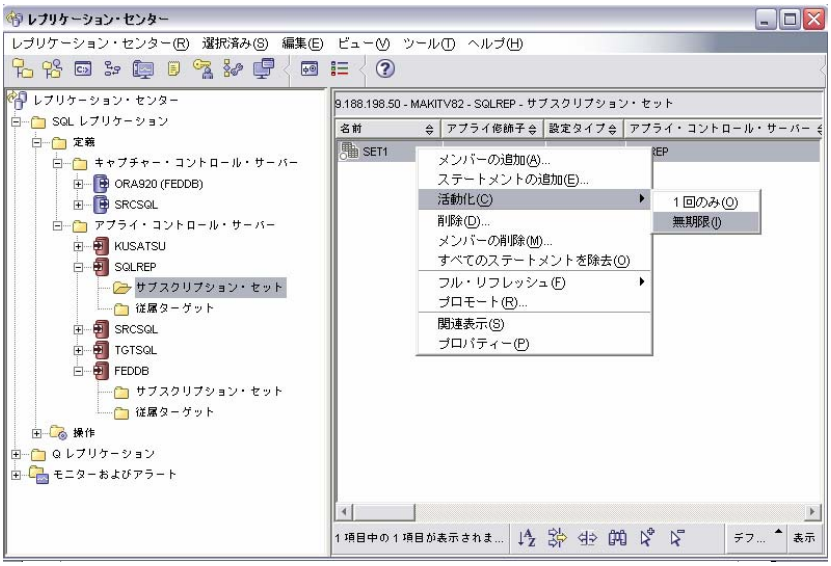


- SQL（IBMSNAP_SUBS_SET表のACTIVATE=0 で非活動化）

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


サブスクリプション単位の活動化

- レプリケーション・センターから



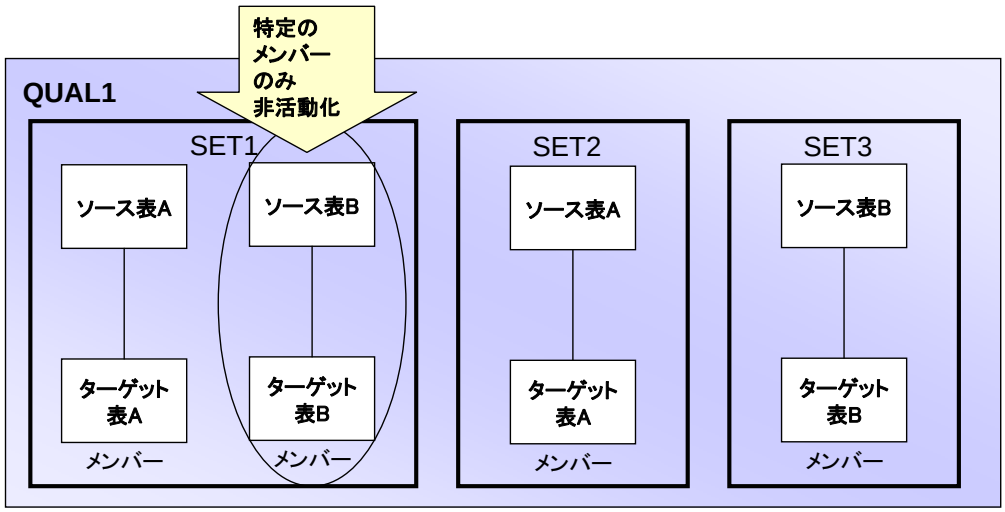
- SQL
（IBMSNAP_SUBS_SET表のACTIVATE=1or 2 で活動化）

```
UPDATE ASN.IBMSNAP_SUBS_SET
        SET ACTIVATE = 1
        WHERE SET_NAME = 'SET1' ;
```

- 1回のみ（ACTIVATE=2）
アプライは1サイクルのみサブスクリプションセットを処理して、非活動化する
- 無制限 （ACTIVATE=1）
活動化し、処理を続ける

