

SQLレプリケーション・ソリューション 概要と構築ガイド

日本IBMシステムズ・エンジニアリング株式会社
インフォメーション・マネージメント

2006/04

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。

©日本IBM システムズ・エンジニアリング(株) インフォメーション・マネージメント



内容

- レプリケーション・ソリューション
 - レプリケーション・ソリューションの種類
 - レプリケーション製品とサポート
- レプリケーション・コンポーネント
 - コンポーネントとデータベース構成概念
 - レプリケーションの流れ
 - レプリケーションの開始
- レプリケーション・ソリューションのプランニング
 - レプリケーション・ソリューションの適用
 - レプリケーションの構成と組み合わせ
- レプリケーション環境のプランニング
 - SQLレプリケーションに関連するオブジェクト、概念
 - DB2データベース・システム
 - データベース・オブジェクト
 - スピル・ファイル
 - サブスクリプション関連
- レプリケーションの定義
 - レプリケーション・センター
 - ASNCLP



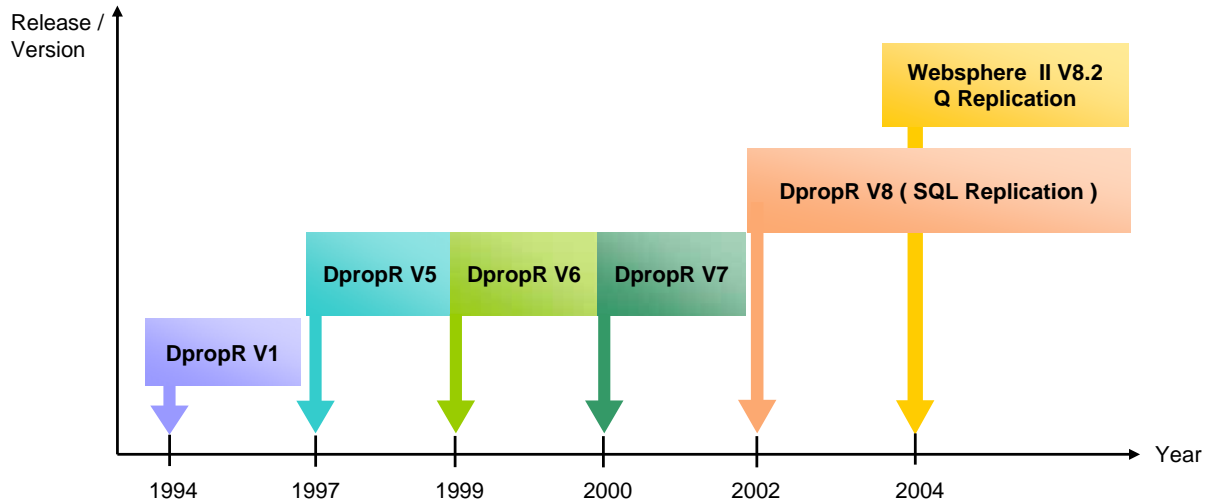
レプリケーション・ソリューション

内容

- レプリケーション・ソリューション
 - レプリケーション・ソリューションの種類
 - レプリケーション製品とサポート

IBM DB2 レプリケーション・ソリューションの発展

- 第一世代 レプリケーション
 - 第二世代 レプリケーション
 - 第三世代 レプリケーション
 - 新世代 レプリケーション
- Data Propagator Relational(DpropR) V1
 - DpropR V5、 DpropR V6、 DpropR V7
 - DpropR V8 (SQLレプリケーションと呼称)
 - Websphere II V8.2 Qレプリケーション



レプリケーション・ソリューションの手法

■ レプリケーション

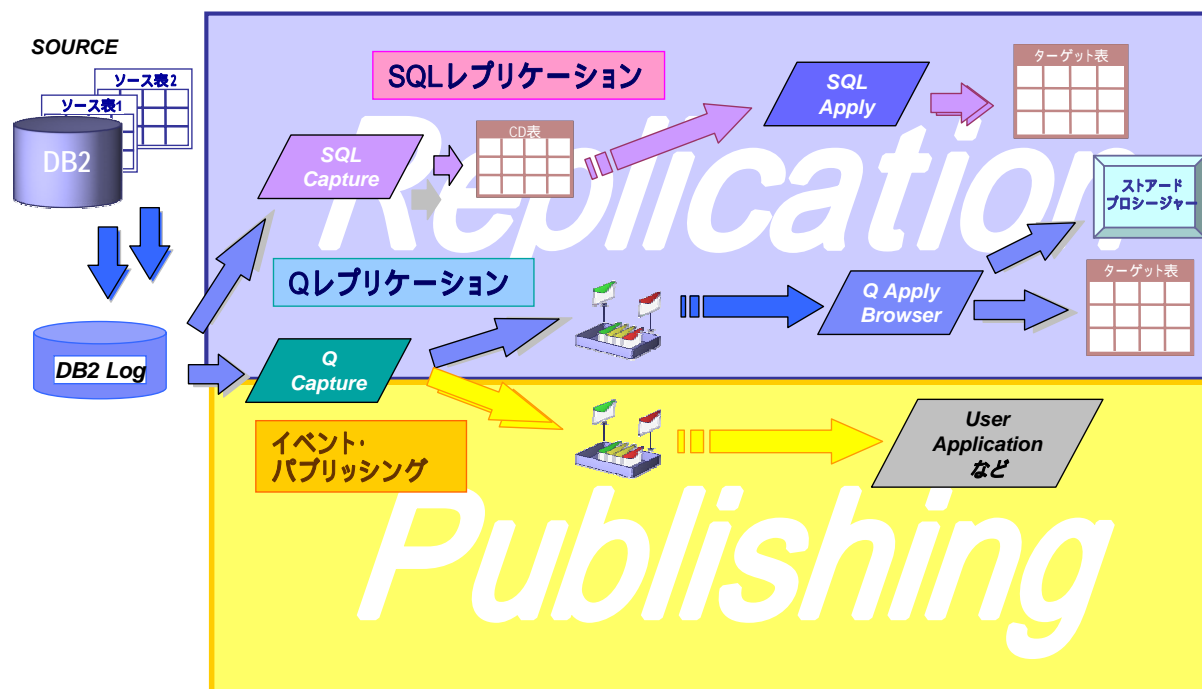
- ソースのコミットされたトランザクション・データを、ターゲットに反映するしくみ
- ソースの更新情報を収集するコンポーネントと更新情報をターゲットに反映する、2つのコンポーネントの連携で構成される
- トランザクション・データの伝達は、DB2の表を介して行う方法と、WebSphere MQメッセージ・キューを介して行う方法が提供されている

■ パブリッシング

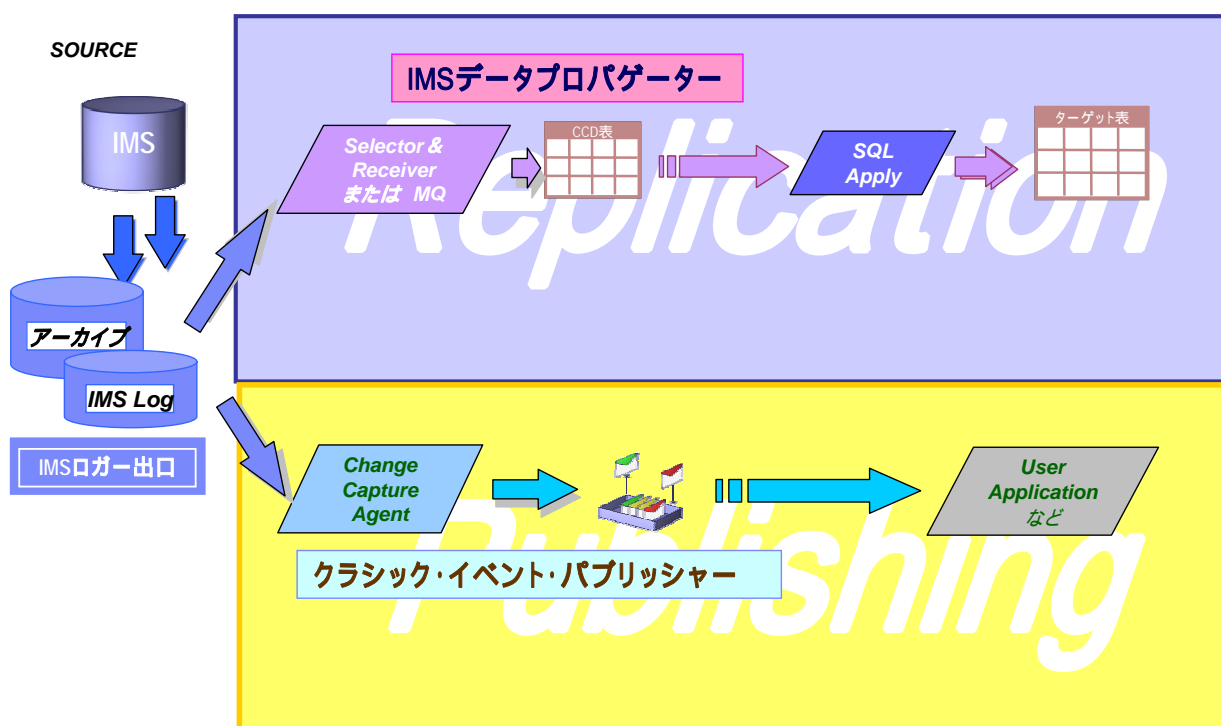
- ソースのコミットされたトランザクション・データを、XML(Extensible Markup Language)形式のメッセージに変換するしくみ
- メッセージは、WebSphere MQメッセージ・キューに置く方法が提供されている



DB2をソースとしたレプリケーションとパブリッシング



IMSをソースとしたレプリケーションとパブリッシング

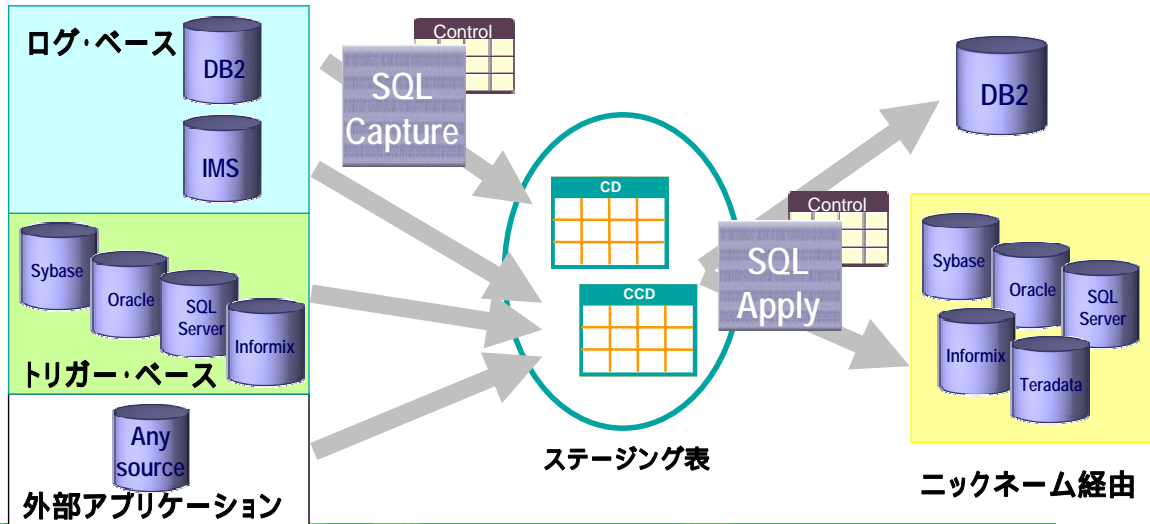


SQL レプリケーション アーキテクチャー

■ DB2の機能によるレプリケーション

■ 特徴

- 接続ログベース、トリガーベースによる変更情報の収集
- 色々な組み合わせでレプリケーションが可能
- データの加工が容易
- オンデマンドなレプリケーション

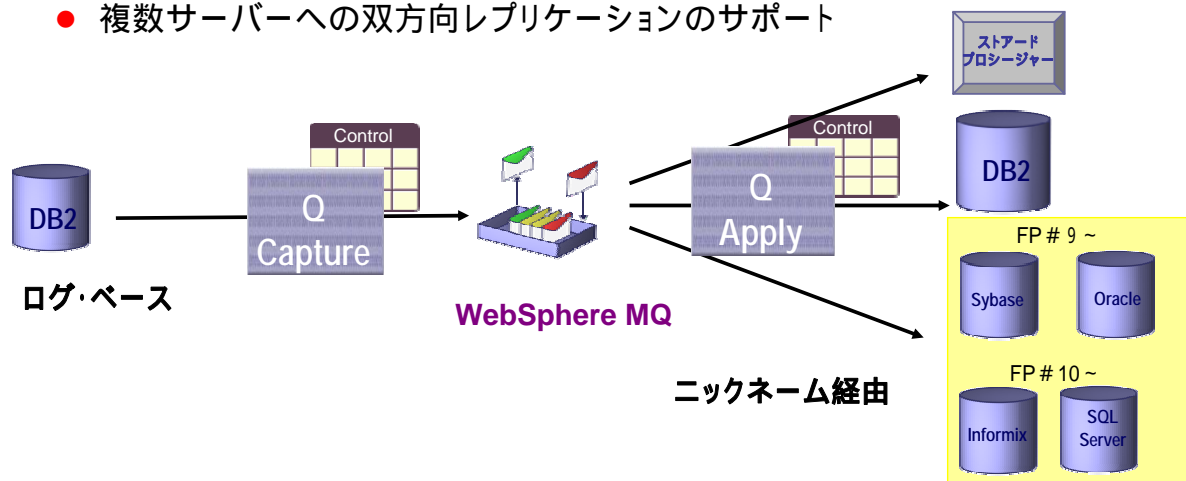


Q レプリケーション アーキテクチャー

■ WebSphere MQを使用

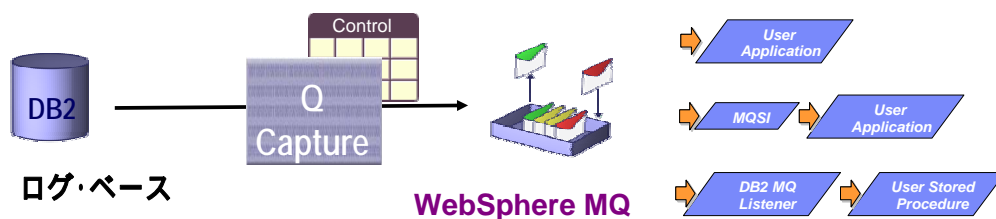
■ 特徴

- スループットの向上
- データの鮮度の向上
- XMLパブリッシング
- 複数サーバーへの双方向レプリケーションのサポート



イベント・パブリッシング

- Qキャプチャーの機能の一部
- データ・イベントをXMLメッセージとしてWebSphere MQ メッセージ・キューに発行
- 特徴
 - Qアプライは不要
 - XML形式のため、情報をどのように利用するのも自在
 - データ鮮度はほぼリアルタイム



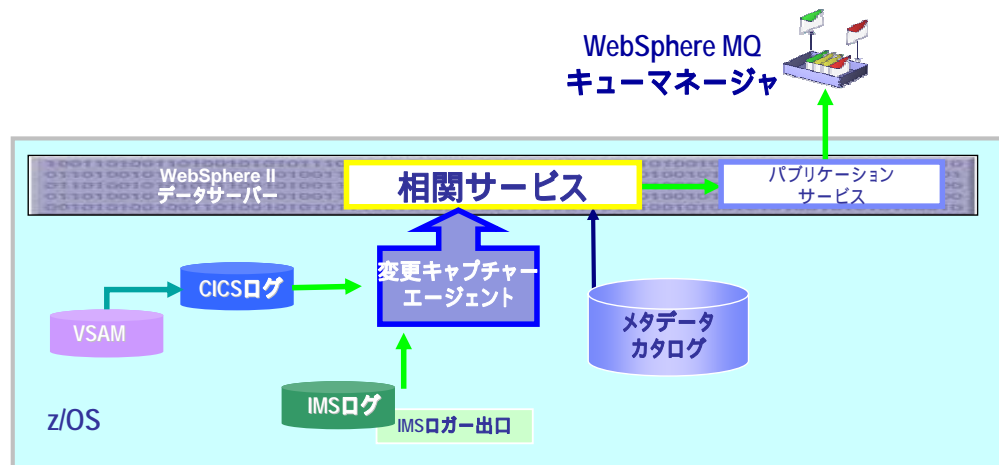
解説:

- DB2 IIV 8.2 を使用すると、DB2UDBでのデータベースの変更を XML メッセージとしてキャプチャーして、それらを WebSphere MQ メッセージ・キューに発行することによって、データ・イベントをビジネス・プロセスに簡単に関連付けることができます。
- 変更は、個々の行メッセージとして、またはトランザクション内のすべての変更を表すメッセージとして発行できます。
- アプリケーションまたはサービスを WebSphere MQ と直接統合するか、Java Message Service (JMS) をサポートすることによって、データの変更が生じるときにそれを非同期で受信できるようにします。これにより、データ稼働のワークフロー処理を作成できます。
- パブリッシュされるデータがユーザーアプリケーションに直接渡される例に加え、データがまず、WBI Event Brokerに渡され、処理されてから、ユーザーアプリケーションに渡される例も示しています。
- また、データを一度MQリスナーで処理した後、DB2のストアード・プロシージャに渡すことも可能です。



IMS,VSAMのイベント・パブリッシャー

- IMS、VSAMデータに関する、新しいレプリケーション方式
- 特徴
 - 階層型のデータを表型のデータ形式に変換してパブリッシュ
 - パブリッシュするデータの形式は、XMLフォーマット
 - データのステー징エリアとしてのz/OSのDB2は不要
 - データ鮮度はほぼリアルタイム



解説:

- DB2 UDB for z/OS または IMS データベースに対するデータ変更や、CICS を介した VSAM ファイルに対する変更をキャプチャーすることができます。これらの変更は、XML メッセージ形式を使用して WebSphere MQ メッセージ・キューに発行されます。
- 変更キャプチャー・エージェントが、変更データを取得
 - IMSログガーEXIT又はログ・ファイルにアクセス
 - 変更データは、Correlationサービスに転送される
- 相関サービス:
 - “コミット”または“ロールバック”まで変更データを保持
 - ロールバック:
 - 変更データは廃棄
 - コミット:
 - 変更データをUTF-8にコード変換
 - リレーショナル型のXMLに再フォーマット
 - Publicationサービスに配布
- パブリケーション・サービス:
 - WebSphere MQに発行
 - 相関サービスに配布済みの確認を返す
- IMSとVSAMのイベント・パブリッシャーの実装のためには、メタデータの定義が必要です。
- これらの論理的な表の定義はデータベースやCICSのログから収集すべき情報を識別するためのものです。



レプリケーション製品

- IBM DB2 Universal Database
- DB2 DataPropagator for z/OS
- IBM WebSphere Information Integration V8.2製品群
 - 分散プラットフォーム
 - z/OSプラットフォーム } 詳細は次ページ
- DB2 DataPropagator for iSeries
- DB2 Data Propagator Q Capture for VSE and VM



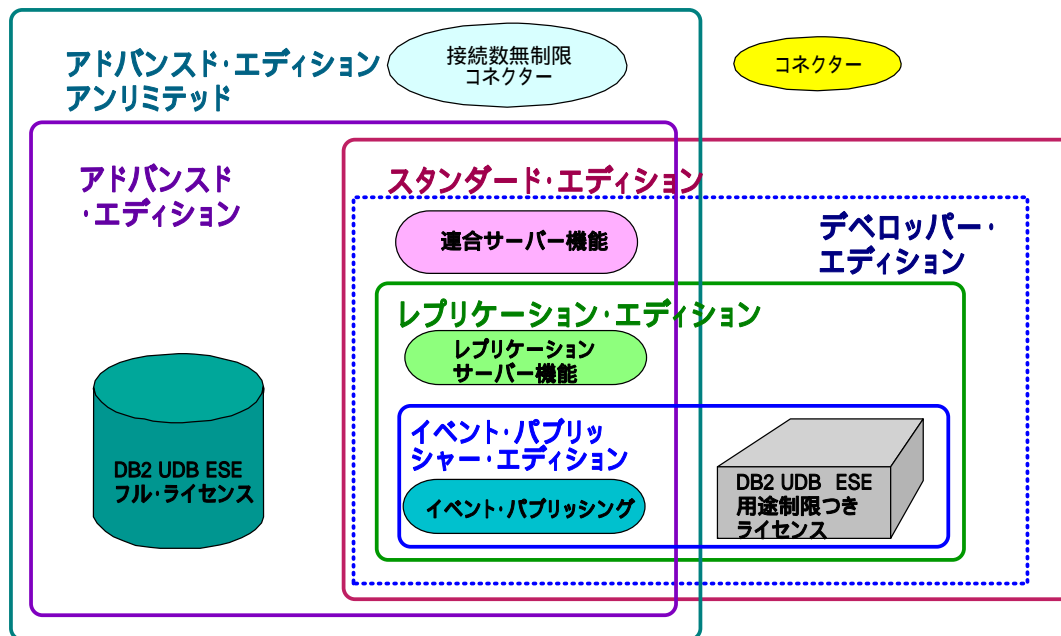
レプリケーション製品

- IBM WebSphere Information Integration V8.2製品群
 - 分散プラットフォーム
 - WebSphere Information Integration Event Publisher Edition
 - WebSphere Information Integration Replication Edition
 - WebSphere Information Integration Standard Edition
 - WebSphere Information Integration Advanced Edition
 - WebSphere Information Integration Advanced Edition Unlimited
 - WebSphere Information Integration Developer Edition
 - z/OSプラットフォーム
 - WebSphere Information Integrator Replication for z/OS
 - WebSphere Information Integration Event Publisher for DB2 UDB z/OS V8.2
 - WebSphere Information Integration Classic Event Publisher for IMS V8.2
 - WebSphere Information Integration Classic Event Publisher for VSAM V8.2
 - WebSphere Information Integration Classic Event Publisher for CA-IDMS
 - WebSphere Information Integration Classic Federation for z/OS V8.2



WebSphere Information Integrator V8.2 Edition

■分散プラットフォーム



解説:

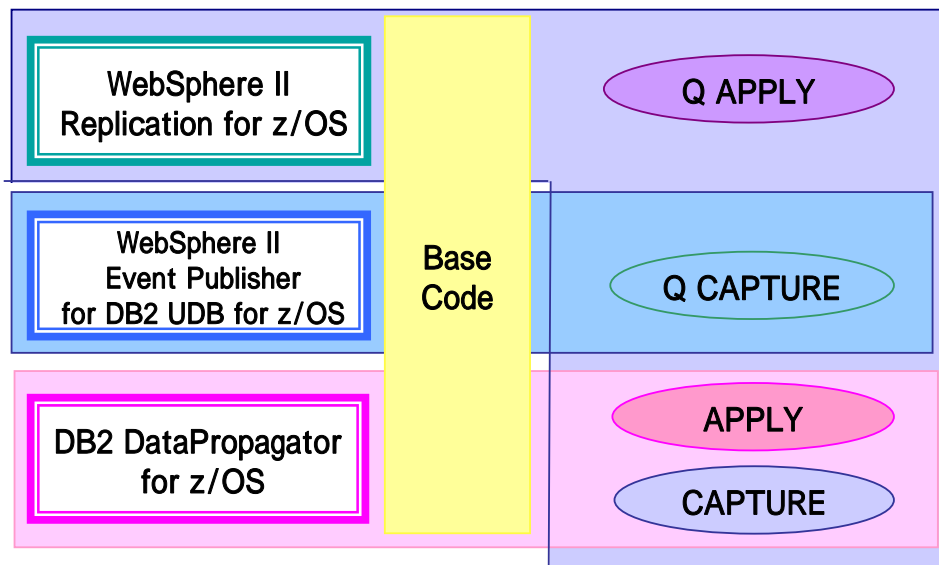
分散プラットフォーム

- 各製品の詳細に関しましては以下の製品発表レターをご参照ください。
 - DBA04079 2004-09-15 DB2 Information Integrator V8.2 の発表
 - DBA05022 2005-03-04 IBM Information Integrator の WebSphereブランドへの変更の発表
- WebSphere Information Integrator Event Publisher Edition
 - DB2 UDB for Linux, UNIX、または Windows データベースのイベントもしくは変更を取り込み、それを XML メッセージにフォーマットして、WebSphere MQ にパブリッシュします。
- WebSphere Information Integrator Replication Edition
 - WebSphere Information Integrator Event Publisher Editionの機能とレプリケーション・サーバーの機能を提供します。お客様は、混合リレーショナル・データ・ソース間でデータをレプリケーションしたり、DB2 UDB のソースおよびターゲットの高スループット・キューのレプリケーションに集中することができます。SQL ベースとキュー・ベースの両方のレプリケーション・アーキテクチャーが組み込まれています。
- WebSphere Information Integrator Standard Edition
 - WebSphere Information Integrator Replication Edition の機能の他に、強力なコスト・ベースの照会最適化および統合キャッシングを含む、フェデレーテッド・データ・サーバーの機能も提供します。
- WebSphere Information Integrator Advanced Edition
 - WebSphere Information Integrator Standard Edition の機能と、DB2 Universal Database Enterprise Server Edition V8.2 の無制限ライセンスを提供します。これにより、データベース・サーバーとしてローカル DB2 UDB を使用することによって得られるパワーと融通性が加わります。
- WebSphere Information Integrator Advanced Edition Unlimited
 - WebSphere Information Integrator Advanced Edition のすべての機能と、接続数無制限のコネクターの権利が含まれています。
- WebSphere Information Integrator Developer Edition
 - 1 人のアプリケーション開発者が、異なるデータおよびコンテンツ・ソースを検索、フェデレーション、レプリケーション、およびイベント・パブリッシングを統合する複合アプリケーションを設計、構築、またはプロトタイプ化するための低コストのパッケージを提供します。