メンバー単位の非活動化 / 活動化

- 一部のメンバーだけを非活動化/活動化
  - 一部のメンバー(ターゲット表)への反映を停止したい時、メンバー単位で非活動化を行い,セット内の処理からスキップさせることが可能
  - DB2 UDB V8 FP2からの機能
  - GUIからは操作不可能
  - 一度非活動化(MEMBER_STATE=Dに)したメンバーには、再活動化した際にはフルリフレッシュが実行される



解説:

非活動化

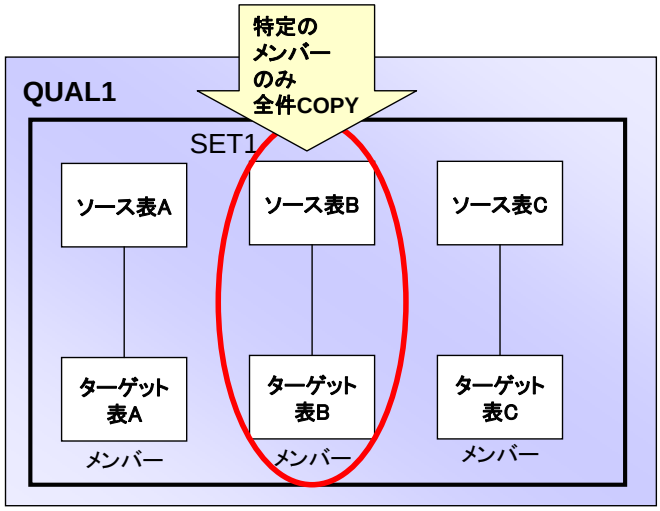
```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
      SET MEMBER_STATE = 'D'
      WHERE APPLY_QUAL= apply_qualifier
                              SET_NAME = set_name
                              WHOS_ON_FIRST = whos_on_first
            SOURCE_OWNER = source_owner
            SOURCE_TABLE = source_table
            SOURCE_VIEW_QUAL = source_view_qualifier
            TARGET_OWNER = target_owner
            TARGET_TABLE = target_table  ;
```

活動化

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
      SET MEMBER_STATE = 'N'
      WHERE APPLY_QUAL= apply_qualifier
                              SET_NAME = set_name
                              WHOS_ON_FIRST = whos_on_first
            SOURCE_OWNER = source_owner
            SOURCE_TABLE = source_table
            SOURCE_VIEW_QUAL = source_view_qualifier
            TARGET_OWNER = target_owner
            TARGET_TABLE = target_table  ;
```

セット内の特定のメンバーのみフルリフレッシュする

- 障害などで、セット内の特定のメンバーのみ整合性が失われた場合、そのメンバーだけフルリフレッシュすることが可能



解説:

```
select target_table,member_state from asn.ibmsnap_subs_membr

TARGET_TABLE MEMBER_STATE
------------ ------------
TGT_A        S
TGT_B        S
TGT_C        S
```

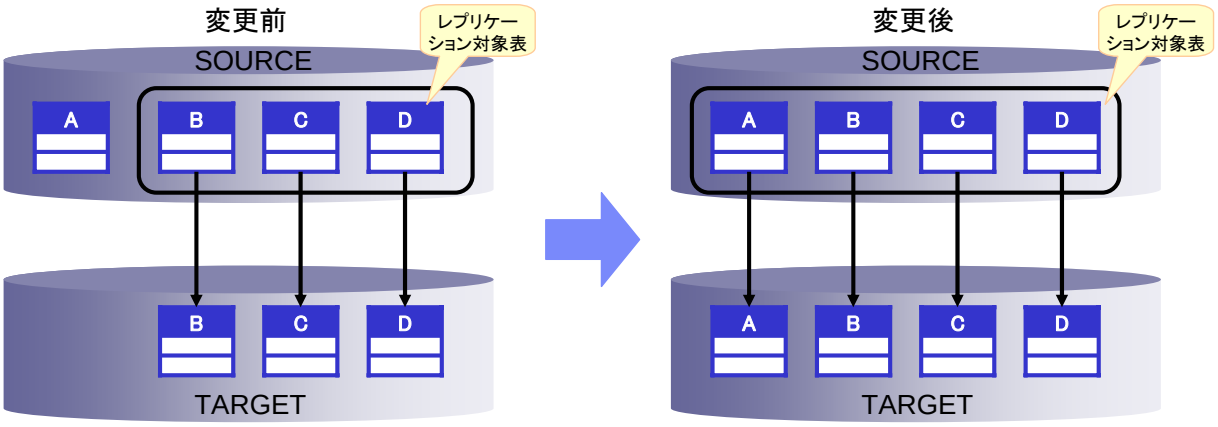
ここで、TGT_Bのみ、MEMBER_STATEを‘N’に手動更新(‘N’は新規に登録したメンバーという意味)
update asn.ibmsnap_subs_membr set membr_state = ‘N’ where target_table = ‘TGT_B’;

```
select target_table,member_state from asn.ibmsnap_subs_membr

TARGET_TABLE MEMBER_STATE
------------ ------------
TGT_A        S
TGT_B        N
TGT_C        S
```

APPLYを起動すると、TGT_Bのみフルリフレッシュのみを実行する。このサイクルで処理すべきであったTGT_A、TGT_Cの差分は、次回のアプライサイクルでターゲット表に適用する。

レプリケーション対象表の追加

■既存のレプリケーション環境にソース表を追加する方法
- 既存環境でのレプリケーション対象表に別の表を追加する手順
- Capture、Applyを停止せずに追加可能



解説:

■ 新規にソース表を追加する手順
※設定方法は、レプリケーション・センター編、ASNCLP編を参照

①ソース表の登録

②
- 既存のアプライ修飾子に追加する場合 → ④へ
- 新規にアプライ修飾子を作成して追加する場合 → ③へ

③サブスクリプション・セットを新規作成し、新しいアプライ修飾子名をつける → ⑥へ

④
- 既存のサブスクリプションに追加する場合 → ⑤へ
- 新規にサブスクリプションを作成して追加する場合 → ⑥へ

⑤サブスクリプション・セットを新規作成

⑥任意のサブスクリプションへのメンバー追加

⑦キャプチャープログラムに対して、制御表の再読み込みをさせる
asnccmd CAPTURE_SERVER reinit

⑧既存のアプライ修飾子に追加した場合は、処理が行われる。
新規にアプライ修飾子を作成した場合は、アプライを起動
asnapply QUALNAME CNTLSERVER

解説：

■ ソース表の登録　スクリプト例

```
CREATE TABLE MAKITV82.CDTEST1
(
 IBMSNAP_COMMITSEQ CHAR ( 10 ) FOR BIT DATA   NOT NULL ,
 IBMSNAP_INTENTSEQ CHAR ( 10 ) FOR BIT DATA   NOT NULL ,
 IBMSNAP_OPERATION CHAR ( 1 )  NOT NULL ,
 C1 INTEGER  NOT NULL ,
 C2 INTEGER  NOT NULL ,
 C3 CHARACTER ( 5 )  NOT NULL
) IN USERSPACE1;


CREATE  UNIQUE  INDEX MAKITV82.IXCDTEST1
 ON MAKITV82.CDTEST1
 (
  IBMSNAP_COMMITSEQ ASC,
  IBMSNAP_INTENTSEQ ASC
 )
 PCTFREE 0
 MINPCTUSED 0;


ALTER TABLE  MAKITV82.CDTEST1  VOLATILE CARDINALITY;

INSERT INTO ASN.IBMSNAP_REGISTER (SOURCE_OWNER, SOURCE_TABLE,
SOURCE_VIEW_QUAL, GLOBAL_RECORD, SOURCE_STRUCTURE,
        SOURCE_CONDENSED,
SOURCE_COMPLETE, CD_OWNER, CD_TABLE, PHYS_CHANGE_OWNER,
PHYS_CHANGE_TABLE, CD_OLD_SYNCHPOINT, CD_NEW_SYNCHPOINT,
DISABLE_REFRESH, CCD_OWNER, CCD_TABLE, CCD_OLD_SYNCHPOINT,
SYNCHPOINT, SYNCHTIME, CCD_CONDENSED, CCD_COMPLETE,
        ARCH_LEVEL,
DESCRIPTION, BEFORE_IMG_PREFIX, CONFLICT_LEVEL,
CHG_UPD_TO_DEL_INS, CHGONLY, RECAPTURE, OPTION_FLAGS,
STOP_ON_ERROR, STATE, STATE_INFO ) VALUES(
```