WebSphere Information Integrator Replication for z/OS

■コンポーネントの関連



解説:

z/OSプラットフォーム

- 各製品の詳細に関しましては以下の製品発表レターをご参照ください。
 - DBA04098 2004-09-15 New and Enhanced Data Management Tools for DB2 の発表
 - DBA05022 2005-03-02 IBM Information Integrator の WebSphereブランドへの変更の発表
 - DBA05023 2005-03-02 DB2 Information Integrator for z/OS製品名称変更の発表
- WebSphere Information Integrator Event Publisher for DB2Universal Database for z/OS
 - DB2 UDB for z/OS データベース・イベントを取り込み、それを XML メッセージにフォーマットして、WebSphere MQ にパブリッシュします。
- WebSphere Information Integrator Classic Event Publisher for IMS
 - IMS データベース・イベントを取り込み、それを XML メッセージにフォーマットして、WebSphere MQ にパブリッシュします。
- WebSphere Information Integrator Classic Event Publisher for VSAM
 - VSAM イベントを取り込み、それを XML メッセージにフォーマットして、WebSphere MQ にパブリッシュします。
- WebSphere Information Integrator Classic Event Publisher for CA-IDMS
 - CA-IDMS イベントを取り込み、それを XML メッセージにフォーマットして、WebSphere MQ にパブリッシュします。
- WebSphere Information Integrator Replication for z/OS
 - お客様のニーズに応じて SQL ベースおよびキュー・ベースの両方のレプリケーション・アーキテクチャを強化する、DB2UDB for z/OS 用の変更取り込みおよび適用機能を提供します。
 - また、WebSphere Information Integrator EventPublisher for DB2 Universal Database for z/OS の機能も組み込まれています。
- WebSphere Information Integrator Classic Federation for z/OS
 - 単一の z/OS インスタンス上に存在する異なるレガシー・データベースにまたがったフェデレーテッド・データ・サーバーを提供します。
 - DB2 UDB、IMS、VSAM、Software-AG Adabas、Computer Associates CA-Datcom、および CA-IDMS データベースにまたがったフェデレーテッド・アクセスが提供されます。



レプリケーション・ソリューションのサポート

■ サポートされるバージョンとソリューション

データソース	バージョン (*11)	SQLレプリケーション		Qレプリケーション		
		ソース	ターゲット	ソース	ターゲット	イベント・パブリッシング
DB2 UDB for LUW	8.1, 8.2			(*1a) (8.2のみ)	(*1a) (8.2のみ)	(*1a) (8.2のみ)
DB2 UDB for z/OS	7, 8	(*2)	(*2)	(*3)	(*3)	(*4)
DB2 UDB for iSeries	5.1, 5.2, 5.3	(*5)	(*5)	x	x	x
DB2 VM & VSE	7.4	x	x	(*6)	x	
IMS	7.1	x	x	x	x	(*7)
VSAM for z/OS	1.4	x	x	x	x	(*8)
Informix	7.31, 8.32, 8.4, 9.3, 9.4			x	(*10)	x
Microsoft SQL Server	7.0, 2000 SP3以上	(*1b)	(*1b)	x	(*1b)(*10)	x
Oracle	8.0.6, 8.1.6, 8.1.7, 9.0, 9.1, 9.2, 9i, 10g	(*1b)	(*1b)	x	(*1b) (*9)	x
Sybase	11.9.2, 12.x	(*1b)	(*1b)	x	(*1b)(*9)	x
Teradata	2.3, 2.4, 2.5	x	(*1b)	x	x	x



blank page




レプリケーション・ソリューションのサポート

■ 選択可能なレプリケーション・ソリューション

		ターゲット				
		DB2 UDB for LUW	DB2 UDB for z/OS	DB2 UDB for iSeries	Non-IBM RDB	XMLメッセージ
ソース	DB2 UDB for LUW	SQL-Rep Q-Rep	SQL-Rep Q-Rep	SQL-Rep	SQL-Rep Q-Rep(*1)	Event-Pub
	DB2 UDB for z/OS	SQL-Rep Q-Rep	SQL-Rep Q-Rep	SQL-Rep	SQL-Rep Q-Rep(*1)	Event-Pub
	DB2 UDB for iSeries	SQL-Rep	SQL-Rep	SQL-Rep	SQL-Rep	n/a
	DB2 VM & VSE	Q-Rep	Q-Rep	N/A	Q-Rep(*1)	n/a
	IMS	n/a	IMS Dprop	n/a	n/a	Classic Event-Pub
	VSAM for z/OS	n/a	n/a	n/a	n/a	Classic Event-Pub
	Non - IBM RDB (*2)	SQL-Rep	SQL-Rep	SQL-Rep	(SQL-Rep)	n/a



注釈:

- (*1a) WebSphere II が別途必要
 - ライセンスのみでもよい
- (*1b) WebSphere II が別途必要
- (*2) DB2 DataPropagator for z/OSが別途必要
- (*3) WebSphere II Replication for z/OS が別途必要
- (*4) WebSphere II Event Publisher for DB2 UDB for z/OSもしくはWebSphere II Replication for z/OSが別途必要
- (*5) DB2 DataPropagator for iSeriesが別途必要
- (*6) DB2 Propagator Q Capture が必要
- (*7) WebSphere II Classic Event Publisher for IMSが別途必要
- (*8) WebSphere II Classic Event Publisher for VSAMが別途必要
- (*9) DB2 UDB FIXPAK9以上
- (*10) DB2 UDB FIXPAK10以上
- (*11)レプリケーション・サポート・データ・ソースに関する最新情報は、下記URLにて確認してください。
<http://www-1.ibm.com/support/docview.wss?rs=845&context=SS4NVZ&uid=swg27005462>
- (*12) Teradataを除く



SQLレプリケーションの観点では

チェック項目	必要なライセンス	備考
キャプチャーおよびアプライの稼動するプラットフォームが オープン系(Linux、Unix、Windows)のみ	×	不要
キャプチャーおよび/またはアプライの稼動するプラットフォームに、 z/OSが含まれる	○	WebSphere Information Integrator Replication for z/OS、または DB2 DataPropagator for z/OS のライセンスおよび導入が必要
ソースおよび/またはターゲットのデータベースに 非DB2サーバーが含まれる	○	WebSphere Information Integration Replication Edition以上のライセンスおよび導入が必要
		SQLレプリケーション機能はDB2 UDBの本体に含まれている
		キャプチャー機能のみ、アプライ機能のみの選択は不可 (設定は一方のみでも可)
		フェデレーション・サーバーになるDB2 UDBにライセンスが必要



SQLレプリケーションとQレプリケーションの特徴

	SQL レプリケーション	Q レプリケーション
アーキテクチャー	プラットフォーム共通一貫	
既存表やアプリケーションの変更	不要	
変更情報の収集	ログベース・トリガーベース	ログベース
ソース・ターゲットの組み合わせ	Non-IBM RDBを含め多種多様 DB2 for VSE/VMはV8レベルのレプリケーションのサポートなし	Non-IBM RDBに関しては一部のみ iSeriesのサポートなし
データの加工	SQLステートメントでできる範囲であれば、容易にできる	ストアドプロシージャの作成が必要であり、やや困難
ステージング・エリア	ソース・サーバー上のデータベースの表	WebSphere MQのキュー
レプリケーション実行タイミング	レプリケーション実行のインターバルを指定	メッセージを受信したタイミングで実行
データの鮮度	レプリケーション実行のインターバルに依存する(数分～1年)	数秒単位(データ量に依存)



サポートに関する参考資料

■ サポートデータソースについて

- <http://www-1.ibm.com/support/docview.wss?rs=845&context=SS4NVZ&uid=swg27005462>

■ サポートプラットフォームについて

- <http://www.ibm.com/software/data/integration/db2ii/requirements.html>



DB2 Universal Database

blank page



DB2 Universal Database

レプリケーション・コンポーネント

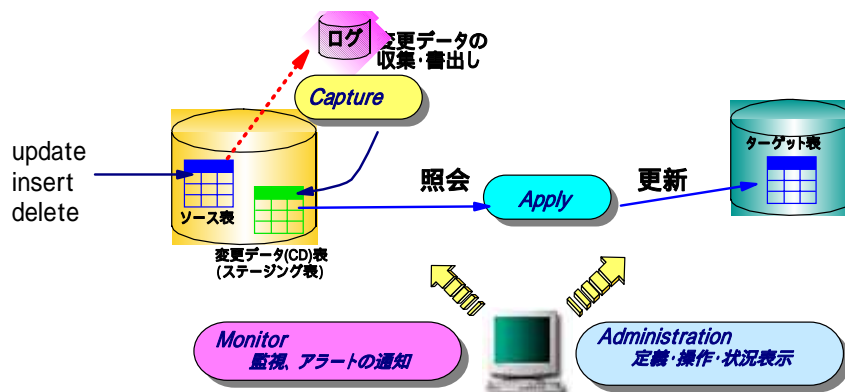
内容

- レプリケーション・コンポーネント
 - コンポーネントとデータベース構成概念
 - キャプチャー・メインタスク
 - アプライ・メインタスク
 - レプリケーションの開始

SQLレプリケーションのコンポーネント

■ 4つのコンポーネント

- キャプチャー・プログラム (Capture: 変更データ収集プログラム)
- アプライ・プログラム (Apply: 適用プログラム)
- モニター・プログラム
- アドミニストレーション
 - レプリケーション・センター (GUIツール)
 - レプリケーションの定義、操作、状況の表示、監視およびアラートの通知
 - ASNCPLコマンド (コマンド・ライン プロセッサ)
 - レプリケーション定義



解説: 各コンポーネントの機能

■ キャプチャー

- キャプチャー・プログラムは、DB2 サーバーにあるレプリケーション・ソースへの変更をログから検索し、それを CD 表(変更データ表)に記録します。
- キャプチャー・プログラムが使用するコントロール表が位置するキャプチャー・コントロール・サーバーで実行されます。キャプチャー・コントロール・サーバーは、UNIX, Windows または Z/OS で稼働しており、レプリケーション・ソースはそのサーバーに位置しています。
- キャプチャー・コントロール・サーバーが iSeries サーバーである場合、リモート側で異なるサーバーにレプリケーション・ソースをジャーナルすることができます。

■ アプライ

- 変更キャプチャー・レプリケーションでは、アプライ・プログラムはフル・リフレッシュを行ったターゲット表を移植した後、ソース表の CD 表から変更データを読み取りターゲット表へ変更を複製します。
- フル・リフレッシュ・レプリケーションでは、アプライ・プログラムがアプライ・サイクルごとに 1 回、ソース表から直接データを読み取り、それをターゲット表に直接コピーします。
- アプライ・プログラムの稼働には、サブスクリプション・セット・メンバーを含むサブスクリプション・セットの情報を元に行われます。これらのメンバーは、ソース表をターゲット表へマップします。サブスクリプション・セットをアクティブにすると、アプライ・プログラムはセットのメンバーにあるすべてのターゲット表へのデータの複製を開始することができます。
- そのセットのメンバーにあるターゲット表について、アプライ・プログラムは最初にフル・リフレッシュを実行します。(すなわち、アプライ・プログラムはターゲット表に含まれる行を削除し、コミットされていない行を含めてソース表を読み取り、ソース表からターゲット表へ直接コピーします。)ソース表が非 DB2 リレーショナル・データベースにある場合、アプライ・プログラムは連合データベースにあるニックネームから行をコピーします。
- アプライ・プログラムは概してターゲット・サーバーで実行されますが、キャプチャー・コントロール・サーバー、アプライ・コントロール・サーバー、およびターゲット・サーバーと接続可能なネットワーク上のサーバーであれば、どのサーバーでも実行することができます。
- 典型的なレプリケーション環境では、同一サーバーで実行されるいくつかのアプライ・プログラムがあり、それぞれがアプライ修飾子と呼ばれる名前で識別されます。




解説: 各コンポーネントの機能

■ アドミニストレーション および モニター

- V8で新しく追加されたレプリケーション・センターのGUIを使って、レプリケーションの定義、操作、監視、アラートの通知を行うことが可能となりました。
- V7までは、UDBコントロール・センターまたはDJRA(DataJoiner Replication Administration)のGUIを使用してレプリケーションの定義を行うことができましたが、コントロール・センターやDJRAから操作や監視はできませんでした。
- レプリケーション・センターはUDBコントロール・センターとDJRAの機能を統合したツールとなっています。即ちV7までは、ソースやターゲットがDB2以外のRDBの場合はDJRAでしか定義できませんでしたが、レプリケーション・センターは、ソースやターゲットがDB2以外であっても定義できるようになっています。(連合データベース機能経由)
- これにより、V8ではDJRAはサポートされなくなりました。またコントロール・センターからレプリケーションの定義も行うことができません。
- レプリケーション・センターの主な機能
 - 定義
 - コントロール表の作成、整理(ブルーニング)、レプリケーション定義の作成、管理
 - オブジェクトの名前やサイズのカスタマイズ
 - 操作
 - キャプチャー、アプライ、モニターのスタート
 - STOP,STATUSのようなコマンドの発行
 - 監視
 - 動的、静的なモニタリングが可能
 - レプリケーション・アラート・モニター機能を使用して、E-mailやポケットベルにアラートを送信
- ASNCPLはコマンド・ラインからレプリケーション定義を行うことができるツールです。(V8.2から提供)
- レプリケーション定義はSQLスクリプトの生成、実行により行われます。



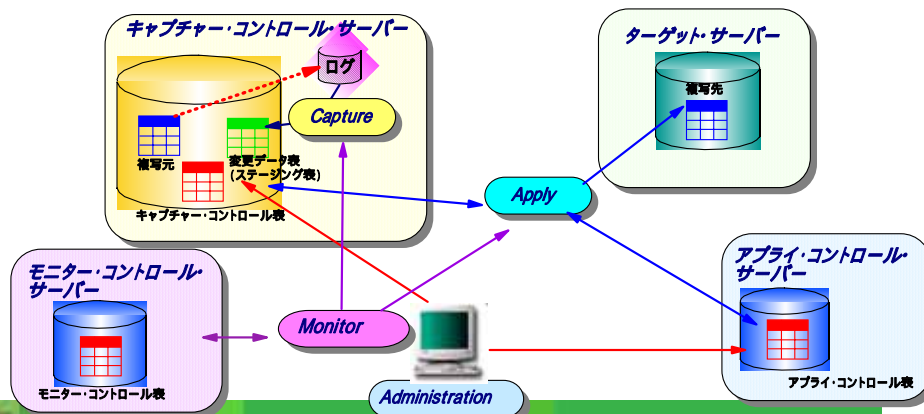

blank page



データベース構成概念

■ データベース・サーバーとコントロール表

- キャプチャー・コントロール・サーバー
 - キャプチャー・コントロール表のあるサーバー
- ターゲット・サーバー
 - ターゲット表のあるサーバー
- アプライ・コントロール・サーバー
 - アプライ・コントロール表のあるサーバー
- モニター・コントロール・サーバー
 - モニター・コントロール表のあるサーバー



解説:

■ キャプチャー・コントロール表

- レプリケーション・ソースのためのコントロール表です。
- ソース表が存在し、キャプチャーを動かすサーバー上に作成する必要があります。

■ キャプチャー・コントロール・サーバー

- キャプチャー・コントロール表が存在し、キャプチャーが稼動するデータベース・サーバーのことです。
- このサーバーが iSeries のサーバーではなく、ユーザーのソース表がリモートでジャーナルされていない限り、このサーバーには、レプリケーションのためにソースとして登録した表も含まれます。
- キャプチャー・プログラムはキャプチャー・コントロール・サーバーにとってのローカルアプリケーションとなります。
- 1つのキャプチャー・コントロール・サーバーに対して複数のキャプチャーの稼動が可能です。
 - この場合、各キャプチャー毎に専用のスキーマ名を持つキャプチャー・コントロール表のセットが必要です。
- 登録済みソース表が非 DB2 リレーショナル・データベースにある場合、キャプチャー・コントロール・サーバーは DB2 連合サーバーです。そこには、非 DB2 リレーショナル・データベースにあるキャプチャー・コントロール表についてのニックネームが含まれます。DB2 連合サーバーには、登録済み表およびその CCD 表についてのニックネームも含まれます。キャプチャー・プログラムの代わりに、非 DB2 リレーショナル・データベースでは、関連のある CCD 表にあるソース表への変更をキャプチャー・トリガーが記録します。

■ ターゲット・サーバー

- ターゲットとなる表が存在するデータベース・サーバーです。
- アプライはターゲットサーバーにとって、ローカルアプリケーションでもリモートアプリケーションであっても構いません。
- 1つのターゲットサーバーに対して複数のアプライを同時稼動させることも可能です。
- ターゲット表が Update-anywhere レプリケーションで使用されているレプリカであるか、または multi-tier のレプリケーションで使用されている外部 CCD 表である場合、ターゲット・サーバーにはキャプチャー・コントロール表も含まれます。

解説:

- アプライ・コントロール表
 - どのようにレプリケーションするかという情報 (サブスクリプション・セット情報) を格納するためのコントロール表です。
 - アプライ・コントロール表があるサーバーを、アプライ・コントロール・サーバーと呼びます。
 - アプライ・コントロール表は常にスキーマ ASN を使用するため、アプライ・コントロール表を 1 セットだけサーバーに作成できます。
- アプライ・コントロール・サーバー
 - アプライが参照するコントロー情報を保管するデータベース・サーバーのことです。
 - アプライ・コントロール・サーバーは、通常キャプチャー・コントロール・サーバーまたはターゲット・サーバーのどちらかにありますが、ネットワーク内の任意のDB2サーバーに置くことができます。
 - 一部の制御情報はキャプチャー・コントロール・サーバーにも保管されます。
 - 各アプライ・プログラムは 1つのアプライ・コントロール・サーバーと関連づけます。
 - ターゲット・サーバーあるいはキャプチャー・コントロール・サーバーと同一のデータベース・サーバーであっても構いません。
 - アプライが稼動時に必ず接続が発生します。
 - 複数のアプライ・プログラムで、アプライ・コントロール・サーバーを共有することが可能ですが、アプライ・コントロール・サーバーに存在するコントロール表のセットはASN1つです。
- モニター・コントロール表
 - V8から追加されたアラート・モニター機能用のコントロール表です。
 - モニター・コントロール表は常にスキーマ ASN を使用するため、モニター・コントロール表を 1 セットだけサーバーに作成できます。
 - UNIX, Windows および z/OS サーバーに作成することができます。
- モニター・コントロール・サーバー
 - レプリケーション・モニター機能によってレプリケーションのモニタリングを行うための、各種コントロール情報を保管するデータベース・サーバーのことです。

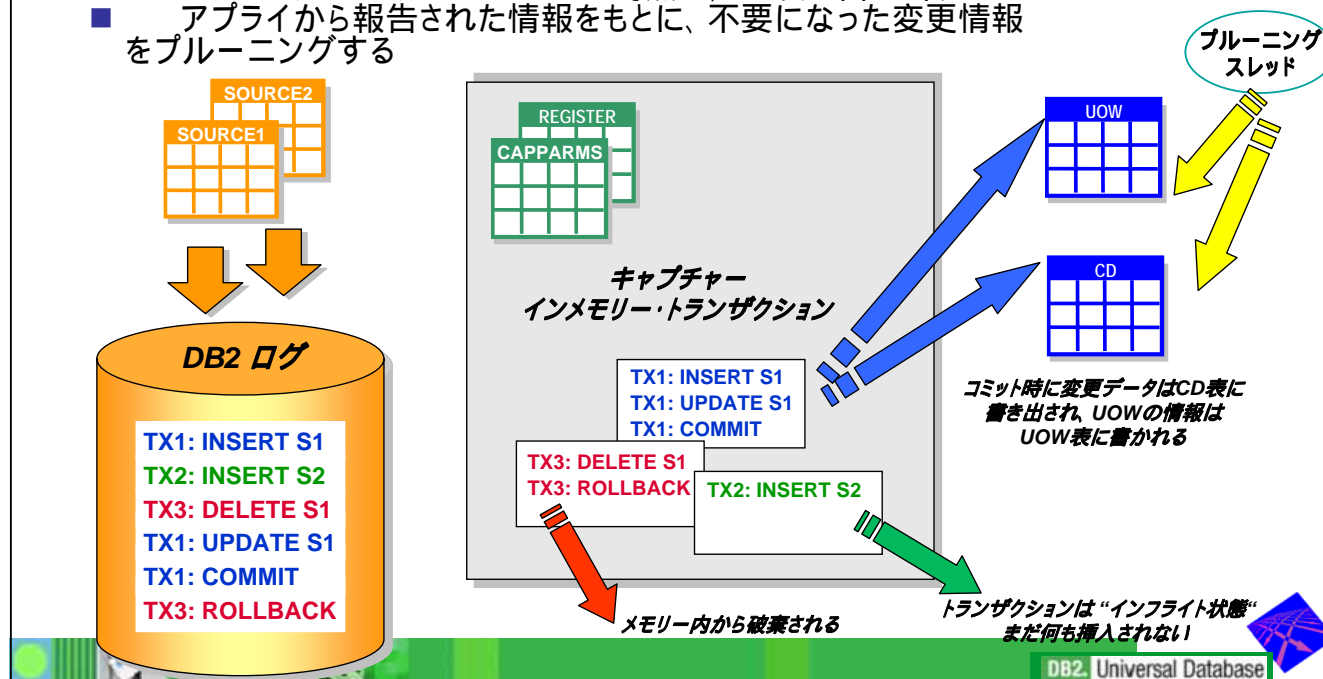



blank page



キャプチャー・メインタスク

- キャプチャーはログから変更情報を収集
- 収集した変更情報をまずメモリーに蓄える
- トランザクションがコミットされた時点で、CD表に書き出す
- アプライから報告された情報をもとに、不要になった変更情報をブルーニングする

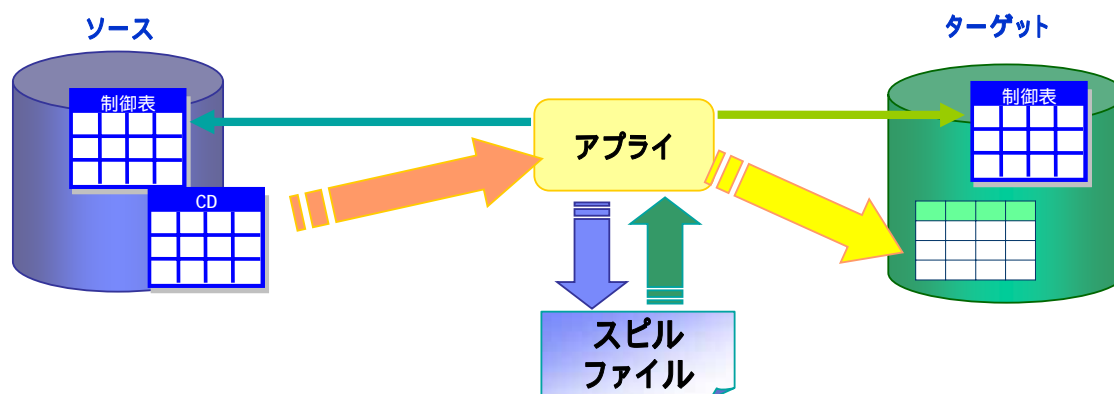


解説:

- キャプチャー・プログラムは、いずれかのソース表に対する変更を読み取ると、メモリー内に保持されている対応するデータベース・トランザクションに変更を追加します。
- メモリー内のトランザクションは、潜在的に、ログ内の対応するトランザクションのサブセットになります。これには、ソース表に対する変更のみが入ります。
- キャプチャー・プログラムは、変更が行われるトランザクションに関する ROLLBACK または COMMIT ステートメントのいずれかを読み取るまで、メモリーに変更を収集します。
- キャプチャー・プログラムは、ROLLBACK ステートメントを読み取ると、ロールバック・トランザクションに関連した変更をメモリーから消去し、COMMIT を読み取ると、コミット済みトランザクションに関連した変更を保管します。
- たとえば、表 A および表 B をレプリケーション・ソースとして登録したとします。これらそれぞれの表で、SQL レプリケーションは、登録プロセスの一部として CD 表を作成します。
- キャプチャー・プログラムを開始し、そのプログラムが、ターゲット表がソース表と同期されていることを示す信号をアプライ・プログラムから受信した後、キャプチャー・プログラムはそれらのソース表の変更に関する DB2 ログを読み取ります。
- トランザクション1 は、表 S1 に対して一連の変更を行います。変更はそれぞれ DB2 ログに記録されます。キャプチャー・プログラムはこれらの変更をメモリーに収集します。その後、トランザクション1は COMMIT ステートメントを発行します。キャプチャー・プログラムがこのステートメントを読み取ると、表 S1 に関するそれぞれの変更のコピーを CD 表に付加します。
- トランザクション2 は、表 S2 に対して一連の変更を行います。変更はそれぞれ DB2 ログに記録されます。キャプチャー・プログラムはこれらの変更をメモリーに収集します。トランザクション2は ROLLBACK ステートメントを発行します。キャプチャー・プログラムがこのステートメントを読み取ると、そのトランザクションに関連した変更をメモリーから消去します。
- トランザクション3は、表 S3 に対して一連の変更を行います。まだ変更を COMMIT も ROLLBACK もしていません。よってこのトランザクション3はメモリー内に保持されたインフライト状態と言えます。
- 一連の変更情報は、変更データ (CD) 表に蓄えられ、またトランザクション情報は UOW 表に蓄えられます。
- アプライが該当の変更をターゲットに適用し終わったあとは、これらのデータはキャプチャーのブルーニング処理によって削除されます。

アプライ・メインタスク

- アプライは、CD表から前回処理した後に溜まった、レプリケーション対象のデータを照会し、スピルファイルに書き込む
- 全てのデータをスピルファイルに書き込み終わると、その情報を元にターゲット表を更新
- 更新が終了すると、どの時点の変更まで適用を完了したかという情報をアプライ・コントロール表に記録
- その後、キャプチャー・コントロール表にも同じ情報を記録



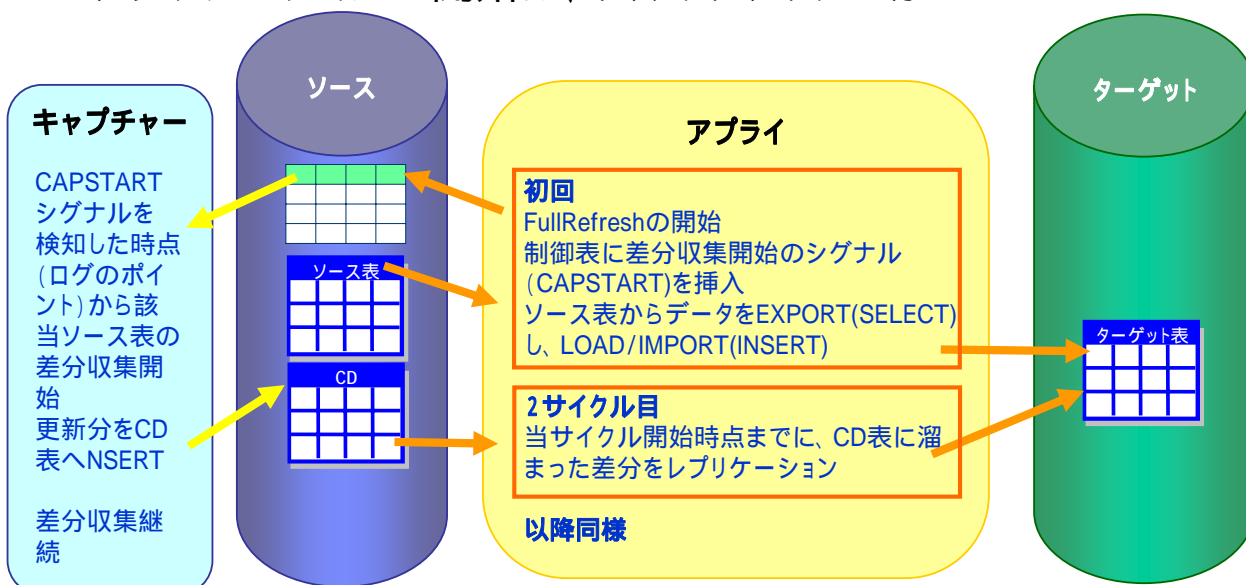
解説:

- アプライ・プログラムは割り当てられたサブスクリプション・セットの処理を開始します。
- アプライ・プログラムは、サブスクリプション・セットの作成時に指定したスケジューリングまたはイベント基準に従って、すべてのアクティブ・サブスクリプション・セットを一度に1つずつ処理します。
- たとえば、サブスクリプション・セット SetAを60分ごとに複製することを選択した場合、アプライ・プログラムは可能な限り60分に近いインターバルでそのサブスクリプション・セットを複製します。
- 60分のインターバルが経過したときにアプライ・プログラムに完了すべき他の作業が残っている場合、SetAはアプライ・プログラムがその他の作業を完了した後すぐに複製されます。
- アプライ・プログラムの処理は、2つのフェーズに分けられます。
- 初めのフェーズは、キャプチャー・コントロール・サーバーに接続し、CD表からレプリケーション対象のデータを検索して、その結果をスピルファイルに書き出します。
- その後、ターゲット・サーバーに接続して、スピルファイルから読み込んだ更新内容に従って、ターゲット表を更新します。
- 1つのサブスクリプション・セットに複数のメンバーが含まれている場合は、全てのメンバーに対応するスピルファイルを書き出してから、各ターゲット表への更新を行います。
- 全ての更新が終了すると、アプライ・コントロール表、キャプチャー・コントロール表に適用済みの更新情報を書き込み、スピルファイルを削除します。



レプリケーションの開始

■レプリケーションの開始は、フルリフレッシュから



●アプライによる自動フルリフレッシュ、
あるいはユーザーによるマニュアル・フルリフレッシュが可能



解説:

■レプリケーションの開始は、フルリフレッシュから

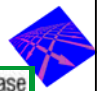
- レプリケーション(差分コピー)を開始するために、ソース表とターゲット表の同期を取る作業
- アプライによる自動フルリフレッシュ、あるいはユーザーによるマニュアル・フルリフレッシュが可能
 - アプライによる自動フルリフレッシュ
 - ソース表からのSELECT、およびターゲット表へのINSERT
 - ソース表からのEXPORT、およびターゲット表へのLOADまたはIMPORT
 - ニックネームからのCursor LOAD、Importユーティリティの利用の選択が可能
 - ターゲットがニックネームの場合にはImportユーティリティのみ選択可能
 - ニックネームに対するロードはサポートされていない
- フルリフレッシュの開始のシグナルを受け取った時点から、キャプチャーは変更情報を収集し始める
 - フルリフレッシュ中もキャプチャーにより変更収集が行われ、Export、LOAD中の変更もフルリフレッシュ完了前に適用しデータの整合性は保たれる
- フルリフレッシュの間、ソース表へのアクセスは可能(読み取り・書き込み)

■差分コピー

- ソース表に実行されたInsert、Update、Delete処理のレプリケーション
 - ログに書き込まれた情報をキャプチャーが読み取り、変更データをCD表に、また作業単位情報をUOW表にINSERT
 - ログに書かれないLOADやnull importによる削除はレプリケーション不可
 - フルリフレッシュの終了後、次のサイクルでアプライは差分適用を開始



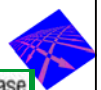
レプリケーション・ソリューションの プランニング



DB2 Universal Database

内容

- レプリケーション・ソリューションのプランニング
 - レプリケーション・ソリューションの適用
 - レプリケーションの構成と組み合わせ



DB2 Universal Database

SQLレプリケーション・ソリューションの適用

- SQLレプリケーション・ソリューションが適用可能かの判断基準
- ソフトウェアのレベル
 - 現在サポートされているSQLレプリケーションのアーキテクチャー・レベルはバージョン8
- レプリケーションの対象の更新処理
 - 変更収集が可能な処理はSQLのINSERT・UPDATE・DELETEのみ
 - 変更収集が不可の処理の例
 - LOADユーティリティによる挿入 (DB2 for z/OS、DB2 UDB for LUW)
 - REPLACEオプションのIMPORTユーティリティによる、既存データの削除処理部分 (DB2 UDB for LUW)
 - DDL (CREATE・DROPなどのデータ定義言語) による処理
- レプリケーション対象の列のタイプ
 - サポートされない特殊なデータタイプ
 - レプリケーションは可能だが、制約のあるタイプ
- レプリケーション対象データのユニーク性
 - 差分レプリケーションの場合、ターゲット表はデータ行を一意に決定するためにユニークとなる列の組み合わせは存在するか



解説:

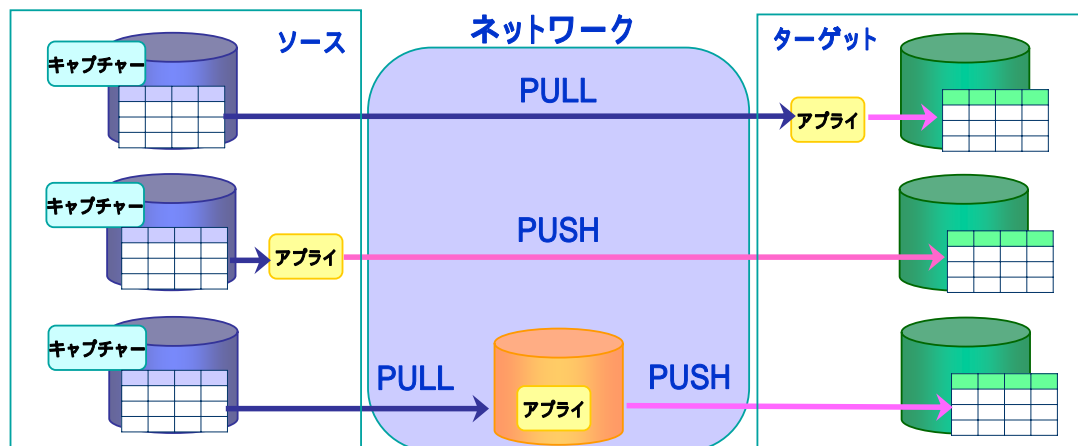
- データ・コピーの要件をSQLレプリケーション・ソリューションで解決しようとした場合、まず対象の環境がレプリケーション・ソリューションで適用可能かどうかを判断する必要があります。
- その際、考えられる主な要因は以下のとおりです。
- ソフトウェアのレベル
 - 現在サポートされているSQLレプリケーションのアーキテクチャー・レベルはバージョン8ですので、バージョン8のレプリケーションをサポートしていないDB2のレベルの場合は、そこでキャプチャーやアプライのプログラムを移動させることはできません。
 - またDB2の接続もバージョンの組み合わせによっては、サポートされていない場合があるので、その場合、SQLレプリケーションは使用できません。
- レプリケーションの対象の更新処理
 - 変更収集が可能な処理はSQLのINSERT・UPDATE・DELETEのみです。
 - よって、以下のような処理は変更収集ができないため、他の手段を考える必要があります。
 - LOADユーティリティによる挿入 (DB2 for z/OS、DB2 UDB for LUW)
 - REPLACEオプションのIMPORTユーティリティによる、既存データの削除処理部分 (DB2 UDB for LUW)
 - DDL (CREATE・DROPなどのデータ定義言語) による処理
- レプリケーション対象の列のタイプ
 - サポートされない特殊なデータタイプは以下のものです。(レプリケーションV8.2レベル)
 - DB2 以外のリレーショナル・ソースからの LOB 列
 - EDITPROC、FIELDPROC、VALIDPROCのいずれかのプロシージャが定義されている場所の列
 - 要約データ・タイプを含む表を複製することはできません。
 - レプリケーションは可能だが、制約のあるタイプ
 - ソース表およびターゲット表が DB2 for z/OS にある場合は、長い可変 GRAPHIC (LONG VARGRAPHIC) データ、長い可変文字 (LONG VARCHAR) データでは、ソース・データベース表がDB2 for z/OS にあるか、またはソース表とターゲット表の両方が DB2 Universal Database (Windows、Linux、および UNIX 版) にあることが必要です。
 - DB2 では、空間データ・タイプ列を含む表を複製することはできません、実際の空間データ・タイプ列を複製することはできません。
 - ユーザー定義のデータ・タイプ (DB2 Universal Database の特殊データ・タイプ) は、複製される前に変更データ (CD) 表で基本データ・タイプに変換されます。さらに、DB2 レプリケーションがターゲット表をサブスクリプション・セット・メンバー定義として作成した場合、ユーザー定義タイプは CD 表内と同様に、ターゲット表内で基本データ・タイプに変換されます。
- レプリケーション対象データのユニーク性
 - 差分レプリケーションの場合、ターゲット表はデータ行を一意に決定するためにユニークとなる列の組み合わせが存在する必要があります。
 - もし存在しない場合は、差分レプリケーションではなく、毎回全件置換えを検討する必要があります。



レプリケーションの構成

- アプライの、CD表からの照会処理またはターゲット表の更新処理のどちらがネットワークを経由する処理になるかによって、以下の3つの形態がある

- PULL構成
- PUSH構成
- PULL・PUSH混合構成



解説:

- アプライの、CD表からの照会処理またはターゲット表の更新処理のどちらがネットワークを経由する処理になるかによって、レプリケーションの構成は以下の3つの形態が考えられます。
- SELECT処理ではデータをブロックして、受信することが可能なため、同じ件数をリモート更新する場合と比較して、数倍のパフォーマンスアップが期待できます。よってPULL構成にするのが一般的ですが、その他の要件によっては、PUSH構成、PULL・PUSH構成も有効です。
- PULL構成
 - アプライがターゲット・サーバー側で稼動し、CD表のリモート照会、ターゲット表に対してローカル更新を行う形態
 - アプライ・コントロール・サーバーとターゲット・サーバーが同一
- PUSH構成
 - アプライがキャプチャー・サーバー側で稼動し、CD表のローカル照会ターゲット表に対してリモート更新を行う形態
 - アプライ・コントロール・サーバーとキャプチャー・サーバーが同一
- PULL・PUSH混合構成
 - アプライがキャプチャー・サーバー、ターゲット・サーバーとは別のサーバーで稼動し、CD表のリモート照会、ターゲット表に対してリモート更新を行う形態



レプリケーションの構成の比較

■ PULL構成

- SELECT処理ではデータをブロックングして、送受信することが可能なため、同じ件数をリモート更新する場合と比較して、数倍のパフォーマンスアップが期待できる
 - 一回のレプリケーションでの対象更新データ量が多く、転送時間を短くしたい場合に有効

■ PUSH構成

- 1つのサーバー側でレプリケーション管理が可能
 - ターゲット・サイトにシステム管理者がいない場合などに有効
- ターゲット側でV8レベルのアプライ機能を稼動できない場合の代替策
 - ターゲット側のDB2がV8レベルのレプリケーションをサポートしていないなど

■ PULL・PUSH混合構成

- ターゲット・サイトにシステム管理者がいない場合に有効
- ターゲット側でV8レベルのアプライ機能を稼動できない場合の代替策
- 非DB2データベースとのレプリケーションで考えられる構成
 - 他社DBが稼動するシステムの資源に余裕がなく、フェデレーティッド・サーバーを別にする場合など



DB2 Universal Database

解説:

■ PULL構成

- アプライはCD表から適用すべき更新情報をSELECTし、その結果を1行ずつINSERT/UPDATE/DELETE処理に変更してターゲット表に適用します。
- SELECT処理では1回のSEND/RECEIVEで、データを32Kのブロックにまとめて、転送できますが、アプライの更新処理は1回のSEND/RECEIVEで、1INSERT/UPDATE/DELETEしか転送できません。
- CD表からある行数をSELECTするという処理と、同じ行数分のINSERT /UPDATE /DELETEを比較すると、SELECT処理の方でネットワークを経由する構成にした方が、数倍のパフォーマンスアップが期待できます。
 - 行長が長いと1回にブロックングできる行数も減るため、効果は低下します。
- よって一回のレプリケーションでの対象更新データ量が多い場合は、PULL構成が有効です。

■ PUSH構成

- メインフレームから分散系のDB2にレプリケーションするような場合、分散系のサイトにはシステム管理者などが不在の場合もあります。
- またレプリケーションの運用は、全てメインフレーム側で制御したいという場合もあります。
- このような場合、キャプチャーもアプライもメインフレーム側で動かすPUSH構成が有効です。
- またターゲットのDB2システムではV8レベルのレプリケーションをサポートしていないが、DB2の接続としてはサポートされている組み合わせであるなら、PUSH構成にすることで、ターゲット側ではアプライ・プログラムを動かさず、表の更新だけをリモートから行うことが可能です。

■ PULL・PUSH混合構成

- PUSH構成の場合、キャプチャーもアプライも同じサーバーで稼動させるため、より多くのシステム資源を必要とします。よってできればアプライは別システムで稼動させたいが、上記のPUSH構成の場合の要件のように、ターゲット・サイトにシステム管理者がいないとか、ターゲット側でV8レベルのアプライ機能を稼動できない場合に、PULLとPUSHの混合構成にすることも可能です。
- また非DB2データベースとのレプリケーションで、例えば他社DBが稼動するシステムの資源に余裕がなく、フェデレーティッド・サーバーを別にしたいという場合も、この構成になります。

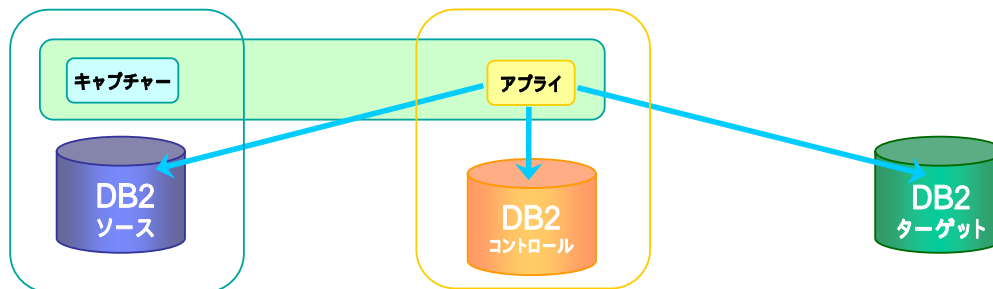


DB2 Universal Database

レプリケーションの組み合わせに関して

- 異なるプラットフォームやバージョンのDB2間でレプリケーションを行うためには、以下の3点が満たされている必要がある

- ・レプリケーション機能とDB2の組み合わせが、サポートされていること
- ・キャプチャーとアプライのバージョンが同じであること
 - V8どうし（あるいはV7どうし）
- ・アプライをDB2クライアント、ソースDB、ターゲットDB、およびコントロールDBをDB2サーバーとした場合、クライアントとサーバー間で、DB2としての接続が可能であること



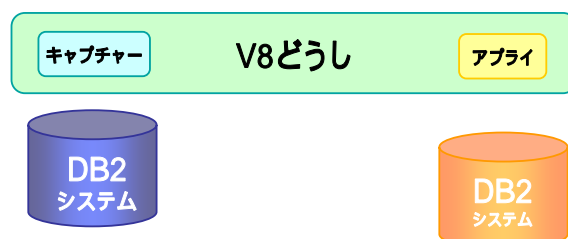
1. レプリケーション機能とDB2の組み合わせ

- オープン系のDB2 UDBでは、レプリケーションはDB2の機能の一部として提供されているため、UDBのバージョンとレプリケーション機能のバージョンは同じになる
 - DB2 UDB for LUW V8に含まれるレプリケーション機能はV8ベース
 - DB2 UDB for LUW V7に含まれるレプリケーション機能はV7ベース
- OS/390(およびz/OS)のDB2では、レプリケーションとDB2本体は別の製品
 - DB2のバージョンとレプリケーション製品のバージョンは同じである必要はない
 - ただしOS/390およびz/OSのレプリケーション製品(V8)は、リリースレベルによって、サポートされているDB2(OS/390およびz/OS)のバージョンが異なるので注意
 - DB2 DataPropagator for z/OS V8.1
 - DB2 V6,V7,V8
 - WebSphere Information Integrator Replication for z/OS V8.2
 - DB2 V7,V8



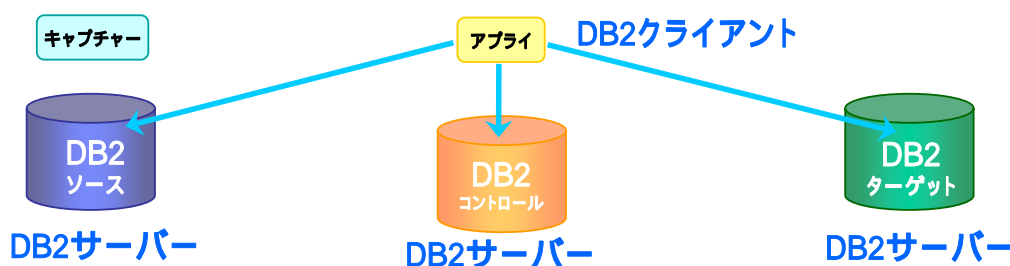
2. キャプチャーとアプライの組み合わせ

- レプリケーション製品(機能)はV7とV8で内部的な仕組みが大きく変更
- そのため基本的にキャプチャーとアプライのバージョンは一致している必要がある
 - V7とV8で異なる組み合わせは、V7からV8への移行期間のみのサポートであり、新規にレプリケーション機能を使用する場合は、V8で統一させなければならない



3. DB2の接続

- オープン系のDB2の場合、サポートされているDB2のクライアントとサーバーの接続は、従来以下のルールがあった
 - クライアントから見て、サーバーは2レベルアップのバージョンまで
 - クライアントから見て、サーバーは1レベルダウンのバージョンまで
 - つまり
 - クライアントがV8の場合、接続可能なサーバーはV7, V8, (V9, V10?)
 - クライアントがV7の場合、接続可能なサーバーはV6, V7, V8, (V9?)
 - ただしこのルールは将来変更される可能性はあり
 - オープン系のDB2とホスト系のDB2の組み合わせの場合、上記のようなルールはないが、オープン系のDB2 UDB V8からの接続がサポートされているホストDB2(OS/390)はV6以降
- レプリケーションを考えた場合、キャプチャーは必ずソース表があるDBのローカルで稼動する必要があるが、アプライはDB2としての接続が可能であれば、どこでも稼動させることは可能
- その場合、アプライをDB2クライアントとし、ソースDB、ターゲットDB、およびコントロールDBをサーバーと考え、まず上記のルールを満たす必要がある



参考:レプリケーション可能な組み合わせ

■以降のページでレプリケーション可能なDB2のレベルとレプリケーションのレベルの組み合わせを提示します。

- 以降の組み合わせは、レプリケーションを行うための3点の条件を満たすということを前提に考えられる組み合わせを記載したもの
 - ケースによっては前例が少なく、構築段階で何らかの問題が発生する可能性がある場合も考えられることを、予めご承知ください。

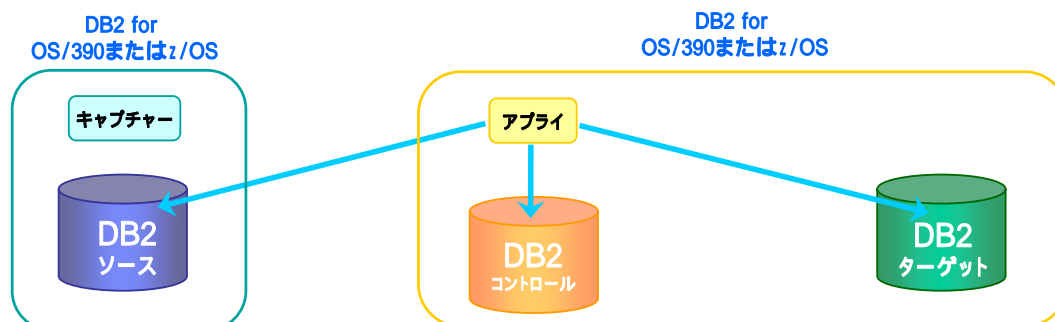


DB2. Universal Database

参考:レプリケーション可能な組み合わせ

■ホストDB2どうし (OS/390またはz/OS)

ソースDB (HOST)	キャプチャー (HOST)	コントロールDB (HOST)	アプライ (HOST)	ターゲットDB (HOST)
V8,V7	V8.2	V8,V7	V8.2	V8,V7
V8,V7	V8.2	V8,V7,V6	V8.1	V8,V7V6
V8,V7 V6はApplyが サポートしていないので不可	V8.1	V8,V7	V8.2	V8,V7
V8,V7,V6	V8.1	V8,V7,V6	V8.1	V8,V7,V6

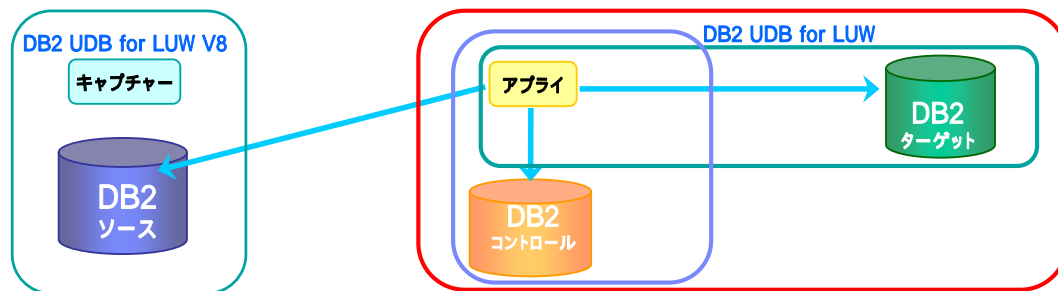


DB2. Universal Database

参考：レプリケーション可能な組み合わせ

■オープン系DB2どうし

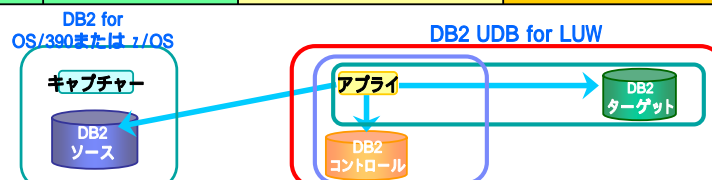
ソースDB (LUW)	キャプチャー (LUW)	コントロールDB (LUW)	アプライ (LUW)	ターゲットDB (LUW)
アプライはコントロールDBのインスタンスで稼働させ、ターゲットDBはリモートデータベースとなる場合 (青枠)				
V8	V8	V8	V8	V8、V7
アプライはターゲットDBのインスタンスで稼働させ、コントロールDBはリモートデータベースとなる場合 (緑枠) (コントロールDBはApplyからみてローカルDBであることが望ましいため、このような構成はお勧めしません。)				
V8	V8	V8、V7	V8	V8
アプライからみてターゲットDBとコントロールDBともローカルDBとなる場合 (赤枠)				
V8	V8	V8	V8	V8