```
'MAKITV82',
'TEST1',
0,
'N',
1,
'Y',
'Y',
'MAKITV82',
'CDTEST1',
'MAKITV82',
'CDTEST1',
null,
null,
0,
null,
null,
null,
null,
null,
null,
null,
'0801',
null,
null,
'0',
'N',
'N',
'Y',
'NNNN',
'Y',
'I',
null );
```


解説：

■ サブスクリプション・セットの作成　スクリプト例

```
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,
 'SQLREP',
 'SQLREP',
 'SQLREP',
 'SQLREP',
 0,
 'R',
 0,
 null,
 null,
 0,
 '0801',
 '2005-10-29-12.10.18.104',
 null,
 'ASN',
 'ASN',
 'NNNN',
 null,
 null,
 null,
 null,
 null
 );
```

解説：

■ メンバーの作成　スクリプト例

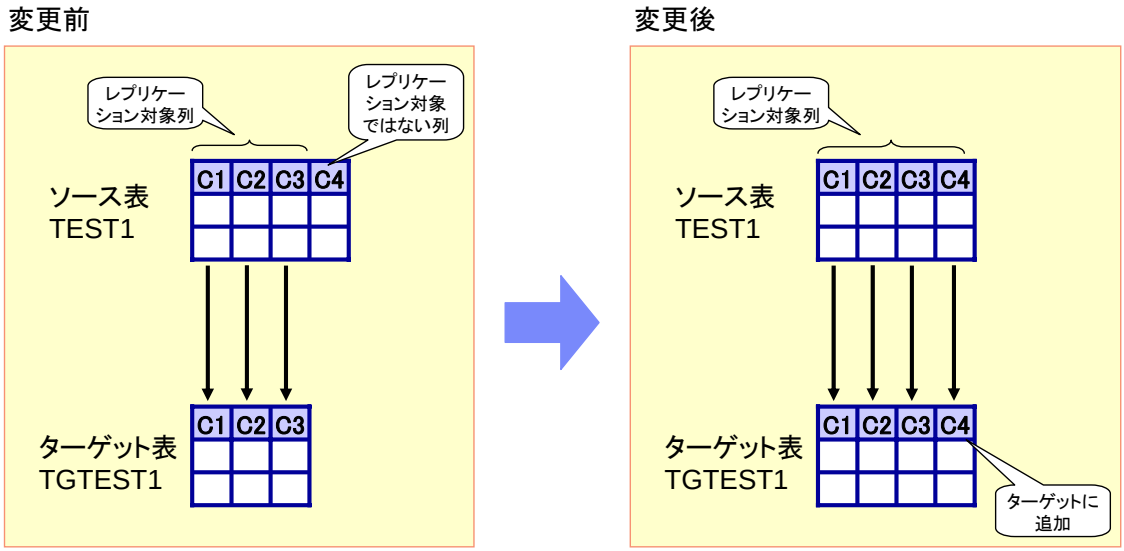
```
INSERT INTO ASN.IBMSNAP_SUBS_MEMBR (
 APPLY_QUAL, SET_NAME, WHOS_ON_FIRST,
 SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL,
 TARGET_OWNER, TARGET_TABLE, TARGET_STRUCTURE,
 TARGET_CONDENSED, TARGET_COMPLETE,
 PREDICATES, UOW_CD_PREDICATES, JOIN_UOW_CD,
 MEMBER_STATE,TARGET_KEY_CHG,LOADX_TYPE,
 LOADX_SRC_N_OWNER,LOADX_SRC_N_TABLE
 ) VALUES (
 'QUAL1',
 'SET1',
 'S',
 'MAKITV82',
 'TEST1',
 0,
 'MAKITV82',
 'TGTEST1',
 8,
 'Y',
 'Y',
 null,
 null,
 null,
 'N',
 'N',
 null,
 null,
 null
 );
```