参考：レプリケーション可能な組み合わせ

- ソース：ホストDB2 ターゲット：オープン系DB2
- アプライ：オープン側

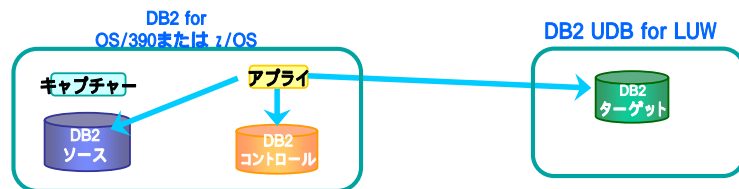
ソースDB (HOST)	キャプチャー (HOST)	コントロールDB (LUW)	アプライ (LUW)	ターゲットDB (LUW)
アプライはコントロールDBのインスタンスで稼働させ、ターゲットDBはリモートデータベースとなる場合 (青枠)				
DB2 V8,V7	V8.2	UDB V8	UDB V8	UDB V8、V7
DB2 V8,V7,V6	V8.1			
アプライはターゲットDBのインスタンスで稼働させ、コントロールDBはリモートデータベースとなる場合 (緑枠) (コントロールDBはApplyからみてローカルDBであることが望ましいため、このような構成はお勧めしません。)				
DB2 V8,V7	V8.2	UDB V8、V7	UDB V8	UDB V8
DB2 V8,V7,V6	V8.1			
アプライからみてターゲットDB,コントロールDBともローカルDBとなる場合 (赤枠)				
DB2 V8,V7	V8.2	UDB V8	UDB V8	UDB V8
DB2 V8,V7,V6	V8.1			



参考：レプリケーション可能な組み合わせ

- ソース：ホストDB2 ターゲット：オープン系DB2
- アプライ：ホスト側

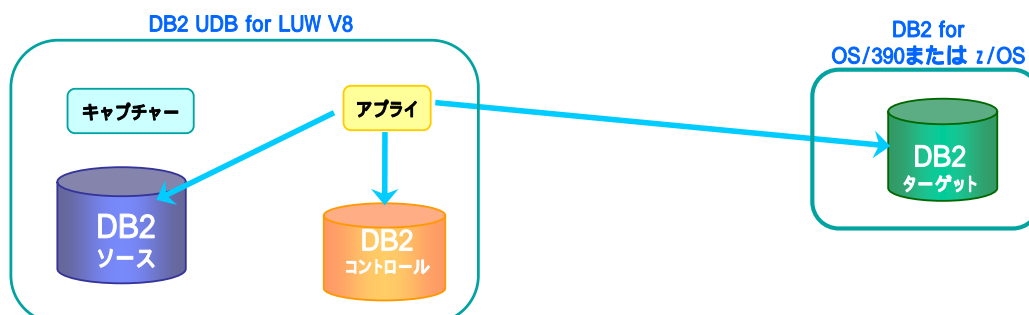
ソースDB (HOST)	キャプチャー (HOST)	コントロールDB (HOST)	アプライ (HOST)	ターゲットDB (LUW)
アプライはコントロールDBのDB2システムで稼働させ、ターゲットDBはリモートデータベースとなる場合				
DB2 V8,V7	V8.2	DB2 V8,V7	V8.2	UDB V8、V7
DB2 V8、V7	V8.2	DB2 V8,V7,V6	V8.1	UDB V8,V7
DB2 V8,V7 V6はApplyが サポートしていないので不可	V8.1	DB2 V8,V7	V8.2	UDB V8,V7
DB2 V8,V7,V6	V8.1	DB2 V8,V7,V6	V8.1	UDB V8,V7



参考：レプリケーション可能な組み合わせ

- ソース：オープン系DB2 ターゲット：ホストDB2
- アプライ：オープン側

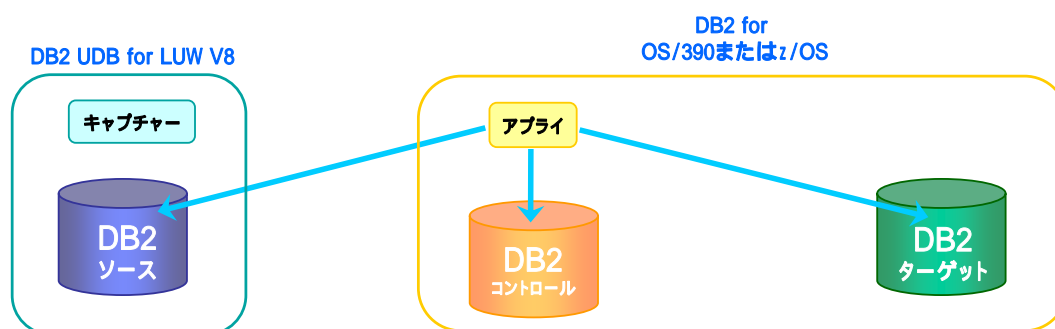
ソースDB (LUW)	キャプチャー (LUW)	コントロールDB (LUW)	アプライ (LUW)	ターゲットDB (HOST)
アプライはキャプチャーと同じインスタンス、又は別のインスタンスで稼働させ、ターゲットDBはリモートデータベースとなる場合				
UDB V8	UDB V8	UDB V8	UDB V8	DB2 V8、V7、V6



参考：レプリケーション可能な組み合わせ

- ソース：オープン系DB2 ターゲット：ホストDB2
- アプライ：ホスト側

ソースDB (LUW)	キャプチャー (LUW)	コントロールDB (HOST)	アプライ (HOST)	ターゲットDB (HOST)
ApplyからみてターゲットDB,コントロールDBともローカルDBとなる場合				
UDB V8	UDB V8	DB2 V8,V7	V8.2	DB2 V8、V7
UDB V8	UDB V8	DB2 V8,V7,V6	V8.1	DB2 V8、V7、V6



DB2. Universal Database

blank page



DB2. Universal Database

レプリケーション環境のプランニング



DB2 Universal Database

内容

■ レプリケーション環境のプランニング

- SQLレプリケーションに関連するオブジェクト、概念
 - DB2データベース・システム
 - データベース・オブジェクト
 - スピル・ファイル
 - サブスクリプション関連



DB2 Universal Database

SQLレプリケーションに関連するオブジェクト、概念

■ DB2データベース・システム

- データベース名(別名)
- データベース・コードページ
- ユーザーID、パスワード
- 権限

■ データベース・オブジェクト

- コントロール表
- ソース表
- 変更データ表(Change Data表)(CD表)
- ターゲット表
- データベース・ログ

■ スピル・ファイル

■ サブスクリプション関連



DB2 Universal Database

解説:

- SQLレプリケーション環境に関連するオブジェクト、概念には以下のようなものがあげられます。
- 以降ではそれぞれに関する説明と、レプリケーション環境のプランニング時に考慮しなければならない事項を取り上げます。

■ DB2データベース・システム

- データベース名(別名)
- データベース・コードページ
- ユーザーID、パスワード
- 権限

■ データベース・オブジェクト

- コントロール表
- ソース表
- 変更データ表(Change Data表)(CD表)
- ターゲット表
- データベース・ログ

■ スピルファイル

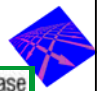
■ サブスクリプション関連



DB2 Universal Database

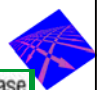
DB2データベース・システム

- データベース名(別名)
- データベース・コードページ
- ユーザーID、パスワード
- 権限



DB2 Universal Database

blank page

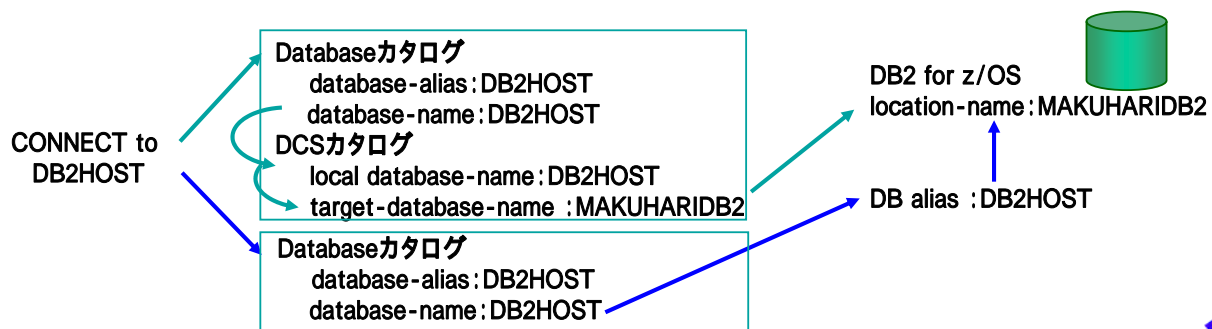


DB2 Universal Database

DB2データベース名(別名)

■ DB2クライアントからの接続時に指定される名前

- DB2 UDB for LUW
 - データベース別名
 - 8バイトまで
- DB2 for OS/390またはz/OS
 - ロケーション名
 - 16バイトまで
 - 8バイトより長い場合は、DB2 UDB for LUW側でDCSカタログを行い、データベース名を8バイト以下で指定
 - CATALOG DCS DATABASE DB2HOST AS MAKUHARIDB2
 - DB2 for z/OS V8ではロケーション名に別名を指定可能
 - 8バイト以下の別名を指定可能



解説:

■ DB2データベース名(別名)

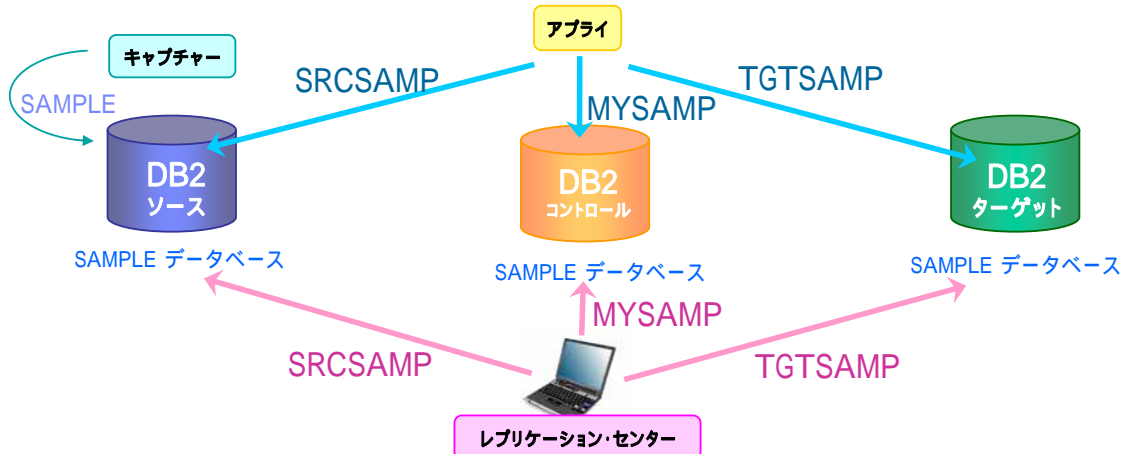
- ここで言うデータベース名とは、DB2クライアントからの接続時に指定される名前の中で、DB2 UDB for LUWの場合は、データベース別名、またDB2 for OS/390またはz/OSの場合はロケーション名に該当します。
- DB2 for OS/390またはz/OSのロケーション名は、16バイトまで指定できますが、DB2 UDB for LUWのデータベース別名は8バイトまでしか指定できません。
- そのため8バイトより長いロケーション名をDB2 UDB for LUW側からデータベース別名として、使用することができません。
- このような場合DB2 UDB for LUW側でDCSカタログを行い、8バイト以下のデータベース名を指定することが可能です。
 - CATALOG DCS DATABASE DB2HOST1 AS MAKUHARIDB2
- またDB2 for z/OS V8ではロケーション名に別名を指定することができるようになったので、DB2 for z/OS V8側で8バイト以下の別名を指定することも可能です。



DB2データベース名(別名)

■ レプリケーションの観点では

- アプリから見て、キャプチャー・コントロール・サーバー、アプリ・コントロール・サーバー、ターゲット・サーバーが全て一意に識別できるように、データベース名、または別名を分ける必要がある
- レプリケーション・センターを使用する場合は、アプリが接続する場合に使用する別名と同じ別名でキャプチャー・コントロール・サーバー、アプリ・コントロール・サーバー、ターゲット・サーバーに接続できるように環境を設定する



解説:

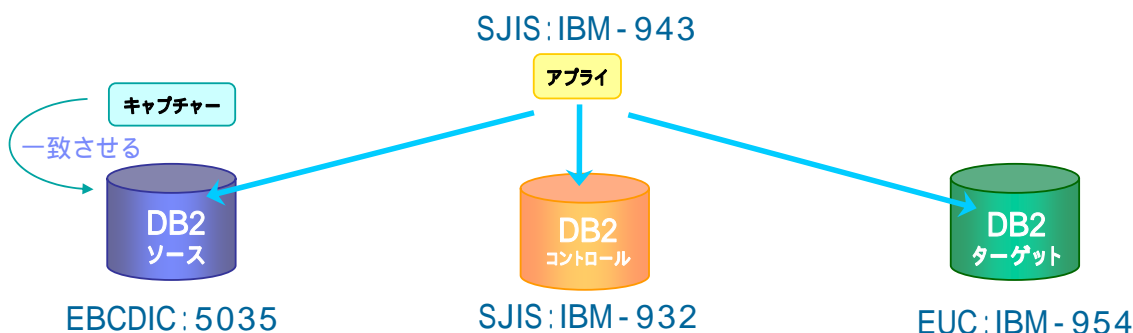
■ レプリケーションの観点では

- リモート・データベース接続があるのは、アプリのみです。
- そのためアプリから見て、キャプチャー・コントロール・サーバー、アプリ・コントロール・サーバー、ターゲット・サーバーが全て一意に識別できるように、データベース名、または別名を分ける必要があります。
- レプリケーション・センターを使用する場合は、アプリが接続する場合に使用する別名と同じ別名でキャプチャー・コントロール・サーバー、アプリ・コントロール・サーバー、ターゲット・サーバーに接続できるように環境を設定してから定義する必要があります。
- 例えば、全てのデータベース名がSAMPLEだった場合、アプリが稼動するインスタンスから各データベースへの接続をカタログする際に、それぞれ異なる別名をつけ、区別できるようにします。
- 上記の例ではSRC SAMP, MYSAMP, TGT SAMPと全てSAMPLEとは異なる別名を指定していますが、いずれかのデータベースはSAMPLEのままでも構いません。
- キャプチャーはアプリ・コントロール・サーバー、ターゲット・サーバーには接続しませんので、それぞれのデータベースに対して異なる別名を指定して、接続できる環境を設定する必要はありません。
- またアプリで使用するいずれかのデータベース別名がSAMPLEのままであっても、キャプチャーから見て、ローカルとなるデータベース名がSAMPLEのままでも構いません。
- なお1つの、キャプチャー・コントロール・サーバーに対して、複数のアプリがそれぞれ別のアプリ・コントロール・サーバーを使用して接続する場合、キャプチャー・コントロール表のIBMSNAP PRUNE SETには、TARGET_SERVER + APPLY_QUAL + SET_NAMEで、ユニーク索引が指定されているため、この組み合わせでユニークになるように命名する必要があります。



DB2データベース・コードページ

- 異なるコード・ページの環境でレプリケーションする場合は、以下の組み合わせの間で互換性があるコード・ページを使用しなければならない
 - ソース・データベースのコードページとアプライのコードページ
 - アプライのコードページとコントロール・データベースのコードページ
 - アプライのコードページとターゲット・データベースのコードページ
- 互換性がない組み合わせの場合、レプリケーションの結果、文字化けや文字の欠落が発生する可能性がある
- キャプチャーはソース・データベースと一致させなければならない



解説:

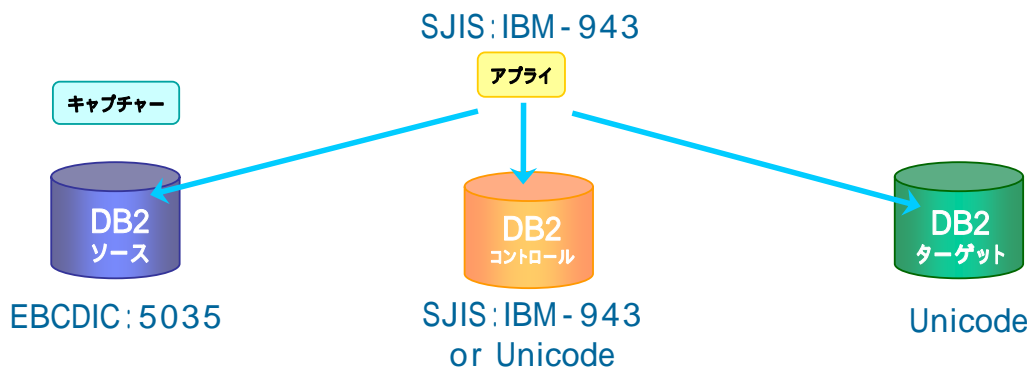
- 互換コード・ページを持つデータベース間でのデータのレプリケーション
 - 異なるコード・ページを使用するデータベース間でデータを複製する予定である場合は、「DB2 管理ガイド」をチェックして、持っているコード・ページが互換性のあるものかどうかを判別してください。たとえば、DB2 for Linux、DB2 for UNIX または DB2 for Windows を使用している場合は、文字データの変換に関するセクションを参照してください。
 - いくつかのデータベース製品のインプリメント・コード・ページのサポートは、他のコード・ページのサポートと異なることがあり、そのことがレプリケーション構成に影響する場合があります。たとえば、iSeries (OS/400) 上の現在の DB2 では、コード・ページの列レベルでの指定を許可していますが、DB2 (Linux、UNIX、および Windows 版) ではデータベース・レベルでの指定しか許可されていません。したがって、異なるコード・ページを使用する複数の列を持つ OS/400 表がある場合、すべてのコード・ページに互換性がある場合を除き、これらの列を単一の DB2 (Linux、UNIX、および Windows 版) データベースに対して複製することはできません。
- レプリケーションに合わせた各国語サポート (NLS) の構成
 - レプリケーション用の NLS 構成は、システム間のデータベース接続をセットアップする際に定義されます。ただし、キャプチャー・プログラムを Linux、UNIX または Windows オペレーティング・システムで実行している場合は、キャプチャー・プログラムはデータをそこから取り込んでいるデータベースと同じコード・ページを使用する必要があります。キャプチャー・プログラムがその同じコード・ページを使用していない場合は、DB2CODEPAGE と呼ばれる DB2 環境変数または登録変数を設定する必要があります。
- コード・ページ変数の設定
 - DB2 は、アプリケーションのコード・ページをそのアプリケーションが実行されているアクティブ環境から導き出します。通常、DB2CODEPAGE 変数が設定されていない場合は、コード・ページはオペレーティング・システムによって指定される言語 ID から導き出されます。ほとんどの場合、データベースの作成時にデフォルトのコード・ページを使用したのであれば、この値はキャプチャー・プログラムにとって正しい値です。しかし、データベースの作成時にデフォルトのコード・ページ以外のコード・ページを明示的に指定した場合は、キャプチャー・プログラムのために DB2CODEPAGE 変数を設定する必要があります。そうしない場合、キャプチャー・プログラムがデータを CD 表に挿入する際にデータが正しく変換されない可能性があります。DB2CODEPAGE 変数に対して使用する値は、CREATE DATABASE ステートメントで指定する値と同じでなければなりません。DB2CODEPAGE 変数の設定については、「DB2 管理ガイド」を参照してください。



DB2データベース・コードページ

■ UnicodeやEUCデータベースへのレプリケーション時の注意事項

- UnicodeやEUCでは、SJISやEBCDICでの日本語の半角カタカナやひらがな、漢字などは2バイト又は3バイトになるため、ターゲット表の列定義はソース表の列定義よりも、長くする必要があります
- アプライを稼働させる場合、通常はアプライ・コントロール・サーバーやターゲット・サーバーと同じコードページで稼働させるが、ターゲット・サーバーがUnicodeやEUCデータベースの場合は、ソース・データベースのコードページに一致させる
 - データのトランケーションを防ぐため
 - 設定はDB2CODEPAGE
- ただしホストDB2がソース・データベースの場合、DB2CODEPAGEでEBCDICのコードページは指定できないため、SJIS用のIBM-943を指定



解説:

■ UnicodeやEUCデータベースへのレプリケーション

- UnicodeやEUCでは、SJISやEBCDICでの日本語の半角カタカナやひらがな、漢字などは2バイト又は3バイトになるため、ターゲット表の列定義はソース表の列定義よりも、長くする必要があります。
- アプライを稼働させる場合、通常はアプライ・コントロール・サーバーやターゲット・サーバーと同じコードページで稼働させますが、ターゲット・サーバーがUnicodeやEUCデータベースの場合は、ソース・データベースのコードページに一致させてください。そうしないとデータのトランケーションが発生する可能性があります。
- 設定はDB2CODEPAGEで指定しますが、ホストDB2がソース・データベースの場合、DB2CODEPAGEでEBCDICのコードページは指定できないため、SJIS用のIBM-943を指定して代用します。

■ 詳細に関しては、以下の「ザ・技術 テクニカルフラッシュ DM-04-024」を参照してください。

- 「DPROPR V8 Applyにおけるデータのトランケーション問題に関する注意事項」

■ その他、LANG 変数の設定

- Linux または UNIX システム上でキャプチャーおよびアプライ・プログラムを実行している場合、LANG 環境変数を設定する必要があるかもしれません。キャプチャーおよびアプライ・プログラムは、この環境変数の内容を使って、使用されている言語のメッセージ・ライブラリーを検索します。たとえば、LANG 環境変数が en_US に設定されると、キャプチャー・プログラムは DB2 2 インスタンスの /sqlib/msg/en_US サブディレクトリにある英語のメッセージ・ライブラリーを検索します。キャプチャーがメッセージ・ライブラリーを見つけれなかった場合、キャプチャー・トレース表 (ASN_IBMSNAP_TRACE) に書き込まれるすべてのメッセージは ASN0000S です。



ユーザーID、パスワード

■ユーザーID、パスワード管理

- キャプチャー
 - LUW
 - キャプチャー・プロセスを起動するユーザーID、パスワードが使用される
 - z/OS
 - キャプチャー・ジョブのUSER/PASSWORDで指定したユーザーID、パスワードが使用される
- アプライ
 - LUW
 - あらかじめパスワード・ファイルで、各DBに接続する時に使用するユーザーID、パスワードを指定しておく
 - z/OS
 - SYSIBM.USERNAMESで各DBに接続する時に使用するユーザーID、パスワードを指定しておく



DB2. Universal Database

解説:

■ユーザーID、パスワード管理

- キャプチャーはキャプチャー・コントロール・サーバーに対してローカル接続になりますが、アプライはキャプチャー・コントロール・サーバー、アプライ・コントロール・サーバー、ターゲット・サーバーに対して接続する必要があり、多くの場合リモート接続になるため、接続時にユーザーID、パスワードの指定が必要です。
- しかし実行時のパラメータで指定するのではなく、あらかじめ設定しておく必要があります。
- OS/390またはz/OSの場合は、アプライとは関係なくDRDAの設定として、リモートのデータベースに接続する場合のユーザーID、パスワードは、SYSIBM.USERNAMESカタログに登録しておきます。
- LUWでは、アプライ専用のパスワード・ファイルを作成しておく必要があります。

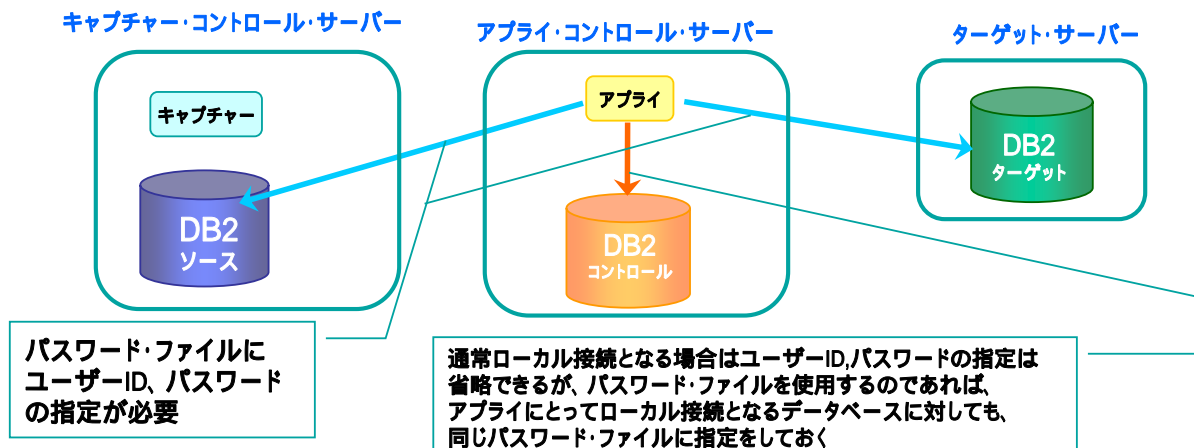


DB2. Universal Database

ユーザーID、パスワード

■ アプリのパスワード・ファイル(LUWプラットフォーム)

- アプリが接続するキャプチャー・コントロール・サーバー、ターゲット・サーバー、アプリ・コントロール・サーバーのいずれかでもアプリから見てリモートのデータベースになる場合で、そのサーバーに接続するためにユーザーIDとパスワードの指定が必要な場合(即ちconnect発行時にuser usingが必要な場合)は、パスワードファイルにてそれを指定しておく
 - 例) アプリ・コントロールサーバーでアプリが稼動する場合



解説:

■ パスワード・ファイル

- ファイル名: 任意
- デフォルト・ファイル名: asnpwd.aut

■ 作成

- asnpwdコマンドを使用して、作成、保守
- 新規作成
 - asnpwd {INIT} [ENCRYPT {ALL|PASSWORD}][[{USING} <file_path_name>]
- ファイルへのユーザーID、パスワードの登録
 - asnpwd {ADD} {ALIAS} <alias_name> {ID} <user_id> {PASSWORD} <password> [{USING} <file_path_name>]
 - (データベース単位)
- ユーザーID、パスワードの変更
 - asnpwd {MODIFY} {ALIAS} <alias_name> {ID} <user_id> {PASSWORD} <password> [{USING} <file_path_name>]
- 登録の削除
 - asnpwd {DELETE} {ALIAS} <alias_name> [{USING} <file_path_name>]