```
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',
 'MAKITV82',
 'TGTEST1',
 'C1',
 'A',
 'Y',
 1,
 'C1'   );

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',
 'MAKITV82',
 'TGTEST1',
 'C2',
 'A',
 'N',
 2,
 'C2'   );

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',
 'MAKITV82',
 'TGTEST1',
 'C3',
 'A',
 'N',
 3,
 'C3'   );
```


blank page

レプリケーション対象列の追加

■ 現在レプリケーション対象でない列を、レプリケーション対象にする方法
- 作業中は列を追加するソース表に関するレプリケーションが行われないことが前提



解説:

■ まず、この作業中は列を追加するソース表に関するレプリケーションが行われないことが前提です。
そのためには、以下を実施する必要があります。

1. Captureを停止する

or

2. 列を追加するソース表への更新をとめる
（他のレプリケーション対象表の変更も収集しなくなると困るなどの理由から、Captureが停止できない状況など）

- 1の場合、Captureを停止しているため列を追加するソース表や他のソース表への更新があっても構いません。
- 2の場合、列を追加するソース表への更新が無いことが前提となります。

3. その表をレプリケーションするサブスクリプション・セットの非活動化

■ 手順

① Capture停止、サブスクリプション非活動化
→ その時点で、変更対象となるソース表への変更データがすべてターゲット表へ適用されたことを確認