■ 使用

- アプリの開始パラメータ PWDFILEで指定
 - (デフォルト以外のファイル名の時)



レプリケーションに必要な権限

■ レプリケーションのセットアップに使用するユーザー

- 全てのサーバーへの接続
- DB2 システム・カタログへの照会
- 表、表スペース、ビューの作成
- プランのバインドまたはパッケージの作成
- ストアド・プロシージャの呼び出し

■ キャプチャー・プログラムを実行するユーザー

- DB2 システム・カタログへの照会
- キャプチャー・コントロール・サーバー上のすべてのレプリケーション・コントロール表への照会と更新
- その他

■ アプライ・プログラムを実行するユーザー

- DB2 システム・カタログの照会
- キャプチャー・コントロールおよびアプライ・コントロール・サーバー上のすべてのレプリケーション・コントロール表への照会と更新
- ターゲット・サーバー上のすべてのターゲット表の照会と更新
- その他



DB2 Universal Database

解説:

■ レプリケーション・センター(またはASNCPLP)を使用してレプリケーションのセットアップに使用するユーザー ID で以下のタスクを実行できることを確認してください。

- すべてのサーバー(ソース・サーバー、キャプチャー・コントロール・サーバー、アプライ・コントロール・サーバー、モニター・コントロール・サーバー、およびターゲット・サーバー)への接続
- ソース・サーバー、キャプチャー・コントロール・サーバー、モニター・コントロール・サーバー、およびターゲット・サーバーに置かれているカタログ表からの選択
- ソース・サーバー、モニター・コントロール・サーバー、キャプチャー・コントロール・サーバー、およびアプライ・コントロール・サーバーでの表(レプリケーション・コントロール表も含む)、表スペース、およびビューの作成
- ターゲット・サーバーでの表および表スペースの作成(新しいターゲット表の作成に DB2 レプリケーション・プログラムを使用する場合)(既存の表をターゲットとして使用する場合は必須ではありません)
- レプリケーションに関与するそれぞれの DB2 データベース(ソース・サーバー、ターゲット・サーバー、モニター・コントロール・サーバー、およびアプライ・コントロール・サーバーも含む)でのプランのバインドまたはパッケージの作成
- 共有ライブラリーを使用したストアド・プロシージャの作成 およびストアド・プロシージャの呼び出し(Linux、UNIX と Windows のみ)
- z/OSに対しては、SYSPROC.DSNWZPストアド・プロシージャをCALLするため、SYSPROC.DSNWZPストアド・プロシージャが稼動するためのOS、およびDB2側の設定が必要
 - 詳細については、以下のワークショップ資料を参照してください。
 - サ・技術 発行日 : 2005年10月21日
 - 「DRDA2005 WAS - DB2 for z/OS連携ワークショップ」
 - DB2 UDB V8クライアントとDB2 for z/OS提供ストアドプロシージャ

■ 使用するレプリケーション環境内のすべてのサーバーに対して同一の許可ユーザー ID を使用するか、あるいはそれぞれのサーバーごとに別々の許可ユーザー ID を使用するかは任意です。



DB2 Universal Database

解説:

■ キャプチャー・プログラムの許可要件

- キャプチャー・プログラムを実行するユーザー ID は、DB2 システム・カタログにアクセスできなければなりません。また、キャプチャー・コントロール・サーバー上のすべてのレプリケーション・コントロール表へのアクセスと更新が可能で、さらにキャプチャー・プログラム・パッケージを実行できなければなりません。レプリケーション管理者のユーザー ID を使用してキャプチャー・プログラムを実行できますが、これは要件ではありません。

■ Linux、UNIX、Windows の場合の要件

- キャプチャー・プログラムを実行するユーザー ID が以下の権限および特権を持っていることを確認してください。
 - DBADM または SYSADM 権限。
 - キャプチャー・パス・ディレクトリーに対する WRITE 特権。これは、キャプチャー・プログラムはキャプチャー・プログラムの始動時に指定された capture_path ディレクトリーに診断ファイルを作成するためです。

■ z/OS の場合の要件

- キャプチャー・プログラムを実行するために使用するユーザー ID は、USS にアクセスできるものとして登録する必要があります。これは、z/OS UNIX または OS/390 UNIX (OMVS セグメントを持っていないといけない) を使用するユーザー ID を定義する必要があることを意味します。
- また、キャプチャー・ロード・ライブラリーに APF 許可が与えられていること、およびキャプチャー・プログラムを実行するユーザー ID が以下の特権を持っていることも確認してください。
 - 一時ディレクトリー (/tmp ディレクトリーか、4 TMPDIR 環境変数によって指定されたディレクトリーのいずれか) への WRITE アクセス権。
 - キャプチャー CONTROL サーバー上のすべてのレプリケーション表に対する SELECT、UPDATE、INSERT、および DELETE 特権。
 - DB2 カタログ (SYSIBM.SYSTABLES および SYSIBM.SYSCOLUMNS) に対する SELECT 特権。
 - TRACE 特権。
 - MONITOR1 および MONITOR2 特権。
 - キャプチャー・プログラム・パッケージに対する EXECUTE 特権。
- また、そのユーザー ID にキャプチャー・パス・ディレクトリーに対する WRITE アクセス権があること (USS)、または上位修飾子が付けられていること (z/OS) を確認してください。キャプチャー・プログラムを USS シェルで実行するためには、STEPLIB システム変数が設定されていて、さらにこの変数にキャプチャー・ロード・ライブラリーが組み込まれている必要があります。PATH には HFS パス (| /usr/lpp/db2repl_08.01/bin) が含まれていなければなりません。



解説:

■ アプライ・プログラムの許可要件

- アプライ・プログラムを実行するユーザー ID は、DB2 システム・カタログにアクセスできなければなりません。また、キャプチャー・コントロールおよびターゲット・サーバー上のすべてのレプリケーション・コントロール表へのアクセスと更新が可能で、さらにアプライ・プログラム・パッケージを実行できなければなりません。レプリケーション管理者のユーザー ID を使用してアプライ・プログラムを実行できますが、これは要件ではありません。

■ Linux、UNIX、Windows の場合の要件

- アプライ・プログラムを実行するユーザー ID が以下の権限および特権を持っていることを確認してください。
 - アプライ・パス・ディレクトリーに対する書き込み特権
 - レプリケーション・ソース表 (関連した CD 表および CCD 表も含む) へのアクセス特権
 - レプリケーション・ターゲット表に対するアクセスおよび更新特権
 - DB2 レプリケーション・プログラムによって生成されて、キャプチャー・コントロール・サーバーおよびアプライ・コントロール・サーバーでビルドされたすべてのコントロール表に対するアクセスと更新特権
 - アプライ・プログラムによって使用される任意のパスワード・ファイルの読み取り特権
- 注: 使用するソース表が非 DB2 リレーショナル・データベース管理システム上にある場合は、ユーザー ID は DB2 フェデレーテッド・データベース および 非 DB2 リレーショナル・データベースの両方において、フェデレーテッド・データベースで定義されているニックネームを使用してソース表にアクセスできるだけの十分な特権を持っている必要があります。

■ z/OS の場合の要件

- アプライ・プログラムを実行するユーザー ID が以下の権限および特権を持っていることを確認してください。
 - 一時ディレクトリー (/tmp ディレクトリーか、4 TMPDIR 環境変数によって指定されたディレクトリーのいずれか) への WRITE アクセス権。
 - アプライ CONTROL サーバー上のすべてのレプリケーション表に 2 対する SELECT、UPDATE、INSERT、および DELETE 特権。
 - DB2 カタログ (SYSIBM.SYSTABLES および SYSIBM.SYSCOLUMNS) に対する SELECT 権限。
- 注: アプライ・プログラムを実行するために使用するユーザー ID は、USS にアクセスできるものとして登録する必要があります。4 ユーザー ID を定義する必要があることを意味します。ロード・ライブラリーに APF 許可が必要となるのは、アプライ・プログラムが ARM の指定付きで登録される場合 だけです。アプライ・プログラムを USS シェルで実行するためには、STEPLIB システム変数が設定されていて、さらにこの変数にアプライ・ロード・ライブラリーが組み込まれている必要があります。PATH には HFS パス (| /usr/lpp/db2repl_08.01/bin) が含まれていなければなりません。



データベース・オブジェクト

- レプリケーション制御表
- ソース表
- 変更データ表(Change Data表)(CD表)
- ターゲット表
- DB2のログ



blank page



データベース・オブジェクト

■ レプリケーション制御表

- キャプチャー用、アプライ用、モニター・プログラム用がある
- ソースがNon-IBM RDBの場合は、一部ニックネームとして持つ

■ ソース表

- 複写元となる表

■ 変更データ表 (Change Data表) (CD表)

- キャプチャーが収集した変更データを書き出す表

■ ターゲット表

- 複写先の表
- 用途によって、いくつかのタイプに分けられる

■ ログ

- キャプチャー・コントロール・サーバー側
- ターゲット・サーバー側



DB2 Universal Database

解説:

■ コントロール表

- データ・ソースとデータ・ターゲットについての情報、およびそれらの間でのデータのレプリケーションについての情報を格納するために、コントロール表(DB2表)が使用されます。

■ ソース表

- 複写元となる表のことです。

■ 変更データ表 (CD表)

- ステージング表とも呼ばれる
- V7までは1つのソース表に対してCD表は1つしか存在しなかったが、V8からは複数のキャプチャーの稼動が可能のため、別々のキャプチャーが同じソース表の変更データを収集する場合は、1つのソース表に対して複数のCD表が存在します。

■ ターゲット表

- SQLレプリケーションでは、以下のタイプがある
 - ユーザー・コピー表
 - ポイント・イン・タイム表
 - 整合変更データ(CCD)表
 - 集約表
 - レプリカ表

■ ログ

- レプリケーション環境を設定した場合、既存の環境より多くのログ情報を書き出します。
- またターゲットのデータベースには、レプリケーションという新しい更新アプリケーションが稼動することになりますので、ターゲット側のログも考慮する必要があります。



DB2 Universal Database

レプリケーション制御表

■ レプリケーション制御表(コントロール表)の種類

- キャプチャー・コントロール表
- アプライ・コントロール表
- モニター・コントロール表

■ 作成

- レプリケーション・センター
- ASNCLP
- サンプルDDL(スクリプト・ファイル)を修正使用
 - DB2 UDB for LUW用
 - sqllib/samples/repl/asnctlw.sql
 - DB2 for z/OS用
 - sqllib/samples/repl/zos/asnctlz.sql
 - DPROPR.V820.SASNSAMP(ASNCTLZD)



解説:

- コントロール表は3種類あります。キャプチャー・コントロール表とアプライ・コントロール表は、必ず作成する必要がありますが、モニター・コントロール表はモニター・プログラムを使用する場合のみ必要となります。
- コントロール表は次の以下のいずれかの方法で作成できます。
 - レプリケーション・センター
 - ASNCLP
 - スクリプト・ファイルを修正使用
 - DB2 UDB for LUW用
 - sqllib/samples/repl/asnctlw.sql
 - DB2 for z/OS用
 - sqllib/samples/repl/zos/asnctlz.sql
 - DPROPR.V820.SASNSAMP(ASNCTLZD)
 - スクリプト・ファイルには、非DB2データベースの表がソース表になる場合の制御表の作成サンプルは含まれていないため、非DB2データソースからのレプリケーションを定義する場合は、レプリケーション・センター又はASNCLPを使用して作成しなければなりません。



キャプチャー・コントロール表

■ キャプチャーが使用するコントロール表

- キャプチャーを複数稼働させる場合は、スキーマ名を別にして、以下のセットをそれぞれ作成する
 - ASN.IBMSNAP_CAPSCHEMAS以外
- 一部の表はアプライからも検索、更新される

	V8新規	内容変更	備考
ASN.IBMSNAP_CAPSCHEMAS			この表のスキーマ名は必ずASN
schema.IBMSNAP_CAPENQ (iSeries以外)			
schema.IBMSNAP_REGISTER			
schema.CD_table			
schema.CCD_table			
schema.IBMSNAP_UOW			
schema.IBMSNAP_CAPMON			
schema.IBMSNAP_CAPPARMS			V7ではIBMSNAP_CCPPARMS
schema.IBMSNAP_CAPTRACE			V7ではIBMSNAP_TRACE
schema.IBMSNAP_PRUNE_LOCK			
schema.IBMSNAP_PRUNE_SET			
schema.IBMSNAP_PRUNCNTL			
schema.IBMSNAP_RESTART			V7のIBMSNAP_WARMSTARTが置換わった
schema.IBMSNAP_SIGNAL			V7のIBMSNAP_CRITSECが置換わった
schema.IBMSNAP_PARTITIONINFO			V8 FixPack#2より追加
schema.IBMSNAP_AUTHTKN (iSeries)			
schema.IBMSNAP_REG_EXT (iSeries)			
schema.IBMSNAP_REG_SYNCH (非 DB2)			
schema.IBMSNAP_SEQTABLE (非 DB2)			

(非 DB2)は、トリガーベース・キャプチャーが使用する表を示します



解説:

- ASN.IBMSNAP_CAPSCHEMAS (キャプチャー・スキーマ表)
 - すべてのキャプチャー・スキーマの名前が含まれます。
- schema.IBMSNAP_CAPENQ (キャプチャー・エンキュー表) (iSeries以外)
 - それぞれのキャプチャー・スキーマごとに、この表を次の項目の確認に使用します。
 - UNIX および Windows の DB2 については、データベースごとにキャプチャー・プログラムを1つだけ実行している。
 - z/OS の非データ共有 DB2 については、サブシステムごとにキャプチャー・プログラムを1つだけ実行している。
 - z/OS の非データ共有 DB2 については、データ共有グループごとにキャプチャー・プログラムを1つだけ実行している。
- schema.IBMSNAP_REGISTER (登録表)
 - レプリケーション・ソース表の名前、その属性、および関連する CD 表および CCD 表の名前など、レプリケーション・ソースに関する情報が含まれます。
- schema.CD_table (変更データ (CD) 表)
 - ソースで発生した変更に関する情報が含まれます。ユーザーがレプリケーション・ソースを登録するまでこの表は作成されません。
- schema.CCD_table (整合のとれた変更データ (CCD) 表)
 - ソースで発生した変更およびその変更の順次配列を示す追加の列に関する情報が含まれます。
- schema.IBMSNAP_UOW (作業単位 (UOW) 表)
 - ソース表にコミットされたトランザクションに関する追加情報を提供します。
- schema.IBMSNAP_CAPMON (キャプチャー・モニター表)
 - キャプチャー・プログラムの進行のモニターに役立つ運用の統計が含まれます。
- schema.IBMSNAP_CAPPARMS (キャプチャー・パラメーター表)
 - キャプチャー・プログラムの運用を管理するために指定できるパラメーターが含まれます。



解説:

- schema.IBMSNAP_CAPTRACE (キャプチャー・トレース表)
 - キャプチャー・プログラムからの重要なメッセージが含まれます。
- schema.IBMSNAP_PRUNE_LOCK (ブルーニング・ロック表)
 - コールド・スタート中または保存期限付きブルーニング中の、キャプチャー・プログラムの CD 表のアクセスをシリアライズするために使用します。
- schema.IBMSNAP_PRUNE_SET (ブルーニング・セット表)
 - CD 表のブルーニングを調整します。
- schema.IBMSNAP_PRUNCNTL (ブルーニング・コントロール表)
 - 同期点の更新を調整します。
- schema.IBMSNAP_RESTART (再始動表)
 - キャプチャー・プログラムが、ログまたはジャーナルにある正確なポイントからキャプチャーを再開できるようにするための情報が含まれます。iSeries 環境については、この表は RCVJRNE (ジャーナルのエントリーを受け取る) コマンドの開始時刻を決定するために使用します。
- schema.IBMSNAP_SIGNAL (シグナル表)
 - キャプチャー・プログラムを促すために使用するすべてのシグナルが含まれます。これらのシグナルは、ユーザーまたはアプライ・プログラムのいずれかから発信される可能性があります。
- schema.IBMSNAP_PARTITIONINFO (パーティション情報表)
 - 複数のパーティションに分割された環境において再始動 (IBMSNAP_RESTART) 表を補強し、2各パーティションのログ・ファイルのセットのうちの、必要とされる最も古いログ・シーケンスからキャプチャー・プログラムを再始動するための情報を含んでいます。



解説:

- schema.IBMSNAP_AUTHTKN (アプライ修飾子相互参照表) (iSeries)
 - Update-anywhere をサポートする情報が含まれます。
- schema.IBMSNAP_REG_EXT (登録拡張表) (iSeries)
 - 登録表の拡張です。ジャーナル名およびリモート・ソース表のデータベース項目名など、レプリケーション・ソースに関する追加情報が含まれます。
- schema.IBMSNAP_REG_SYNCH (登録同期表) (非 DB2)
 - 非 DB2 のデータ・ソースから複製する場合、この表での更新トリガーにより、キャプチャー・プログラムのシミュレートを行います。アプライ・プログラムが登録表から情報を読み取る前に、すべての行について SYNCHPOINT 値の更新を開始します。
- schema.IBMSNAP_SEQTABLE (順序付け表) (非 DB2)
 - DB2 レプリケーションが Informix 表のためのログ ID またはジャーナル ID (同期点) として使用する、ユニーク番号の順序が含まれます。



アプライ制御表

■ アプライが使用するコントロール表

- スキーマ名は、全て必ずASN

	V8新規	内容変更
ASN.IBMSNAP_APPENQ (iSeries以外)		
ASN.IBMSNAP_APPLYTRACE		
ASN.IBMSNAP_APPLYTRAIL		
ASN.IBMSNAP_SUBS_SET		
ASN.IBMSNAP_SUBS_MEMBR		
ASN.IBMSNAP_SUBS_COLS		
ASN.IBMSNAP_SUBS_STMTS		
ASN.IBMSNAP_SUBS_EVENT		
ASN.IBMSNAP_APPPARMS		
ASN.IBMSNAP_COMPENSATE		
ASN.IBMSNAP_APPLY_JOB (iSeries)		



解説:

- ASN.IBMSNAP_APPENQ (アプライ・エンキュー表) (iSeries以外)
 - アプライ修飾子ごとに実行しているアプライ・プログラムが 1 つだけであることを確認するために使用します。
- ASN.IBMSNAP_APPLYTRACE (アプライ・トレース表)
 - アプライ・プログラムからの重要なメッセージが含まれます。
- ASN.IBMSNAP_APPLYTRAIL (アプライ・トレール表)
 - アプライ・プログラムに関する監査証拠情報が含まれます。
- ASN.IBMSNAP_SUBS_SET (サブスクリプション・セット表)
 - アプライ・プログラムがグループとして処理するサブスクリプション・セット・メンバーの各セットの 処理情報が含まれます。
- ASN.IBMSNAP_SUBS_MEMBR (サブスクリプション・メンバー表)
 - ソース表およびターゲット表のペアを識別し、そのペアについて処理情報を指定します。
- ASN.IBMSNAP_SUBS_COLS (サブスクリプション列表)
 - ターゲット表またはビューの列を、ソース表またはビューの関連のある列へマップします。
- ASN.IBMSNAP_SUBS_STMTS (サブスクリプション・ステートメント表)
 - サブスクリプション・セットについて定義した SQL ステートメントまたはストアド・プロシージャ呼び出しが含まれます。アプライ・プログラムがセットを処理する前または後で呼び出されます。
- ASN.IBMSNAP_SUBS_EVENT (サブスクリプション・イベント表)
 - アプライ・プログラムがサブスクリプション・セットをいつ処理するかを管理するために定義するイベントが含まれます。
- ASN.IBMSNAP_APPPARMS (アプライ・パラメーター表)
 - アプライ・プログラムの操作をコントロールするためにユーザーが変更できるパラメーターを保持します。
- ASN.IBMSNAP_COMPENSATE (アプライ補償表)
 - UPDATE ANYWHERE(双方向レプリケーション)を構成した時に使われる可能性のある表です。
 - コンフリクトが発生して、トランザクションを補償するための処理(コンペンセイト)を行う場合で、かつ 1 セットに150メンバー以上定義されている時にだけワーク的に使用します。
- ASN.IBMSNAP_APPLY_JOB (アプライ・ジョブ表) (iSeries)
 - アプライ・コントロール・サーバーで実行しているアプライ・プログラムの各インスタンスについてユニークのアプライ修飾子があるかどうかを確認するために使用します。



モニター制御表

■ モニター・プログラムが使用するコントロール表

- スキーマ名は、全て必ずASN

	V8 新規
ASN.IBMSNAP_CONTACTGRP	
ASN.IBMSNAP_CONDITIONS	
ASN.IBMSNAP_CONTACTS	
ASN.IBMSNAP_GROUPS	
ASN.IBMSNAP_MONENQ	
ASN.IBMSNAP_MONSERVERS	
ASN.IBMSNAP_MONTRAIL	
ASN.IBMSNAP_MONTRACE	
ASN.IBMSNAP_MONPARMS	



解説:

- ASN.IBMSNAP_CONTACTGRP (モニター・アラート連絡先表)
 - 各グループに属する連絡先を保管します。
- ASN.IBMSNAP_ALERTS (モニター・アラート表)
 - 発行済みのアラートを保管します。
- ASN.IBMSNAP_CONDITIONS (モニター条件表)
 - キャプチャー・コントロール・サーバー、アプライ・コントロール・サーバーおよび現行のモニター・コントロール・サーバーでモニターするためのアラート条件が含まれます。
- ASN.IBMSNAP_CONTACTS (モニター連絡先表)
 - この表には、名前、常用のEメール・アドレスおよび携帯電話のEメール・アドレスおよび各連絡先の記述が保管されます。アラート条件ごとの連絡先を、この表から選択します。
- ASN.IBMSNAP_GROUPS (モニター・グループ表)
 - この表には、連絡先の各グループの名前および記述が保管されます。
- ASN.IBMSNAP_MONENQ (モニター・エンキュー表)
 - この表を使用して、実行されているモニター処理がモニター修飾子ごとに1つだけであることを確認します。
- ASN.IBMSNAP_MONSERVERS (モニター・サーバー表)
 - この表は、キャプチャー・コントロール・サーバーおよびアプライ・コントロール・サーバーが、モニター修飾子で識別されたモニター・プログラムによって最後にモニターされた時刻を記録します。
- ASN.IBMSNAP_MONTRAIL (モニター・トレール表)
 - この表には、モニター・サイクルの履歴を保持します。
- ASN.IBMSNAP_MONTRACE (モニター・トレース表)
 - この表は、キャプチャー・トレース (IBMSNAP_CAPTRACE) 表およびアプライ・トレール (IBMSNAP_APPLYTRAIL) 表と同じように使用されます。モニター・プログラムのオペレーションのログです。
- ASN.IBMSNAP_MONPARMS (モニター・パラメーター表)
 - モニター・プログラムの操作をコントロールするためにユーザーが変更できるパラメーターを保持します。



制御表の表スペース- DB2 UDB for LUWの場合

■ レプリケーション・センターを使用して作成した場合のデフォルト設定

- キャプチャー・コントロール表用の表スペース名
 - TS<スキーマ名>CA
 - TS<スキーマ名>UOW
 - IBMSNAP_UOW表用
- アプライ・コントロール表用の表スペース名
 - TS<スキーマ名>AA

キャプチャー・コントロール表

TSASNCA

ASN.IBMSNAP_CAPENQ
ASN.IBMSNAP_REGISTER
ASN.IBMSNAP_CAPMON
ASN.IBMSNAP_CAPPARMS
ASN.IBMSNAP_CAPTRACE
ASN.IBMSNAP_CAPSCHEMAS
ASN.IBMSNAP_PRUNE_LOCK
ASN.IBMSNAP_PRUNE_SET
ASN.IBMSNAP_PRUNCNTL
ASN.IBMSNAP_RESTART
ASN.IBMSNAP_SIGNAL

TSASNUOW

ASN.IBMSNAP_UOW

アプライ・コントロール表

TSASNAA

ASN.IBMSNAP_APPENQ
ASN.IBMSNAP_APPLYTRACE
ASN.IBMSNAP_APPLYTRAIL
ASN.IBMSNAP_SUBS_SET
ASN.IBMSNAP_SUBS_MEMBR
ASN.IBMSNAP_SUBS_COLS
ASN.IBMSNAP_SUBS_STMTS
ASN.IBMSNAP_SUBS_EVENT
ASN.IBMSNAP_APPPARMS
ASN.IBMSNAP_COMPENSATE



制御表の表スペース- DB2 for z/OSの場合

■ レプリケーション・センターを使用して作成した場合のデフォルト設定

■ ロックサイズが異なる

- キャプチャー・コントロール表用の表スペース名
 - TS<スキーマ名>CR (行ロック)
 - TS<スキーマ名>CP (ページロック)
 - TS<スキーマ名>UOW (ページロック)
 - IBMSNAP_UOW表用
- アプライ・コントロール表用の表スペース名
 - TS<スキーマ名>AR (行ロック)
 - TS<スキーマ名>AP (ページロック)

レプリケーション・センターのデフォルトの設定では、このように1表スペースに複数表が定義されるようになってますが、ロックの競合などの問題が発生する可能性があるため、それぞれのロックのサイズはそのまま、1表スペース1表になるように定義してください。サンプルDDLを使用する場合も同様！

TS<スキーマ名>CR (LOCK=ROW)

ASN.IBMSNAP_REGISTER
ASN.IBMSNAP_PRUNE_SET
ASN.IBMSNAP_PRUNCNTL
ASN.IBMSNAP_SIGNAL
ASN.IBMSNAP_CAPSCHEMAS
ASN.IBMSNAP_COMPENSATE

TS<スキーマ名>UOW (LOCK=PAGE)

ASN.IBMSNAP_UOW

TS<スキーマ名>CP (LOCK=PAGE)

ASN.IBMSNAP_CAPENQ
ASN.IBMSNAP_CAPMON
ASN.IBMSNAP_CAPPARMS
ASN.IBMSNAP_CAPTRACE
ASN.IBMSNAP_PRUNE_LOCK
ASN.IBMSNAP_RESTART

TS<スキーマ名>AR (LOCK=ROW)

ASN.IBMSNAP_APPENQ
ASN.IBMSNAP_APPLYTRACE
ASN.IBMSNAP_APPLYTRAIL
ASN.IBMSNAP_SUBS_SET
ASN.IBMSNAP_SUBS_EVENT
ASN.IBMSNAP_APPPARMS

TS<スキーマ名>AP (LOCK=PAGE)

ASN.IBMSNAP_SUBS_MEMBR
ASN.IBMSNAP_SUBS_COLS
ASN.IBMSNAP_SUBS_STMTS
ASN.IBMSNAP_APPPARMS
ASN.IBMSNAP_COMPENSATE



制御表の表スペース-モニター制御表

■ レプリケーション・センターを使用して作成した場合のデフォルト設定

- アラート、トレース関連以外
 - REPLMONTS1
- アラート用
 - REPLMONTS2
- トレース関連
 - REPLMONTS3

REPLMONTS1

ASN.IBMSNAP_CONTACTGRP
ASN.IBMSNAP_ALERTS
ASN.IBMSNAP_CONDITIONS
ASN.IBMSNAP_CONTACTS
ASN.IBMSNAP_GROUPS
ASN.IBMSNAP_MONENQ
ASN.IBMSNAP_MONSERVERS

REPLMONTS2

ASN.IBMSNAP_ALERTS

REPLMONTS3

ASN.IBMSNAP_MONTRAIL
ASN.IBMSNAP_MONTRACE



DB2 Universal Database

blank page



DB2 Universal Database

参考: キャプチャー・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_REGISTER(
  SOURCE_OWNER          VARCHAR(30) NOT NULL,
  SOURCE_TABLE          VARCHAR(128) NOT NULL,
  SOURCE_VIEW_QUAL      SMALLINT NOT NULL,
  GLOBAL_RECORD         CHAR( 1) NOT NULL,
  SOURCE_STRUCTURE      SMALLINT NOT NULL,
  SOURCE_CONDENSED      CHAR( 1) NOT NULL,
  SOURCE_COMPLETE       CHAR( 1) NOT NULL,
  CD_OWNER              VARCHAR(30),
  CD_TABLE              VARCHAR(128),
  PHYS_CHANGE_OWNER     VARCHAR(30),
  PHYS_CHANGE_TABLE     VARCHAR(128),
  CD_OLD_SYNCHPOINT     CHAR( 10) FOR BIT DATA,
  CD_NEW_SYNCHPOINT     CHAR( 10) FOR BIT DATA,
  DISABLE_REFRESH       SMALLINT NOT NULL,
  CCD_OWNER             VARCHAR(30),
  CCD_TABLE             VARCHAR(128),
  CCD_OLD_SYNCHPOINT    CHAR( 10) FOR BIT DATA,
  SYNCHPOINT           CHAR( 10) FOR BIT DATA,
  SYNCHTIME            TIMESTAMP,
  CCD_CONDENSED         CHAR( 1),
  CCD_COMPLETE         CHAR( 1),
  ARCH_LEVEL           CHAR( 4) NOT NULL,
  DESCRIPTION           CHAR(254),
  BEFORE_IMG_PREFIX     VARCHAR( 4),
  CONFLICT_LEVEL        CHAR( 1),
  CHG_UPD_TO_DEL_INS   CHAR( 1),
  CHGONLY              CHAR( 1),
  RECAPTURE            CHAR( 1),
  OPTION_FLAGS         CHAR( 4) NOT NULL,
  STOP_ON_ERROR        CHAR( 1) WITH DEFAULT 'Y',
  STATE               CHAR( 1) WITH DEFAULT 'I',
  STATE_INFO           CHAR( 8))
IN TSASNCA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_REGISTERX
ON ASN.IBMSNAP_REGISTER(
  SOURCE_OWNER          ASC,
  SOURCE_TABLE          ASC,
  SOURCE_VIEW_QUAL      ASC);

CREATE INDEX ASN.IBMSNAP_REGISTERX1
ON ASN.IBMSNAP_REGISTER(
  PHYS_CHANGE_OWNER     ASC,
  PHYS_CHANGE_TABLE     ASC);

CREATE INDEX ASN.IBMSNAP_REGISTERX2
ON ASN.IBMSNAP_REGISTER(
  GLOBAL_RECORD         ASC);
```

IBMSNAP_REGISTER



参考: キャプチャー・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_PRUNCNTL(
  TARGET_SERVER          CHAR(18) NOT NULL,
  TARGET_OWNER          VARCHAR(30) NOT NULL,
  TARGET_TABLE          VARCHAR(128) NOT NULL,
  SYNCHTIME             TIMESTAMP,
  SYNCHPOINT            CHAR( 10) FOR BIT DATA,
  SOURCE_OWNER          VARCHAR(30) NOT NULL,
  SOURCE_TABLE          VARCHAR(128) NOT NULL,
  SOURCE_VIEW_QUAL      SMALLINT NOT NULL,
  APPLY_QUAL            CHAR( 18) NOT NULL,
  SET_NAME              CHAR( 18) NOT NULL,
  CNTL_SERVER           CHAR( 18) NOT NULL,
  TARGET_STRUCTURE      SMALLINT NOT NULL,
  CNTL_ALIAS            CHAR( 8),
  PHYS_CHANGE_OWNER     VARCHAR(30),
  PHYS_CHANGE_TABLE     VARCHAR(128),
  MAP_ID               VARCHAR(10) NOT NULL)
IN TSASNCA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_PRUNCNTLX
ON ASN.IBMSNAP_PRUNCNTL(
  SOURCE_OWNER          ASC,
  SOURCE_TABLE          ASC,
  SOURCE_VIEW_QUAL      ASC,
  APPLY_QUAL            ASC,
  SET_NAME              ASC,
  TARGET_SERVER         ASC,
  TARGET_TABLE          ASC,
  TARGET_OWNER          ASC);

CREATE UNIQUE INDEX ASN.IBMSNAP_PRUNCNTLX1
ON ASN.IBMSNAP_PRUNCNTL(
  MAP_ID               ASC);

CREATE INDEX ASN.IBMSNAP_PRUNCNTLX2
ON ASN.IBMSNAP_PRUNCNTL(
  PHYS_CHANGE_OWNER     ASC,
  PHYS_CHANGE_TABLE     ASC);

CREATE INDEX ASN.IBMSNAP_PRUNCNTLX3
ON ASN.IBMSNAP_PRUNCNTL(
  APPLY_QUAL            ASC,
  SET_NAME              ASC,
  TARGET_SERVER         ASC);
```

```
CREATE TABLE ASN.IBMSNAP_PRUNE_SET(
  TARGET_SERVER          CHAR( 18) NOT NULL,
  APPLY_QUAL            CHAR( 18) NOT NULL,
  SET_NAME              CHAR( 18) NOT NULL,
  SYNCHTIME             TIMESTAMP,
  SYNCHPOINT            CHAR( 10) FOR BIT DATA NOT NULL)
IN TSASNCA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_PRUNE_SETX
ON ASN.IBMSNAP_PRUNE_SET(
  TARGET_SERVER         ASC,
  APPLY_QUAL            ASC,
  SET_NAME              ASC);
```

IBMSNAP_PRUNCNTL

IBMSNAP_PRUN_SET

```
CREATE TABLE ASN.IBMSNAP_SIGNAL(
  SIGNAL_TIME           TIMESTAMP NOT NULL WITH DEFAULT,
  SIGNAL_TYPE           VARCHAR( 30) NOT NULL,
  SIGNAL_SUBTYPE        VARCHAR( 30),
  SIGNAL_INPUT_IN       VARCHAR(500),
  SIGNAL_STATE          CHAR( 1) NOT NULL,
  SIGNAL_LSN            CHAR( 10) FOR BIT DATA)
IN TSASNCA
DATA CAPTURE CHANGES;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_SIGNALX
ON ASN.IBMSNAP_SIGNAL( SIGNAL_TIME ASC);
```

IBMSNAP_SIGNAL



参考: キャプチャー・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_RESTART(
  MAX_COMMITSEQ          CHAR( 10) FOR BIT DATA NOT NULL,
  MAX_COMMIT_TIME        TIMESTAMP NOT NULL,
  MIN_INFLIGHTSEQ        CHAR( 10) FOR BIT DATA NOT NULL,
  CURR_COMMIT_TIME        TIMESTAMP NOT NULL,
  CAPTURE_FIRST_SEQ      CHAR( 10) FOR BIT DATA NOT NULL)
IN TSASNCA;
```

IBMSNAP_RESTART

IBMSNAP_CAPPARMS

IBMSNAP_CAPTRACE

```
CREATE TABLE ASN.IBMSNAP_CAPTRACE(
  OPERATION              CHAR( 8) NOT NULL,
  TRACE_TIME             TIMESTAMP NOT NULL,
  DESCRIPTION            VARCHAR(1024) NOT NULL)
IN TSASNCA;
```

```
CREATE INDEX ASN.IBMSNAP_CAPTRACEX
ON ASN.IBMSNAP_CAPTRACE
( TRACE_TIME ASC);
```

IBMSNAP_CAPSCHEMAS

```
CREATE TABLE ASN.IBMSNAP_CAPPARMS(
  RETENTION_LIMIT        INT,
  LAG_LIMIT              INT,
  COMMIT_INTERVAL        INT,
  PRUNE_INTERVAL         INT,
  TRACE_LIMIT            INT,
  MONITOR_LIMIT          INT,
  MONITOR_INTERVAL       INT,
  MEMORY_LIMIT           SMALL INT,
  REMOTE_SRC_SERVER      CHAR( 18),
  AUTOPRUNE              CHAR( 1),
  TERM                   CHAR( 1),
  AUTOSTOP               CHAR( 1),
  LOGREUSE               CHAR( 1),
  LOGSTDOUT              CHAR( 1),
  SLEEP_INTERVAL         SMALL INT,
  CAPTURE_PATH           VARCHAR(1040),
  STARTMODE              VARCHAR( 10))
IN TSASNCA;
```

```
CREATE TABLE ASN.IBMSNAP_CAPENQ(
  LOCK_NAME              CHAR( 9))
IN TSASNCA;
```

IBMSNAP_CAPENQ

```
CREATE TABLE ASN.IBMSNAP_CAPSCHEMAS(
  CAP_SCHEMA_NAME        VARCHAR(30))
IN TSASNCA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_CAPSCHEMAX
ON ASN.IBMSNAP_CAPSCHEMAS(
  CAP_SCHEMA_NAME        ASC);
```



参考: キャプチャー・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_CAPMON(
  MONITOR_TIME           TIMESTAMP NOT NULL,
  RESTART_TIME           TIMESTAMP NOT NULL,
  CURRENT_MEMORY         INT NOT NULL,
  CD_ROWS_INSERTED       INT NOT NULL,
  RECAP_ROWS_SKIPPED     INT NOT NULL,
  TRIGR_ROWS_SKIPPED     INT NOT NULL,
  CHG_ROWS_SKIPPED       INT NOT NULL,
  TRANS_PROCESSED        INT NOT NULL,
  TRANS_SPILLED          INT NOT NULL,
  MAX_TRANS_SIZE         INT NOT NULL,
  LOCKING_RETRIES        INT NOT NULL,
  JRN_LIB                CHAR( 10),
  JRN_NAME               CHAR( 10),
  LOGREADLIMIT           INT NOT NULL,
  CAPTURE_IDLE           INT NOT NULL,
  SYNCHTIME              TIMESTAMP NOT NULL)
IN TSASNCA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_CAPMONX
ON ASN.IBMSNAP_CAPMON(
  MONITOR_TIME          ASC);
```

IBMSNAP_PRUNE_LOCK

IBMSNAP_CAPMON

```
CREATE TABLE ASN.IBMSNAP_UOW(
  IBMSNAP_UOWID          CHAR( 10) FOR BIT DATA NOT NULL,
  IBMSNAP_COMMITSEQ      CHAR( 10) FOR BIT DATA NOT NULL,
  IBMSNAP_LOGMARKER      TIMESTAMP NOT NULL,
  IBMSNAP_AUTHID         VARCHAR(30) NOT NULL,
  IBMSNAP_AUTHID         VARCHAR(30) NOT NULL,
  IBMSNAP_REJ_CODE       CHAR( 1) NOT NULL WITH
  DEFAULT,
  IBMSNAP_APPLY_QUAL      CHAR( 18) NOT NULL WITH
  DEFAULT )
IN TSASNUOW;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_UOWX
ON ASN.IBMSNAP_UOW(
  IBMSNAP_COMMITSEQ      ASC,
  IBMSNAP_LOGMARKER      ASC);
ALTER TABLE ASN.IBMSNAP_UOW VOLATILE;
```

IBMSNAP_UOW

```
CREATE TABLE ASN.IBMSNAP_PRUNE_LOCK(
  DUMMY                  CHAR( 1))
IN TSASNCA;
```



参考: アプライ・コントロール表の定義

```
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
) IN TSASNA;

CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_SETX
ON ASN.IBMSNAP_SUBS_SET(
  APPLY_QUAL          ASC,
  SET_NAME            ASC,
  WHOS_ON_FIRST       ASC);
```

IBMSNAP_SUBS_SET

```
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)
) IN TSASNA;

CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_MEMBRX
ON 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);
```

IBMSNAP_SUBS_MEMBR

DB2 Universal Database

参考: アプライ・コントロール表の定義

```
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
) IN TSASNA;

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);
```

IBMSNAP_SUBS_COLS

```
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)
) IN TSASNA;

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);
```

IBMSNAP_SUBS_STMTS

```
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
) IN TSASNA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_SUBS_EVENTX
ON ASN.IBMSNAP_SUBS_EVENT(
  EVENT_NAME          ASC,
  EVENT_TIME          ASC);
```

IBMSNAP_SUBS_EVENT

DB2 Universal Database

参考: アプライ・コントロール表の定義

```
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),
```

IBMSNAP_APPLYTRAIL

つづき

```
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))
IN TSASNA;
```

```
CREATE INDEX ASN.IBMSNAP_APPLYTRAILX
ON ASN.IBMSNAP_APPLYTRAIL(
  LASTRUN             DESC,
  APPLY_QUAL          ASC);
```

```
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)
IN TSASNA;
```

```
CREATE INDEX ASN.IBMSNAP_APPLYTRACEX
ON ASN.IBMSNAP_APPLYTRACE(
  APPLY_QUAL          ASC,
  TRACE_TIME          ASC);
```

IBMSNAP_APPLYTRACE



参考: アプライ・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_APPPARMS(
  APPLY_QUAL          CHAR( 18) NOT NULL,
  APPLY_PATH          VARCHAR(1040),
  COPYONCE            CHAR( 1) WITH DEFAULT 'N',
  DELAY               INT WITH DEFAULT 6,
  ERRWAIT             INT WITH DEFAULT 300,
  INAMSG              CHAR( 1) WITH DEFAULT 'Y',
  LOADXIT             CHAR( 1) WITH DEFAULT 'N',
  LOGREUSE            CHAR( 1) WITH DEFAULT 'N',
  LOGSTDOUT           CHAR( 1) WITH DEFAULT 'N',
  NOTIFY              CHAR( 1) WITH DEFAULT 'N',
  OPT4ONE             CHAR( 1) WITH DEFAULT 'N',
  SLEEP              CHAR( 1) WITH DEFAULT 'Y',
  SQLERRCONTINUE      CHAR( 1) WITH DEFAULT 'N',
  SPILLFILE           VARCHAR( 10) WITH DEFAULT 'DISK',
  TERM                CHAR( 1) WITH DEFAULT 'Y',
  TRLREUSE            CHAR( 1) WITH DEFAULT 'N')
IN TSASNA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_APPPARMSX
ON ASN.IBMSNAP_APPPARMS(
  APPLY_QUAL          ASC);
```

IBMSNAP_APPPARMS

IBMSNAP_APPLYENQ

IBMSNAP_COMPENSATE

```
CREATE TABLE ASN.IBMSNAP_COMPENSATE(
  APPLY_QUAL          CHAR( 18) NOT NULL,
  MEMBER              SMALLINT NOT NULL,
  INTENTSEQ           CHAR( 10) FOR BIT DATA NOT NULL,
  OPERATION            CHAR( 1) NOT NULL)
IN TSASNA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_COMPENSATX
ON ASN.IBMSNAP_COMPENSATE(
  APPLY_QUAL          ASC,
  MEMBER              ASC);
```

```
CREATE TABLE ASN.IBMSNAP_APPENQ(
  APPLY_QUAL          CHAR( 18))
IN TSASNA;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_APPENQX
ON ASN.IBMSNAP_APPENQ(
  APPLY_QUAL          ASC);
```



参考：モニター・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_CONTACTS(
CONTACT_NAME          VARCHAR(127) NOT NULL,
EMAIL_ADDRESS         VARCHAR(128) NOT NULL,
ADDRESS_TYPE         CHAR(1) NOT NULL,
DELEGATE             VARCHAR(127),
DELEGATE_START        DATE,
DELEGATE_END          DATE,
DESCRIPTION           VARCHAR(1024))
IN REPLMONTS1;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_CONTACTSX
ON ASN.IBMSNAP_CONTACTS(
CONTACT_NAME          ASC);
```

IBMSNAP_CONTACTS

```
CREATE TABLE ASN.IBMSNAP_ALERTS(
MONITOR_QUAL          CHAR(18) NOT NULL,
ALERT_TIME            TIMESTAMP NOT NULL,
COMPONENT             CHAR( 1) NOT NULL,
SERVER_NAME           CHAR(18) NOT NULL,
SERVER_ALIAS          CHAR( 8),
SCHEMA_OR_QUAL        VARCHAR(30) NOT NULL,
SET_NAME              CHAR(18) NOT NULL
WITH DEFAULT ' ',
CONDITION_NAME        CHAR(18) NOT NULL,
OCCURRED_TIME         TIMESTAMP NOT NULL,
ALERT_COUNTER         SMALLINT NOT NULL,
ALERT_CODE            CHAR( 10) NOT NULL,
RETURN_CODE           INT NOT NULL,
NOTIFICATION_SENT     CHAR(1) NOT NULL,
ALERT_MESSAGE         VARCHAR(1024) NOT NULL)
IN REPLMONTS2;
```

```
CREATE INDEX ASN.IBMSNAP_ALERTX
ON ASN.IBMSNAP_ALERTS(
MONITOR_QUAL          ASC,
COMPONENT             ASC,
SERVER_NAME           ASC,
SCHEMA_OR_QUAL        ASC,
SET_NAME              ASC,
CONDITION_NAME        ASC,
ALERT_CODE            ASC);
```

```
CREATE TABLE ASN.IBMSNAP_MONTRACE(
MONITOR_QUAL          CHAR(18) NOT NULL,
TRACE_TIME            TIMESTAMP NOT NULL,
OPERATION             CHAR( 8) NOT NULL,
DESCRIPTION           VARCHAR(1024) NOT NULL)
IN REPLMONTS3;
```

```
CREATE INDEX ASN.IBMSNAP_MONTRACEX
ON ASN.IBMSNAP_MONTRACE(
MONITOR_QUAL          ASC,
TRACE_TIME            ASC);
```

IBMSNAP_MONTRACE

IBMSNAP_ALERTS



参考：モニター・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_GROUPS(
GROUP_NAME            VARCHAR(127) NOT NULL,
DESCRIPTION           VARCHAR(1024))
IN REPLMONTS1;
```

IBMSNAP_GROUPS

```
CREATE TABLE ASN.IBMSNAP_CONTACTGRP(
GROUP_NAME            VARCHAR(127) NOT NULL,
CONTACT_NAME          VARCHAR(127) NOT NULL)
IN REPLMONTS1;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_CONTACTGPX
ON ASN.IBMSNAP_CONTACTGRP(
GROUP_NAME            ASC,
CONTACT_NAME          ASC);
```

IBMSNAP_CONTACTGRP

```
CREATE TABLE ASN.IBMSNAP_CONDITIONS(
MONITOR_QUAL          CHAR(18) NOT NULL,
SERVER_NAME           CHAR(18) NOT NULL,
COMPONENT             CHAR( 1) NOT NULL,
SCHEMA_OR_QUAL        VARCHAR(30) NOT NULL,
SET_NAME              CHAR(18) NOT NULL
WITH DEFAULT ' ',
SERVER_ALIAS          CHAR( 8),
ENABLED               CHAR( 1) NOT NULL,
CONDITION_NAME        CHAR(18) NOT NULL,
PARM_INT              INT,
PARM_CHAR             VARCHAR(128),
CONTACT_TYPE          CHAR( 1) NOT NULL,
CONTACT              VARCHAR(127) NOT NULL)
IN REPLMONTS1;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_MONCONDX
ON ASN.IBMSNAP_CONDITIONS(
MONITOR_QUAL          ASC,
SERVER_NAME           ASC,
COMPONENT             ASC,
SCHEMA_OR_QUAL        ASC,
SET_NAME              ASC,
CONDITION_NAME        ASC);
```

IBMSNAP_MONSERVERS

```
CREATE TABLE ASN.IBMSNAP_MONSERVERS(
MONITOR_QUAL          CHAR(18) NOT NULL,
SERVER_NAME           CHAR(18) NOT NULL,
SERVER_ALIAS          CHAR( 8),
LAST_MONITOR_TIME     TIMESTAMP NOT NULL,
START_MONITOR_TIME    TIMESTAMP,
END_MONITOR_TIME      TIMESTAMP,
LASTRUN               TIMESTAMP NOT NULL,
LASTSUCCESS           TIMESTAMP,
STATUS                SMALLINT NOT NULL)
IN REPLMONTS1;
```

```
CREATE UNIQUE INDEX ASN.IBMSNAP_MONSERVERX
ON ASN.IBMSNAP_MONSERVERS(
MONITOR_QUAL          ASC,
SERVER_NAME           ASC);
```

IBMSNAP_MONENQ

IBMSNAP_CONDITIONS



参考：モニター・コントロール表の定義