② CD表に新規列を追加（ALTER TABLE）

③ ターゲット表に新規列を追加（ALTER TABLE）

④ IBMSNAP_SUBS_COLSに追加した新規列情報をINSERT

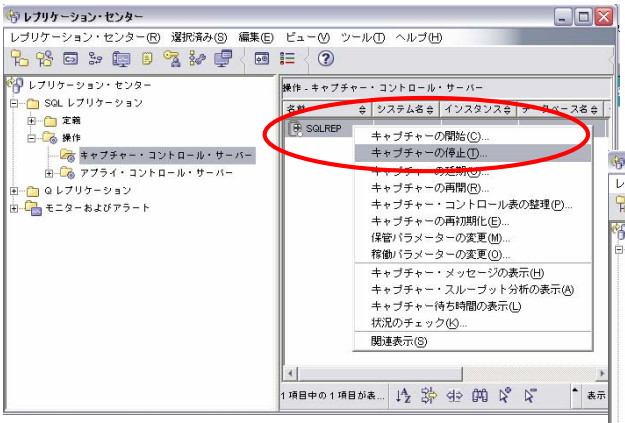
⑤ Captureの再起動、サブスクリプションセットの活動化

解説：

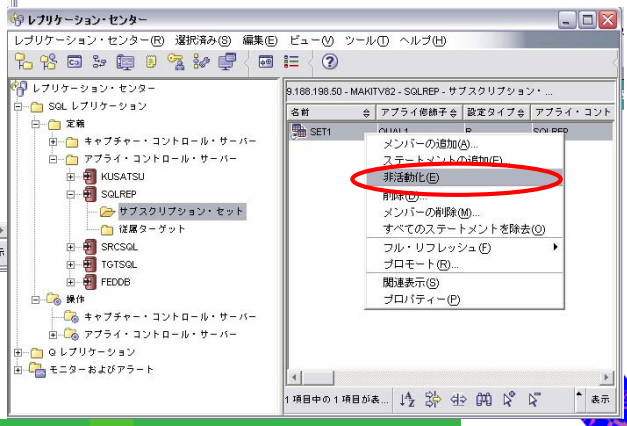
■ レプリケーション対象列の追加方法

①Capture停止とサブスクリプション非活動化

■ Captureの停止
GUIから停止 or asnccmdコマンドで停止



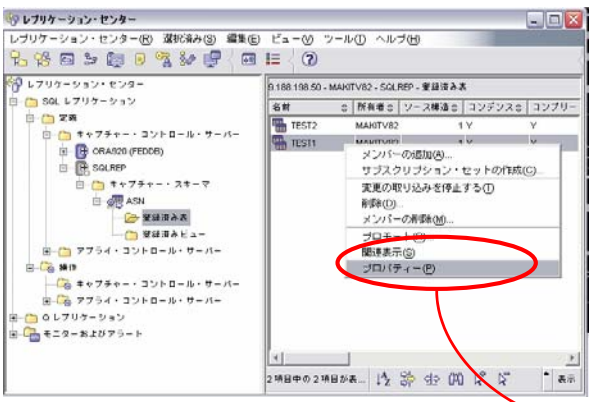
■ サブスクリプション非活動化
GUIから非活動化 or IBMSNAP_SUBS_SET表の該当セットのACTIVATEを 0 に更新



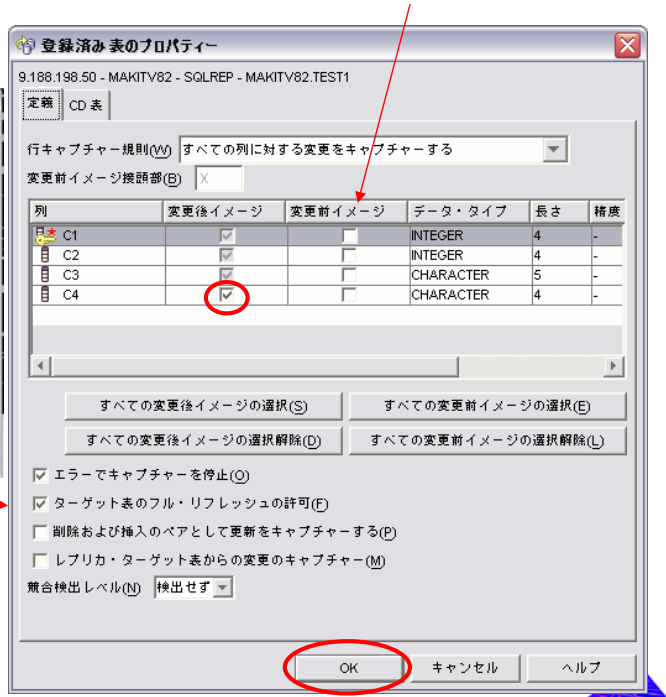
解説：

■ レプリケーション対象列の追加方法

② CD表に新規列を追加（ALTER TABLE）



必要な場合変更前イメージもチェック



CD表に列を追加するスクリプトが生成される
手動でSQLを書く場合、ソース表の列タイプと合わせる

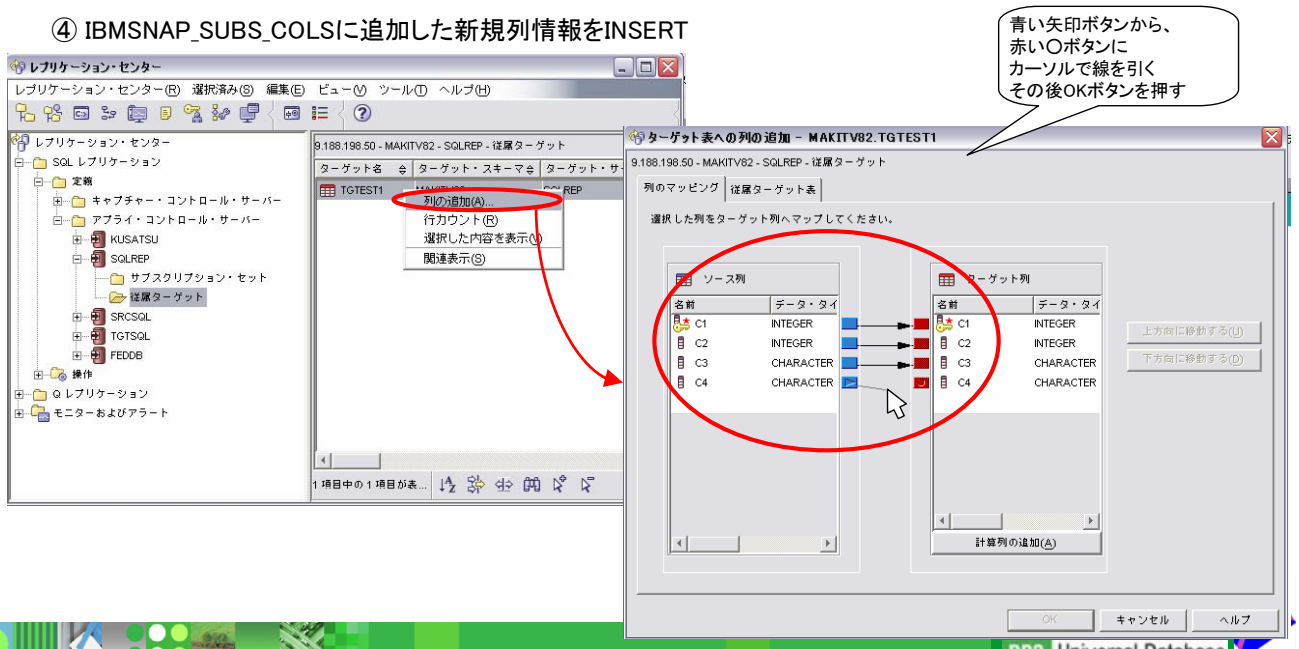
```
ALTER TABLE MAKITV82.CDTEST1
ADD C4 CHARACTER ( 4 ) ;
```

解説：

■ レプリケーション・センターを使用したレプリケーション対象列の追加

③ターゲット表に新規列を追加
- ターゲット表に対して、ALTER TABLE ADD COLUMN を発行

④ IBMSNAP_SUBS_COLSに追加した新規列情報をINSERT



解説：

④で生成されるスクリプト

```
INSERT INTO ASN.IBMSNAP_SUBS_COLS
        (APPLY_QUAL,
        SET_NAME ,
        WHOS_ON_FIRST,
        TARGET_OWNER,
        TARGET_TABLE,
        COL_TYPE,
        TARGET_NAME,
        IS_KEY,
        COLNO,
        EXPRESSION)
SELECT
        'QUAL1',
        'SET1',
        'S',
        'MAKITV82',
        'TGTEST1',
        'C',
        'C4',
        'N',
        MAX(COLNO + 1),
        'C4'
FROM ASN.IBMSNAP_SUBS_COLS
 WHERE APPLY_QUAL='QUAL1'
        AND SET_NAME='SET1'
        AND WHOS_ON_FIRST='S'
        AND TARGET_OWNER='MAKITV82'
        AND TARGET_TABLE='TGTEST1';
```

■ レプリケーション・センターを使わずにSQLを作成する場合、まず、IBMSNAP_SUBS_COLS表で、該当する表のカラム番号の最大値を調べ、それに +1 したものをCOLNOとし、列情報を追加します。

```
SELECT MAX(COLNO) FROM
ASN.IBMSNAP_SUBS_COLS
        WHERE APPLY_QUAL = 'QUAL1'
        AND SET_NAME = 'SET1'
        AND TARGET_TABLE = 'TGTEST1';
```

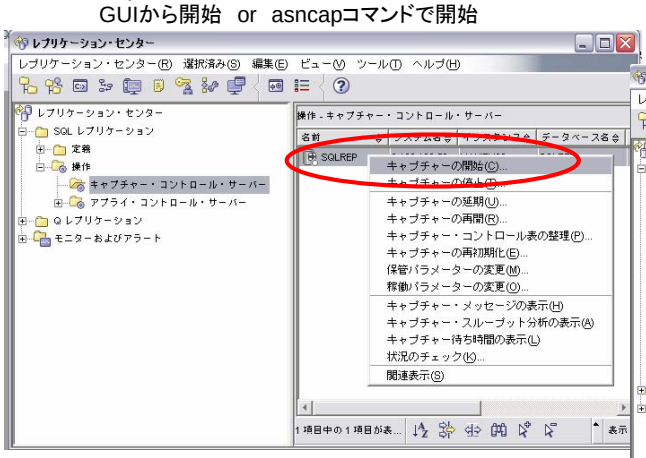
■ SUBS_COLSにINSERTする際に指定するデータなどはマニュアルを参照し、適切なデータを指定してください。
http://publib.boulder.ibm.com/infocenter/db2help/topic/com.ibm.db2.ii.doc/admin/re0tac06.htm

解説：

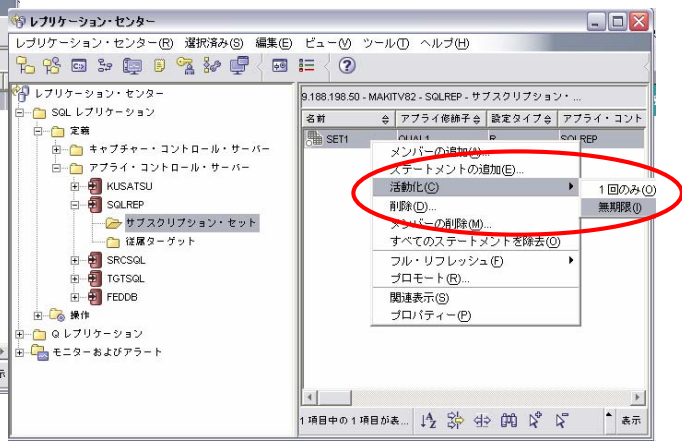
■ レプリケーション・センターを使用したレプリケーション対象列の追加

⑤ Captureの再起動、サブスクリプションセットの活動化

■ Captureの開始
GUIから開始　or　asncapコマンドで開始



■ サブスクリプション活動化
GUIから活動化　or　IBMSNAP_SUBS_SET表の該当セットのACTIVATEを 1 に更新