```
CREATE TABLE ASN.IBMSNAP_MONENQ(  
  MONITOR_QUAL CHAR( 18) NOT NULL)  
  IN REPLMONTS1;
```

IBMSNAP_MONENQ

```
CREATE TABLE ASN.IBMSNAP_MONTRAIL(  
  MONITOR_QUAL CHAR(18) NOT NULL,  
  SERVER_NAME CHAR(18) NOT NULL,  
  SERVER_ALIAS CHAR( 8),  
  STATUS SMALLINT NOT NULL,  
  LASTRUN TIMESTAMP NOT NULL,  
  LASTSUCCESS TIMESTAMP,  
  ENDTIME TIMESTAMP NOT NULL WITH DEFAULT ,  
  LAST_MONITOR_TIME TIMESTAMP NOT NULL,  
  START_MONITOR_TIME TIMESTAMP,  
  END_MONITOR_TIME TIMESTAMP,  
  SQLCODE INT,  
  SQLSTATE CHAR(5),  
  NUM_ALERTS INT NOT NULL,  
  NUM_NOTIFICATIONS INT NOT NULL)  
  IN REPLMONTS3;
```

IBMSNAP_MONTRAIL



DB2 Universal Database

blank page



DB2 Universal Database

ソース表

- 変更データの収集を行なうためには、表にDATA CAPTURE CHANGESの属性が必要
 - 既存の表をソース表とする場合は、ALTER SQLステートメント文でDATA CAPTURE CHANGESの属性を追加
 - DATA CAPTURE CHANGESの属性を追加すると、ログに書き出される情報が変わる
 - Partial Before/Partial After => Full Before/Partial After
- キャプチャーは直接LOGバッファやLOGファイルを読むのではなく、DB2に対してLOG読み込み用のAPIを発行して、DB2から更新情報を受け取る

更新処理

update from tabA set col2=100, col5='XXX'

更新前

Col1	Col2	Col3	Col4	Col5	Col6
1	2	AAA	BBB	CCC	DDD

更新後

Col1	Col2	Col3	Col4	Col5	Col6
1	100	AAA	BBB	XXX	DDD

ログに書かれる情報

Data Capture None の場合

更新前

Col2	Col3	Col4	Col5
2	AAA	BBB	CCC

+

更新後

Col2	Col3	Col4	Col5
100	AAA	BBB	XXX

Data Capture Changes の場合

Col1	Col2	Col3	Col4	Col5	Col6
1	2	AAA	BBB	CCC	DDD

+

Col2	Col3	Col4	Col5
100	AAA	BBB	XXX

DB2 Universal Database

解説:

- レプリケーションのソース表には、変更データの収集を行なうために、表にDATA CAPTURE CHANGESの属性が必要です。
- DATA CAPTURE CHANGESを指定すると、ログには更新された行の全列の変更前イメージと更新対象列に挟まれた部分の列の変更後イメージがとられます。
- DATA CAPTURE CHANGESの指定がない場合は、ログには更新対象列に挟まれた部分の列の変更前/後のイメージがとられます。
- 既存の表をソース表とする場合は、ALTER SQLステートメント文でDATA CAPTURE CHANGESの属性を追加できます。
- レプリケーション・センター(V8)を使用してソース表の登録をする場合は、自動的にALTERステートメントを生成します。

変更データ (Change Data : CD)表

- 変更データの収集を行なうためには、各ソース表に対して変更データ表 (Change Data : CD)表が必要
 - レプリケーション対象列に3つの制御列が追加されたもの
 - IBMSNAP_COMMITSEQ
 - IBMSNAP_INTENTSEQ
 - IBMSNAP_OPERATION
 - CD表には、更新後値のみ、または更新前値と更新後値の両方を含めることができる

CD表:更新後イメージのみ

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	Col1	Col2	Col3
x'43464391000000010000'	x'00000000000003FBBA214'	U	1	100	AAA

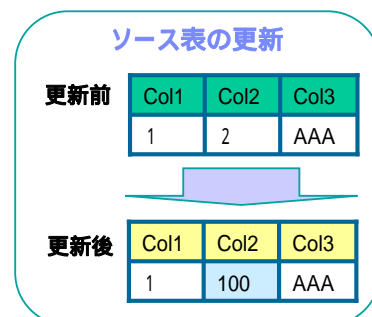
更新後

CD表:更新前・後イメージを含む

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	XCol1	XCol2	XCol3	Col1	Col2	Col3
x'43464391000000010000'	x'00000000000003FBBA214'	U	1	2	AAA	1	100	AAA

更新前

更新後



DB2 Universal Database

解説:

- 変更データの収集を行なうためには、各ソース表に対して変更データ表 (CD表:ChangeData表) が必要です。
 - V7までは各ソース表に対して1つ
 - V8からはスキーマ名ごとにCD表の作成が可能
- キャプチャー・プログラムがアクティブ DB2 データベース・ログまたはジャーナルを読み取って、登録済み表への 変更を検索する場合、CD 表にコミット済みトランザクションを記録します。
- アプライ・プログラムは、CD 表にコミットされた行を読み取り、登録済み表にマップされたターゲット表にそれらを複製します。
- CD 表に作成するように設定した変更後イメージ列および変更前イメージ列に加え、常に存在する列が他に 3つあります。
 - IBMSNAP_COMMITSEQ
 - キャプチャーされたコミット・ステートメントのログ・レコード・シーケンス
 - IBMSNAP_INTENTSEQ
 - 1つの作業単位内の更新トランザクションの実行順序を識別するためのID
 - IBMSNAP_OPERATION
 - ソース表に対する操作 (U,I,D)
- IBMSNAP_COMMITSEQ 列には、キャプチャーされたコミット・ステートメントのログ・レコード・シーケンスが含まれます。この列は、レプリケーション・ソースで発生したオリジナルのトランザクションによる挿入、更新、および削除をグループ化します



DB2 Universal Database

CD表の容量見積もり

- CD表のサイズはレプリケーションの要件によって大きく変化
- 行長と行数
 - 行長: CD表に含めた列の長さの合計+制御列(21バイト)
 - 行数: 最低数=キャプチャーによるブルーニング間隔の間に更新されるソース表の行数

■ 見積もりのガイド

推奨のCD表サイズ =

(21 bytes) + (全ての登録済み列の長さの合計)

× (ブルーニング間隔でのソース表のinserts, updates, deletesの行数)

× 余裕率

- 余裕率とは、以下のような状況の発生を想定し限度を決定
 - アプライが停止していたためにCD表に滞留
 - キャプチャー停止後の再開で、一気にCD表に書かれる
 - 1つのソース表から複数へのサイトへレプリケーションしているが、あるサイトのレプリケーションの進度が遅い見積もりのガイド
 - UPDATE更新をDELETEとINSERTに変更する形式のレプリケーションを行う場合は、UPDATE行数に関しては2倍



DB2 Universal Database

解説:

- CD 表および一部のレプリケーション・コントロール表 (IBMSNAP UOW、IBMSNAP CAPTRACE、IBMSNAP APPLYTRACE、IBMSNAP APPLYTRAIL、IBMSNAP CAPMON、IBMSNAP ALERTS) も、DB2 ソース・データベース用として必要なディスク・スペースに影響を与えます。レプリケーション環境のセットアップの仕方によっては、これらの表が非常に大きくなる可能性があります。上記以外のレプリケーション・コントロール表の場合は、必要なスペースは通常少なく、かつ静的です。
- CD 表のサイズは、ソース表に変更が加えられるたびに大きくなり、やがてキャプチャー・プログラムによって整理されます。CD 表に必要なスペースを見積もるには、データをどのくらい保持してから整理するのかについて、その期間を最初に決定した後、キャプチャー・プログラムがこれらの表を自動的に整理する頻度、またはユーザー自身がコマンドを使用してこれらの表を整理する頻度を指定します。
- レプリケーション済みデータのバイト数を計算する場合は、キャプチャー・プログラムが CD 表に追加したオーバーヘッド・データの分として 21 バイトをそれぞれの行ごとに組み込む必要があります。キャプチャー・プログラムが CD 表にデータをキャプチャーし続けられる期間 (たとえばネットワークが停止するなどしてデータが適用不能なときも含む) を決定してください。上記の偶発的な障害が発生した期間内にソース表で通常キャプチャーされる可能性のある挿入、更新、および削除の数を見積もってください。



DB2 Universal Database

UOW表の容量見積もり

- UOW表のサイズはレプリケーションの要件によって大きく変化
- 行長と行数
 - 行長: 109バイト
 - 行数: 最低数=キャプチャーによるブルーニング間隔の間の、全ての登録済みソース表に対する更新コミット・トランザクション数

■ 見積もりのガイド

推奨のUOW表サイズ =
(109バイト)

× (ブルーニング間隔での全ての登録済みソース表に対する更新コミット・トランザクション数)

× 余裕率

- 余裕率とは、以下のような状況の発生を想定し限度を決定
 - あるアプリが停止していたためにブルーニングができない
 - キャプチャー停止後の再開で、一気に更新情報が収集される
- 当初は必要なスペースを多めに見積もっておき、実際に使用されるスペースをモニターして判断



解説:

- UOW 表は、特定のコミット・インターバルの間にキャプチャー・プログラムによって挿入される行数、および整理される行数に基づいて、拡大および縮小します。アプリケーション・トランザクションが COMMIT を発行するたび、およびトランザクションが登録済みのレプリケーション・ソース表に対して INSERT、DELETE、または UPDATE 操作を実行するたびに、行が 1 つ UOW 表に挿入されます。当初はこの表に必要なスペースを多めに見積もっておき、実際に使用されるスペースをモニターして、回収できるスペースがあるかどうかを判別してください。



ターゲット表タイプ

■ ターゲット表のタイプ

- ユーザー・コピー表
- ポイント・イン・タイム表
- 整合変更データ(CCD)表
- 集約表
- レプリカ表



解説:

- ターゲットサーバー側で作成、使用できる表は、利用目的にあわせていくつかのタイプがあります。
 - ユーザーコピー表
 - ポイント・イン・タイム表
 - Point-in-Time表(時刻表)
 - 整合変更データ表
 - CCD (Consistent Change Data)表(ステージング表)
 - 集約表
 - レプリカ表
- いずれのタイプもSQLによって更新は可能ですが、レプリカ表以外の表は、この表を更新した後アプライ・プログラムによってフル・リフレッシュが実行されると、ユーザーの更新情報が失われる危険性があります。



ターゲット表タイプ - ユーザーコピー表

■ ごく一般的なタイプ

- CD表、全件フルリフレッシュのみの場合はソース表の全てまたは一部の列のみから構成される表
- SQLレプリケーションが追加する制御列は含まない
- IBMSNAP_SUBS_MEMBR表のTARGET_STRUCTURE列には8が入る

ソース表

Col1	Col2	Col3	Col4

CD表

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	Col1	Col2	Col3	Col4

ユーザーコピー表

CD表またはソース表の
全ての列からなる

Col1	Col2	Col3	Col4

CD表またはソース表の
一部の列からなる

Col1	Col2	Col4



解説:

- ユーザー・コピー表は、ごく一般的なタイプで、ソース表の列のコピーを含むターゲット表、即ちソース表とほぼ同じ構造の表です。
- ターゲット表の中の列名はソース表の中の列名と一致している必要はありませんが、データ・タイプは一致している必要があります。
- ターゲット表では、ソース表の行または列のサブセットを使用できますが、追加列を含めることはできません。
 - ただし、SQL 式から派生したユーザー定義の列やソース・データ・タイプを別のターゲット・データ・タイプに変換するために、SQL 関数で算出された列を使用することができます。
- 既存の視点をユーザーコピーのターゲット表として定義することは可能ですが、使用できる視点には制約があります。
 - 1表から列、行を選択した単純な視点であり、視点に含まれない元表の列はNULL可能でなければならない。
 - 複数の同じ構造の表のUNION ALL視点であるが、情報制約付きのUNION ALL視点である。



ターゲット表タイプ - ポイント・イン・タイム表

- ユーザーコピー表にタイム・スタンプ列(IBMSNAP_LOGMARKER)を追加したもの
 - タイム・スタンプ列は、初期値はNULL(空白値)で、変更が複製されると更新がなされたソース側の時刻を示す値に更新される
 - IBMSNAP_SUBS_MEMBR表のTARGET_STRUCTURE列には4が入る
- 変更の時刻を記録したい場合に便利

ソース表

Col1	Col2	Col3	Col4

CD表

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	Col1	Col2	Col3	Col4

ポイント・イン・タイム表

IBMSNAP_LOGMARKER	Col1	Col2	Col3	Col4



解説:

- タイム・スタンプ列のあるレプリケーション・ソースのコピー表です。
- タイム・スタンプ列は元は NULLです。変更が複製されると値が追加され、更新が行われた時刻を示します。
- ソース表への変更が行われた時刻のトラックを保持する場合は、このタイプのターゲット表を使用してください。



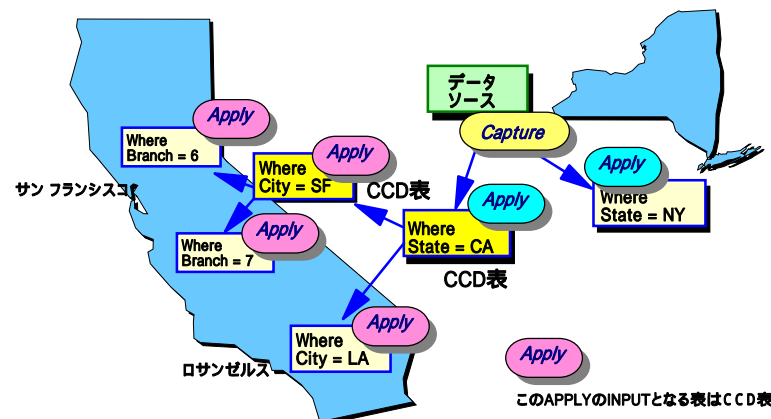
ターゲット表タイプ - 整合変更データ(CCD)表

■ ステージング表の1つのタイプ

- この表を基に、アプライがさらに別のターゲット表に変更データを複製することが可能なタイプ

■ 利用例

- 扇状型(Fan-Out)レプリケーション
 - 3階層目以降はCaptureがなくともレプリケーション可能



解説:

■ 整合変更データ (CCD::Consistent Change Data) 表

- CCD表はCD表と類似していますが、キャプチャーが書き出す先ではなく、アプライが更新するターゲット表の1つの種類です。
- ソース表へのコミット済みの変更を保管し、変更後イメージ列と同様に変更前イメージ列を含むこともできます。
- CCD 表の形式は公開されているので、キャプチャー・プログラムが提供されていない環境からのレプリケーションを行う場合などでも、CCD表の形式に従って更新情報を書き出せば、アプライやユーザープログラムによって、レプリケーションを行ったり、情報の活用が可能です。
 - 非DB2データベースからのトリガーベースのレプリケーションでも、内部的に使用されています。



ターゲット表タイプ - 整合変更データ(CCD)表

■ CCD表の構造

- ユーザーコピーに4列の制御列が追加された構造
- CD表にUOW表のIBMSNAP_LOGMARKER列を追加したイメージ
- オプションでUOW表の他の情報を含む列を追加することも可能
- IBMSNAP_SUBS_MEMBR表のTARGET_STRUCTURE列には3が入る

■ CCDの種類

- CCD表の属性
 - Condensed / Non-Condensed (圧縮 / 非圧縮)
 - Complete / Non-Complete (完全 / 不完全)
- 上記の2種・2タイプの組み合わせで、合計4種類のタイプがある

■ CCDの場所

- 外部CCD
 - Fan-Outレプリケーション一時などに使用される一般的なCCD表
- 内部CCD
 - ソース表と同じデータベース内に作成した、不完全タイプの特殊なCCD表

CCD表 - CD表の代わりに使用される

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	IBMSNAP_LOGMARKER	Col1	Col2	Col3	Col4



DB2 Universal Database

解説:

■ CCD表は必ず含めなければならない4つの制御列があります。

- IBMSNAP_INTENTSEQ : 1つの作業単位内の更新トランザクションの実行順序を識別するためのID
- IBMSNAP_OPERATION : ソース表に対する操作(U,I,D)
- IBMSNAP_COMMITSEQ : キャプチャーされたコミット・ステートメントのログ・レコード・シーケンス
- IBMSNAP_LOGMARKER : キャプチャー・コントロール・サーバーでのコミット時間

■ また不完全外部CCD表の場合は、オプションとしてUOW表に含まれる以下の列もユーザー列として含むこともできます。

- IBMSNAP_APPLY_QUAL : このCCD表を処理する適用修飾子
- IBMSNAP_AUTHID : このトランザクションに関連する許可ID
- IBMSNAP_AUTHTKN : このトランザクションに関連する許可トークン
- IBMSNAP_REJ_CODE : レプリカ設定時のコンフリクトの発生状況を示すコード
- IBMSNAP_UOWID : 作業単位ID

斜体の列はUOW表にある列

■ キャプチャー・プログラムおよびアプライ・プログラムのどちらも、自動的に CCD 表を整理(ブルーニング)することはできません。

■ CCD 表の属性

- CCD 表は完全または不完全、また圧縮または圧縮されていないものにすることができます。

■ 外部CCD

- 次の目的で 사용할 수 있는 타겟 테이블의 타입입니다.
 - 単一層でのレプリケーション環境で、データを複製する
 - multi-tier でのレプリケーション環境で、データを複製する
 - キャプチャー・プログラム以外に、変更キャプチャーのメカニズムからのデータを保管する
 - 履歴および監査情報を収集する

■ 内部CCD

- 레플리케이션·소스의 CD 表의 대체로 존재하는 타겟 테이블의 타입입니다. 타겟 테이블로써의 내부 CCD 表と一緒にサブ스크립ション·세트·멤버를 이틀만 작성해버리면, 同一의 레플리케이션·소스에 구독하는 다른 모든 타겟 테이블은, 레플리케이션·소스의 CD 表의 대체로 내부 CCD 表에서 변경 데이터를 받습니다. 즉, 아플라이·프로그램은 내부 CCD 表에서 변경 데이터를 읽어들이고, 그 데이터를 타겟 테이블에 복제합니다.



DB2 Universal Database

ターゲット表タイプ - 整合変更データ(CCD)表

■ 圧縮 / 非圧縮

- 圧縮 : 1つのキーに対して常に最新の1行の状態のみ持つ
- 非圧縮 : 1つのキーに対して複数行存在し、変更履歴が分かる(変更データは常にINSERTされる)

■ 例: ソース表に対する更新

- INSERT INTO T1 (C1,C2,C3) VALUES (1,'AAA','BBB');
- INSERT INTO T1 (C1,C2,C3) VALUES (2,'CCC','DDD');
- UPDATE T1 SET C2='XXX' WHERE C1=1
- DELETE FROM T1 WHERE C1=2

非圧縮CCD表

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	IBMSNAP_LOGMARKER	Col1	Col2	Col3
		I		1	AAA	BBB
		I		2	CCC	DDD
		U		1	XXX	BBB
		D		2	CCC	DDD

圧縮CCD表

IBMSNAP_COMMITSEQ	IBMSNAP_INTENTSEQ	IBMSNAP_OPERATION	IBMSNAP_LOGMARKER	Col1	Col2	Col3
		U		1	XXX	BBB
		D		2	CCC	DDD



解説:

■ 圧縮

- 1つのキーに対して常に最新の1行の状態のみ持つものです。
- 同じキーに対して複数回の更新があっても、最終的なオペレーションの情報のみをもつので、CCD表の大きさの節約になります。

■ 非圧縮

- 同じキーに対して複数回の更新があった場合は、その全てのオペレーションの情報をもつので、変更履歴が分かります。
- したがって1つのキーに対する行が複数行存在する可能性があります。
- また変更データは常にINSERTされます



ターゲット表タイプ - 整合変更データ(CCD)表

■ 完全 / 不完全

- 完全 : ソース表と同じ全行のデータを持つ
- 不完全 : 変更データのみ持つ

■ 例: ソース表に対する更新

- INSERT INTO T1 (C1,C2,C3) VALUES (1,'AAA','BBB');
- INSERT INTO T1 (C1,C2,C3) VALUES (2,'CCC','DDD');
- UPDATE T1 SET C2='XXX' WHERE C1=1
- DELETE FROM T1 WHERE C1=2

ソース表初期状態

Col1	Col2	Col3
90	YYY	ZZZ
100	KKK	LLL

不完全
(圧縮)
CCD表

	IBMSNAP_ OPERATION		Col1	Col2	Col3
	U		1	XXX	BBB
	D		2	CCC	DDD

	IBMSNAP_ OPERATION		Col1	Col2	Col3
	I		1	AAA	BBB
	I		2	CCC	DDD
	U		1	XXX	BBB
	D		2	CCC	DDD

不完全
(非圧縮)
CCD表

完全
(圧縮)
CCD表

	IBMSNAP_ OPERATION		Col1	Col2	Col3
	I		90	YYY	ZZZ
	I		100	KKK	LLL
	U		1	XXX	BBB
	D		2	CCC	DDD

	IBMSNAP_ OPERATION		Col1	Col2	Col3
	I		90	YYY	ZZZ
	I		100	KKK	LLL
	I		1	AAA	BBB
	I		2	CCC	DDD
	U		1	XXX	BBB
	D		2	CCC	DDD

完全
(非圧縮)
CCD表



DB2 Universal Database

解説:

■ 完全

- ソース表またはソース・ビューの全ての行を入れたCCD表です。

■ 不完全

- ソース表またはソース・ビューに加えられた変更データだけを入れたCCD表です

■ 以下の表では、CCD 表の構成を決定する属性についてサマリーしています。

完全 不完全	圧縮 非圧縮	CCD 表を作成するときの行数	変更データの記録方法
完全	圧縮	レプリケーション・ソースと同じ行から開始	ユニーク索引あり。CCD 表にレプリケーション・ソースと同じデータをもたせるため、既存の行を更新する。
完全	非圧縮	レプリケーション・ソースと同じ行から開始	ソースの登録時に指定した、行のキャプチャー規則に従い、登録済み列または登録抹消済み列への変更が発生するたびに 1 つの行を CCD 表に追加する。
不完全	圧縮	不特定の行から開始	ユニーク索引があるため、ユニーク・キーの値ごとに 1 つだけ挿入される。ソースの登録時に指定した、行のキャプチャー規則に従い、登録済み列または登録抹消済み列のソース行ごとに最新の変更だけが CCD 表に含まれる。
不完全	非圧縮	不特定の行から開始	ソースの登録時に指定した、行のキャプチャー規則に従い、登録済み列または登録抹消済み列への変更が発生するたびに 1 つの行を CCD 表に追加する。



DB2 Universal Database

ターゲット表タイプ - 集約表

- SQL 列関数 (SUM や AVG など) を使用し、ソース表の内容全体またはソース表のデータに加えられた最近の変更に関する要約データを計算した結果が格納される表
 - 集約表は常に行は追加
- 集約表の種類
 - 基礎集約表
 - ソース表のデータの全体に対する要約データを計算
 - IBMSNAP_SUBS_MEMBR表のTARGET_STRUCTURE列には5が入る
 - 変更集約表
 - CD表にある変更データに対する要約データを計算
 - IBMSNAP_SUBS_MEMBR表のTARGET_STRUCTURE列には6が入る

ソース表

Col1	Col2	Col3	Col4

CD表

			Col1	Col2	Col3	Col4

基礎集約表

			SUM(Col4)

変更集約表

			SUM(Col4)



解説:

- 集約表はSQL 列関数 (SUM や AVG など) を使用して、ソース表の内容全体のサマリーや、ソース表データに新しく加えられた変更内容の集約を計算します
- 常に行は追加され、表を集約していきます。
- 表の集約には基礎集約表および変更集約表の 2 種類があります。
- 基礎集約表
 - ソース表の内容をサマリーします。
 - 基礎集約表を使用して、定型ベースでのソース表の状態をトラッキングします。
 - たとえば、毎月の顧客数の平均値を求めるとします。顧客ごとの行がソース表にある場合、ソース表にある行の平均値を月ベースで計算し、その結果を基礎集約表に保管します。
 - 変更情報をトラッキングしません。たとえば、1 月は平均 500 人の顧客数で 2 月も平均 500 人の顧客数でしたが、2 月は既存の 2 人の顧客を失い、新規の顧客を 2 人獲得したとします。基礎集約表からは、どちらの月も顧客数は同じ平均値であることがわかりますが、2 月に発生した変更内容はわかりません。変更情報をトラッキングする場合は、変更集約表を使用してください。