Captureを停止しない場合の注意）
■Captureを停止しない場合、列を追加するソース表への変更が無いことが前提です。
■手順が終了し、列が追加されたソース表へ更新がなされると、Captureは一度エラーを出力します。
（Captureがキャッシュしているcd表の構造と実際収集したデータ構造に違いがあるためです。）
■その後、Captureは再度CD表の定義を読み込み、追加された列に関する変更データをCD表へ書き込みます。

blank page

キャプチャー済みアーカイブ・ログの判別

- ログの消去を行う場合、キャプチャーが読み終えたログのみを消さなければならない
  - まだキャプチャーが読み込み終了していないログが存在しない場合、キャプチャーはCOLDSTARTにスイッチする
- キャプチャー済みログの確認方法
  ①RESTART表からMIN_INFLIGHTSEQ 値 をSELECTする

```
$ db2 "SELECT MIN_INFLIGHTSEQ FROM ASN.IBMSNAP_RESTART WITH UR"

MIN_INFLIGHTSEQ
----------------------------------
x'0000000000000B987583'

  1 record(s) selected.
```

MIN_INFLIGHTSEQとは、まだ未コミットの一番若いログシーケンス番号のこと

②取得したMIN_INFLIGHTSEQ値の下12桁を使用して、db2flsnコマンドでキャプチャー済みログ番号を調べる。

```
$ db2 get db cfg for sqlrep | grep Path
 Path to log files                                = /dbland1/makitv82/NODE0000/SQL00004/SQLOGDIR/
$
$ cd /dbland1/makitv82/NODE0000/SQL00004
$
$ db2fls n -q 000000000B987583
S0000315.LOG
```

これより古いログファイル、S0000314.LOGまではキャプチャーは読み込み済み。

解説:

- DB2 リカバリー・ログには、DB2 リカバリー機能の提供と、実行中のキャプチャー・プログラムへの情報の提供という 2 つの目的があります。DB2 リカバリー、および DB2 レプリケーションの両方についてログ・データを保存する必要があります。このデータを削除する前に、キャプチャー・プログラムおよび DB2 が、ログの処理を完全に終了していることを確実に確認する必要があります。
- 参考
  レプリケーション環境の保守
  http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.ii.doc/admin/te0ma000.htm