- 変更集約表
 - ソース表の内容ではなく CD 表の内容を処理します。
 - 変更集約表を使用して、時間を通して行われた変更 (UPDATE、INSERT および DELETE 操作) のトラッキングができます。
 - たとえば、月ごとに獲得した新規の顧客数 (INSERT) および失った既存の顧客数 (DELETE) を知りたい場合などです。ソース表にある行に対する変更を月ベースでカウントし、その数値を変更集約表に保管します。



ターゲット表タイプ - レプリカ表

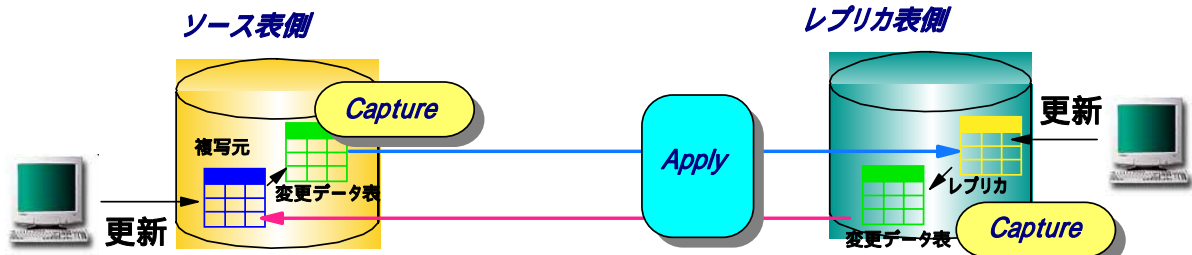
- その表に対する変更もソース表に戻す事が可能
 - ユーザーのアプリケーションから直接更新できる唯一のターゲット表のタイプ
 - IBMSNAP_SUBS_MEMBR表のTARGET_STRUCTURE列には7が入る
- レプリカは DB2データベースでのみサポート
 - レプリカ表を使用する場合は、レプリカ側のデータベース(ターゲット・サーバー)でもキャプチャーを稼働させる必要がある
 - アプライはレプリカ側またはソース側どちらかで1つ

ソース表

Col1	Col2	Col3	Col4

レプリカ表

Col1	Col2	Col3	Col4



解説:

- レプリカは表、ターゲットおよびソースの両方の役目を果たす Update-anywhere レプリケーションで使用する表であり、ユーザーのアプリケーションから直接更新し、その更新をソース表にも反映できる唯一のターゲット表のタイプです。
- レプリカへのソースはマスター表と呼ばれますが、レプリカから複製された変更で更新することができます。
- キャプチャー・プログラムは、レプリカのあるターゲット・サーバーにも必要です。キャプチャー・プログラムは、同じサーバーを共用するアプリケーションによってレプリカに対して行われた変更をキャプチャーします。そのため各レプリカ表には、専用の CD 表があります。
- マスター表への変更をキャプチャーするキャプチャー・プログラム、およびレプリカへの変更をキャプチャーするキャプチャー・プログラムとがありますが、アプライ・プログラムは、どちらか一方だけです。
 - 通常はターゲット側
- キャプチャー・コントロール・サーバーのマスター表の CD 表からデータをプルし、プログラムを実行しているターゲット・サーバーのレプリカの CD 表から、データをキャプチャー・コントロール・サーバーへプッシュします。
- 注: レプリカには CD 表が必要であるため、非 DB2 リレーショナル・データベースにレプリカを作成することはできません。



キャプチャー・コントロール・サーバーのログ

■キャプチャーの稼働のログへの影響

- ソース表へのDATA CAPTURE CHANGES属性の追加
 - 変更前イメージは全列分
 - 1列のみ、あるいは行の一部にまとまった列の更新を行うアプリケーションが多い場合は、ログに書き出される情報量が非常に多くなる場合もある
- CD表、UOW表および制御表への情報の挿入、
- プルーニングによるCD表、UOW表、その他の制御表の行の削除



- 一般的には現行ログ量の3倍に増加



解説:

- 一般に、レプリケーションに関係するすべての表のために、現行ログの3倍の量のログが追加が必要となります。基本的に、ソース表用、CD表用、およびレプリケーション・コントロール表用のログ・スペースが必要です。
- Linux、UNIX、Windows、およびz/OS:
 - DB2はUPDATEステートメントごとに完全な行イメージをログに記録します。これが発生するのは、DATA CAPTURE CHANGESキーワードを使用して表を作成(または変更)してからでなければ表をレプリケーションできないからです。
 - ログへの追加量が最大になるレプリケーション要件の1つとして、変更前イメージと変更後イメージのキャプチャー(Update-anywhereレプリケーションシナリオにおけるレプリケーション・ターゲット表の例が当てはまる)が挙げられます。ログの量を少なくする方法の1つとして、レプリケーション・ソースに対して定義されている列数を少なくすることができます。たとえば、必要でない場合は変更前イメージをキャプチャーしないでください。



ターゲット・サーバーのログ

■ アプライによる更新のログへの影響

- アプライのコミット・モードによって、必要になるログのサイズが異なる

■ アプライのコミット・モード

- 表モード
 - フェッチしたデータがすべて適用された後で単一コミットを発行
 - アプライ・プログラムが 1 回のインターバルで処理する最大データ量を見積もり、そのデータ量を格納できるようにログ・スペースを調整
- トランザクション・モード
 - それぞれの更新をソース・トランザクションの順序どおりにすべてターゲット表にコピー
 - ソース表のコミットの回数をカウントアップし、指定されたコミット・カウントに達する度にコミットを発行
 - コミットのインターバルの設定
 - サブスクリプション・セット・オプション (`commit_count (x)`) の値



DB2 Universal Database

解説:

- ソース・データベースのロギングに加えて、行が適用されるターゲット・データベースでもロギングされます。ログへの影響は、アプライ・プログラム用に選択したコミット・モードによって異なります。
- 表モード
 - 表モード処理では、アプライ・プログラムはフェッチしたデータがすべて適用された後で単一コミットを発行します。アプライ・プログラムは一時チェックポイントを発行しません。このケースでは、アプライ・プログラムが 1 回のインターバルで処理する最大データ量を見積もり、そのデータ量を格納できるようにログ・スペースを調整する必要があります。
- トランザクション・モード
 - トランザクション・モード処理では、アプライ・プログラムはそれぞれの更新をソース・トランザクションの順序どおりにすべてターゲット表にコピーし、それらの変更を 1 回のインターバルでトランザクション境界に達したときにまとめてコミットします。一時コミットのインターバルを設定するには、サブスクリプション・セット・オプション (`commit_count (x)`) の値 `x` を設定します。すべての応答セットを取り出すと、アプライ・プログラムはスビル・ファイルの内容をコミット・シーケンスの順序に従って適用します。このタイプの処理では、すべてのスビル・ファイルを同時に開いて処理できます。たとえば、コミット・カウントを 1 に設定と、アプライ・プログラムは 1 つのトランザクションが終わるたびにコミットします。コミット・カウントを 2 に設定した場合は、アプライ・プログラムは 2 つのトランザクションのセットが 1 つ終わるたびにコミットします。
- その他、MAX SYNCH MINUTESを指定することにより、ミニ・サイクルでのレプリケーションが可能ですが、MAX SYNCH MINUTESはソース側のトランザクションを指定された時間で区切るため、トランザクションの発生が一樣でない、ミニサイクルごとの更新量が不定であり、必要となるログのサイズもばらつきが出る可能性があります。



DB2 Universal Database

スピル・ファイル



DB2 Universal Database

blank page



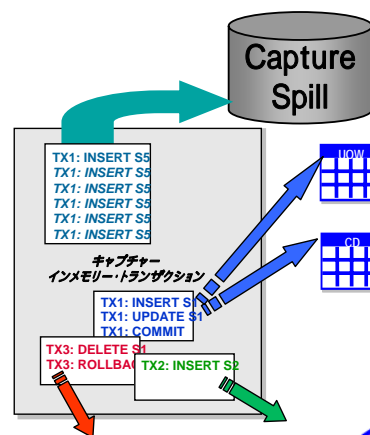
DB2 Universal Database

キャプチャー スピル・ファイル

- キャプチャーは、コミット前の更新情報をメモリーに保管
- メモリーのデフォルト・サイズは32M
- トランザクションがメモリーに入りきらなくなると、キャプチャーは最大のトランザクションをスピル・ファイルに書き出す
 - LUWの場合: Capture_pathに指定されたディレクトリーのファイル
 - z/OS の場合: 仮想入出力 (VIO)

■ スピル・ファイルのサイズに影響する要因

- メモリー限度
 - memory_limit 稼働パラメーター
- トランザクションのサイズ
- 同時実行されるトランザクションの数
- コミット・インターバル

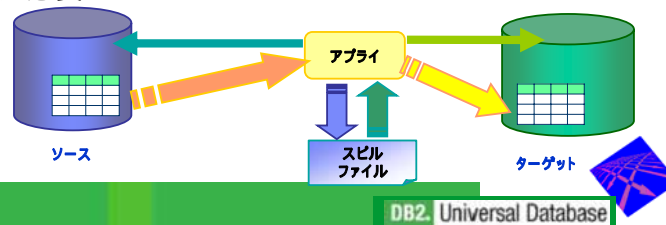


解説:

- 十分なメモリーがない場合、キャプチャー・プログラムはトランザクションをスピル・ファイルに書き込みます (つまり、メモリーに入りきらないトランザクションをここに入れます)。キャプチャー・プログラムは、最大のトランザクションを書き込みます。しかし、最大のトランザクションがメモリー限度を超える原因となったトランザクションであるとは限りません。
- Linux、UNIX、Windows:
 - スピル・ファイルは常にディスク上にある。
 - 1 トランザクション当たり 1 つのファイルが capture_path ディレクトリーに作成されます。
- z/OS
 - スピル・ファイルは仮想入出力 (VIO) に送られます。
- キャプチャースピル・ファイルのサイズは、以下の要因によって決まります。
 - メモリー限度
 - memory_limit 稼働パラメーターを使用して、キャプチャー・プログラムが使用できるメモリー量を指定します。許可するメモリーが多ければ多いほど、キャプチャー・プログラムが入りきらない分をファイルに入れる可能性が少なくなります。
 - トランザクションのサイズ
 - トランザクションのサイズが大きいと、入りきらないためにファイルに入れる必要性が高まります。
 - 同時実行されるトランザクションの数
 - キャプチャー・プログラムが同時に処理するトランザクションの数が増えた場合、またはキャプチャー・プログラムがインターリーブド・トランザクションを処理する場合は、キャプチャー・プログラムはより多くの情報をメモリーまたはディスクに保管する必要があります。
 - コミット・インターバル
 - 通常、コミット・インターバルの値が小さければ小さいほど、必要とされるストレージは少なくなります。なぜなら、キャプチャーが情報をコミットする前にその情報をメモリーに保管しておく期間が短くなるからです。
- LongVarchar LongGraphicが含まれる場合、設定メモリー・サイズ以上にメモリーが使用される場合があるので、注意
 - 詳細に関しては、以下のテクニカル・フラッシュ参照
 - 技術 DM-05-020 'Q/SQLレプリケーション: ソース表に複数のLong Field列がある場合のメモリーに関する注意点'

アプライ スピル・ファイル

- アプライのスピル・ファイルは、レプリケーション対象のデータを一時的に保管するためのもの
 - LUWの場合：Apply pathに指定されたディレクトリーのファイル
 - z/OS の場合：メモリーに入れるように指定可能
- スピル・ファイルのサイズ
 - 各レプリケーション・インターバル内でのレプリケーション対象データのサイズに比例
 - 通常はこのデータのサイズのおよそ 2 倍
- スピル・ファイルの行サイズ
 - Linux、UNIX、Windows、および z/OS
 - ターゲット行のサイズ+レプリケーション・オーバーヘッド
- フルリフレッシュの時
 - ソース表の全てのデータを保管できるスペースが必要
 - ASNLOAD ユーティリティを使用する場合は、EXPORTしたファイルを使用
- 1つのセットに複数メンバーが定義されている場合
 - 各メンバー用にスピル・ファイルが作成される
 - それらが同時に保管できるスペースが必要



解説：

- アプライ・プログラムには、データを保管するための一時的なスペースが必要です。アプライ・プログラムは、更新をターゲット表に適用するまでの間それらの更新を保留するためにスピル・ファイルを使用します。
- 一般に、スピル・ファイルはディスク・ファイルです。ただし、z/OS オペレーティング・システムでは、入りきらないデータをメモリーに入れるように指定できます。仮想メモリーの制約がない限り、スピル・ファイルはディスク上ではなく仮想メモリーに保管してください。
- スピル・ファイルのサイズは、各レプリケーション・インターバル内で複製するよう選択されたデータのサイズに比例します。スピル・ファイルのサイズは、通常はこのデータのサイズのおよそ 2 倍です。スピル・ファイルのサイズを見積もるには、アプライ・プログラム用に計画した頻度インターバル (またはデータ・ブロック値) と、同じ期間内 (または変更のピーク時) になされた変更の量を比較します。
- Linux、UNIX、Windows、および z/OS では、スピル・ファイルの行サイズは ターゲット行のサイズであり、これにはあらゆるレプリケーション・オーバーヘッド列が含まれます。
- この行サイズは、DB2 パック内部フォーマットでのサイズではなく、拡張された解釈済みの文字フォーマット (SELECT から取り出された状態) でのサイズです。
- 行には、行の長さ、個々の列ストリング上の NULL 終止符も含まれます。
- フルリフレッシュの時は、ソース表の全てのデータを保管できるスペースが必要です。ASNLOAD ユーティリティを使用する場合は、スピル・ファイルではなくロード入力ファイル (EXPORTしたファイル) になります。
- 1つのセットに複数メンバーが定義されている場合は、各メンバー用にスピル・ファイルが作成され、かつそれらが同時に保管できるスペースが必要になります。

サブスクリプション関連



DB2 Universal Database

blank page



DB2 Universal Database

サブスクリプションとメンバー

■ サブスクリプション

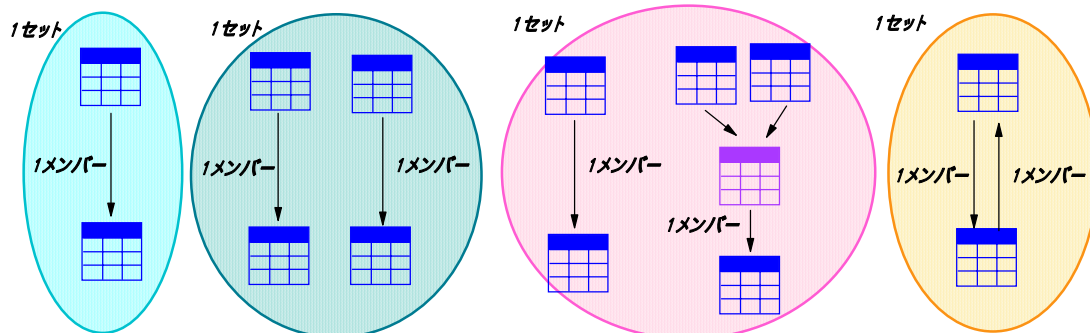
- アプライが必要とする情報を全て含んだ定義体であり、アプライが行うレプリケーションの行為そのものを指すこともある

■ セット

- ひとつのサブスクリプションの単位
- 関連する適用修飾子内でユニークなセット名を持つ
- 1セット内の同方向のレプリケーションが1作業単位で実行される
- 1セットには、少なくとも1メンバーあるいは複数メンバーのレプリケーション定義が含まれる

■ メンバー

- 1表から1表、あるいは1結合ビューから1表へのレプリケーションがメンバーの単位となる



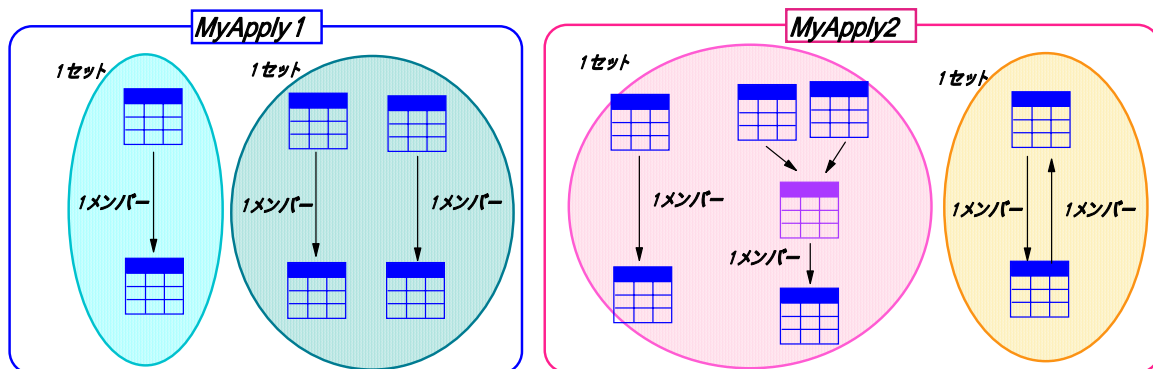
解説:

- アプライが必要とする情報を全て含んだ定義体をサブスクリプションと呼びます。どのようにレプリケーションするかということを定義することを、サブスクリプションを定義するといいます。
- またアプライが行うレプリケーションの行為そのものを指すこともあります
- 1サブスクリプションの単位はセットといいます。よってサブスクリプションを定義することを、サブスクリプション・セットを作成するとも言います。
- 1セットには、少なくとも1メンバーあるいは複数メンバーのレプリケーション定義が含まれます。
- 1表から1表、あるいは1結合ビューから1表へのレプリケーションがメンバーの単位となります。
- 1セット内の同方向のレプリケーションが1作業単位で実行されます。通常はソース表からターゲット表への1方向のレプリケーションとなり、セットの中に複数の表のレプリケーション(メンバー)が定義されていても、それら全てが1つの作業単位(トランザクションのコミット)で処理されます。
- ターゲットのタイプがレプリカの場合は、ソースからレプリカへのレプリケーションとレプリカからソースへのレプリケーションがそれぞれ別の作業単位で行われます。
- セットの必要性
- 整合性が必要な複数の表のレプリケーションを1セットとしてまとめることにより、それらの表のレプリケーションは同一作業論理単位で実行されるようになるので、いずれかの表のレプリケーションが失敗した場合は、セット内の他の表のレプリケーションも同時にロールバックすることができます
- 参照の整合性がある複数表、ターゲットへの更新をトランザクションの発生順に行いたい複数表は1セットにまとめるとよいです。
 - セット内の処理の順序は、ターゲット表の名前の順序で行われますが、参照の整合性が定義されている表間で更新の順序をソース表でのトランザクションの発生順序にしたい場合は、アプライの開始パラメータのCOMMIT(n)(V7まで)、あるいはセット単位でCOMMIT_COUNT(n)(V8から)を指定してください。



適用修飾子 (Apply Qualifier)

- 適用修飾子
 - アプライが処理をするべきサブスクリプションのグループを示す識別子
 - アプライ・コントロール・サーバー内でユニークな識別子を持つ
 - 各サブスクリプション・セットは定義時に、どの適用修飾子で実行させるのかを決定しておく
 - 1つの適用修飾子に含められるセットの数には、特に制限はない
- アプライ実行時に適用修飾子を指定することにより、その適用修飾子のグループに含まれるサブスクリプション・セットのみを処理することができる



解説:

- 適用修飾子 (Apply Qualifier) とはアプライが処理をするべきサブスクリプションのグループを示す識別子です。
- 予め各サブスクリプション・セットは、どの適用修飾子で実行させるのかを定義しておく必要があります。
- 1つの適用修飾子に含められるセットの数には、特に制約はありませんが、各セットは定義されているスケジュールによって、順次に処理されますので、1つの適用修飾子の中で複数のセットが同時に処理されることはありません。



サブスクリプション(セット)の考え方

- 基本的には1セット1メンバーでよい
- 複数メンバーを1セットに含める必要性
 - 参照の整合性がある複数の表
 - 論理的に同期してレプリケーションさせたい複数の表
 - 同じトランザクションで更新される複数の表を、トランザクションの発生順にターゲット表も更新させたい場合など
- 1回のレプリケーション量が特に多い表は、上記のような要件がない限り、1セットで独立させる
 - レプリケーション時にエラーが発生し、ロールバックした時の影響を最小にするため
- 考慮点
 - アプライはセットの処理ごとに、ソース・サーバーに接続
 - 多数の表のレプリケーションを定義していて、かつレプリケーション間隔が短い場合、1セット1メンバーだと、ソース・データベースへの接続の負荷が高くなる可能性がある
 - その場合、論理的な関係がない表でも複数のメンバーを1セットにまとめることにより、接続の負荷の軽減が期待できる
 - 1セットにまとめても、セット内の全ての表のレプリケーションが現行の時間間隔内で収まる程度のものをまとめる
 - 1セットに定義できるメンバーは200個まで



DB2 Universal Database

解説:

- 1セットに含めるメンバーは、基本的には1メンバーずつでよいでしょう。ただし以下のような要件がある場合は、関連する複数の表のレプリケーションを1つのセットに含めなければなりません。
 - 参照の整合性がある複数の表
 - 論理的に同期してレプリケーションさせたい複数の表
 - 同じトランザクションで更新される複数の表を、トランザクションの発生順にターゲット表も更新させたい場合など
- 1回のレプリケーション量が特に多い表は、上記のような要件がない限り、1セットで独立させることをお勧めします。なぜならレプリケーション時にエラーが発生し、ロールバックした場合、レプリケーションが成功した他の表もロールバックされ、次のレプリケーション時に再度時間がかったり、その逆として大量更新が成功した後に、別の表のエラーによって、成功した大量更新がロールバックされたりすることを避けるためです。
 - なお、1回のレプリケーション量が多い表に関しては、ロールバックの単位を最小限にするため、COMMIT COUNTの指定をお勧めします。
- 1セット1メンバーにした場合の考慮点としては、以下のことがあげられます
 - アプライはセットの処理ごとに、ソース・サーバーに接続、つまりCONNECTを発行します。そのため、100を超えるような多数の表のレプリケーションを、それぞれ1分間隔で指定しているような場合は、1分の間に100以上のソース・データベースへの接続要求が発生することになります。
 - 特にソース・データベースがOS/390(またはz/OS)の場合、このようなケースでは、DDFの負荷が非常に高くなる可能性があります。
 - このような場合は、論理的な関係がない表でも複数のメンバーを1セットにまとめることにより、接続の負荷の軽減を図ることが可能です。
 - ただし1セットにまとめても、セット内の全ての表のレプリケーションが現行の時間間隔内で収まる程度のものをまとめるようにしてください。
 - なお、1セットに定義できるメンバーは200個までという制限があります。



DB2 Universal Database

適用修飾子とサブスクリプション(セット)の考え方

- 適用修飾子はアプライの開始、停止の単位
 - 業務ごとでレプリケーションの開始、停止を行いたい場合は、それぞれ適用修飾子を分ける
- 適用修飾子内のセットは、並行処理されない
 - 同じタイミングで処理すべきセットがあっても、1つのアプライはそれらを並行処理はできない
 - アプライの並行処理度を上げたい場合は、適用修飾子を複数にする
- 1回のレプリケーションに時間のかかるセットと、レプリケーション間隔が短いセットは同じ適用修飾子には含めない
 - あるセットは1回のレプリケーションに10分かかる
 - 同じ適用修飾子の他のセットは1分間隔で定義
 - 10分かかるレプリケーションを処理中は、他のセットの処理はできない
 - 長時間かかるセットは独立させて、専用の適用修飾子で動かすようにする
- 1セット、1適用修飾子の場合、アプライの開始パラメータのOPT4ONEを指定すると、毎回サブスクリプションの情報を制御表から読み込む処理を回避することが可能
- 適用修飾子の数は多すぎても煩雑になり、システム資源も消費するため、運用面との兼ね合いで判断



解説:

- 適用修飾子はアプライの開始、停止の単位となるので、業務ごとでレプリケーションの開始、停止を行いたい場合は、それぞれ適用修飾子を分けるとよいでしょう。
- またその時間に処理すべきセットが複数あっても、アプライは一時点で1つのセットの処理しかできません。そのためアプライの並行処理度を上げたい場合は、適用修飾子を分け、複数のアプライを起動してください。
- 上記と同じ理由からですが、1回のレプリケーションに時間のかかるセットと、レプリケーション間隔が短いセットは同じ適用修飾子には含めないようにしましょう。
- 例えばあるセットは1回のレプリケーションに10分かかるとします。同じ適用修飾子の他のセットは1分間隔で定義されていますが、アプライがこの10分かかるレプリケーションを処理している間、他のセットの処理はできません。従って他のセットは1分間隔に定義していても、それが実現できないこととなります。
- このような場合は、長時間かかるセットは独立させて、専用の適用修飾子で動かすようにするとよいでしょう。
- また1セット、1適用修飾子の場合、アプライの開始パラメータのOPT4ONEを指定すると、毎回サブスクリプション情報を制御表から読み込む処理を回避することが可能です。
- ただし、適用修飾子の数は多すぎても煩雑になり、システム資源も消費するため、運用面との兼ね合いで判断 してください。
- アプライもマルチスレッドのトランザクションのため、1アプライで4つのdb2agentが起動します。そのためアプライを複数稼働させる場合は、データベースのメモリーやMAXAGENTの数などにも考慮が必要です。



レプリケーションの定義



内容

- レプリケーションの定義
 - レプリケーション・センター
 - ASNCLP



レプリケーション・センター

- レプリケーション・センターとは？
- ランチパッド
- コントロール表の作成
- 表の登録
- サブスクリプションの作成



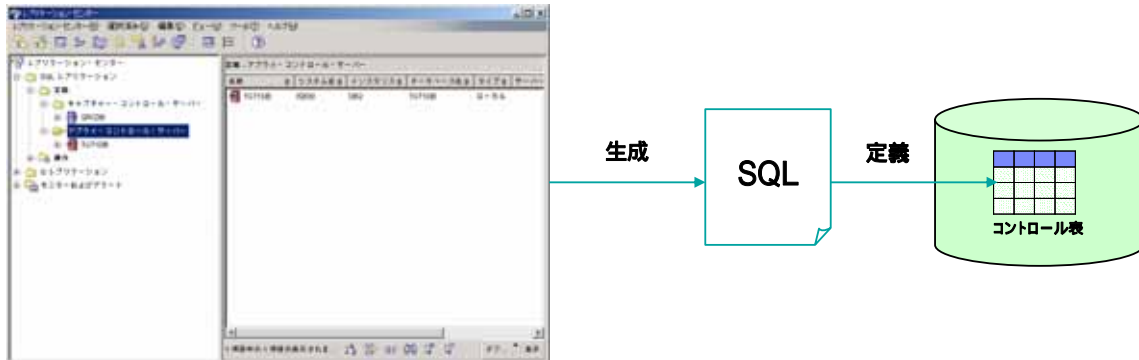
blank page



レプリケーション・センターとは？

■ レプリケーション・センター

- レプリケーション定義、操作のためのSQL、コマンドを生成するツール
 - コントロール表の作成
 - 表の登録
 - サブスクリプション・セット、メンバーの作成
 - キャプチャー、アプライ起動
 - フル・リフレッシュ
 - など



解説

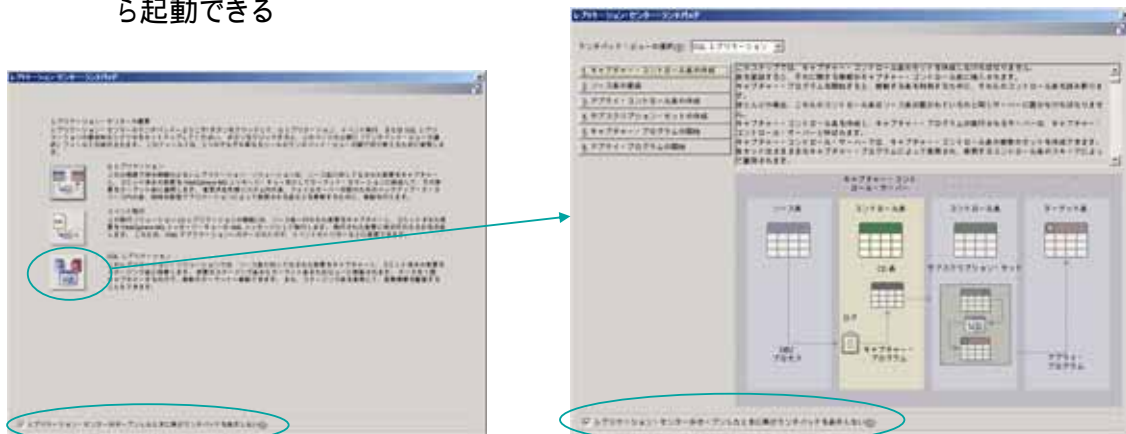
- レプリケーションの定義は、レプリケーションのコントロール表に格納されます。コントロール表の作成、コントロール表のデータの作成はSQLによって行われます。これらのSQLを生成するツールがレプリケーション・センターです。
- レプリケーション・センターはV8.1から提供されています。V8.2からSQLレプリケーションに加え、Qレプリケーションの定義も可能です。
- レプリケーション・センターでは、コントロール表の作成、表の登録、サブスクリプション・セットの作成などのレプリケーションの定義だけでなく、アプライ、キャプチャーの起動コマンドの生成も可能です。また、自動、手動フル・リフレッシュに必要なスクリプトの生成も可能です。
- 生成されたスクリプトは、即時に実行可能です。また、スクリプトとして保管することも可能です。



ランチパッド

■ ランチパッド

- ランチパッドに従って操作していくと簡単にレプリケーションの定義が可能
- レプリケーション・センターの初回起動時に起動される
 - 必要であれば、「レプリケーション・センターがオープンしたときに再びランチパッドを表示しない」にチェックする
- レプリケーション・センターのメニュー【レプリケーション・センター】 【ランチパッド】から起動できる



ランチパッドの起動が必要であればチェックします。



解説

- 初めてレプリケーション・センターを起動した時にランチパッドが起動されます。必要であれば閉じてください。
- 最初の画面では、Qレプリケーション、イベント・パブリッシング、SQLレプリケーションのボタンが表示されます。
- ランチパッドに従って、操作していくと簡単にレプリケーションの構成、定義、実行が行えます。レプリケーションの操作に慣れていない場合に有効です。
- 次回からランチパッドの起動が必要であれば、「レプリケーション・センターがオープンしたときに再びランチパッドを表示しない」にチェックをします。
- レプリケーション・センターのメニュー【レプリケーション・センター】 【ランチパッド】からランチパッドを起動することができます。



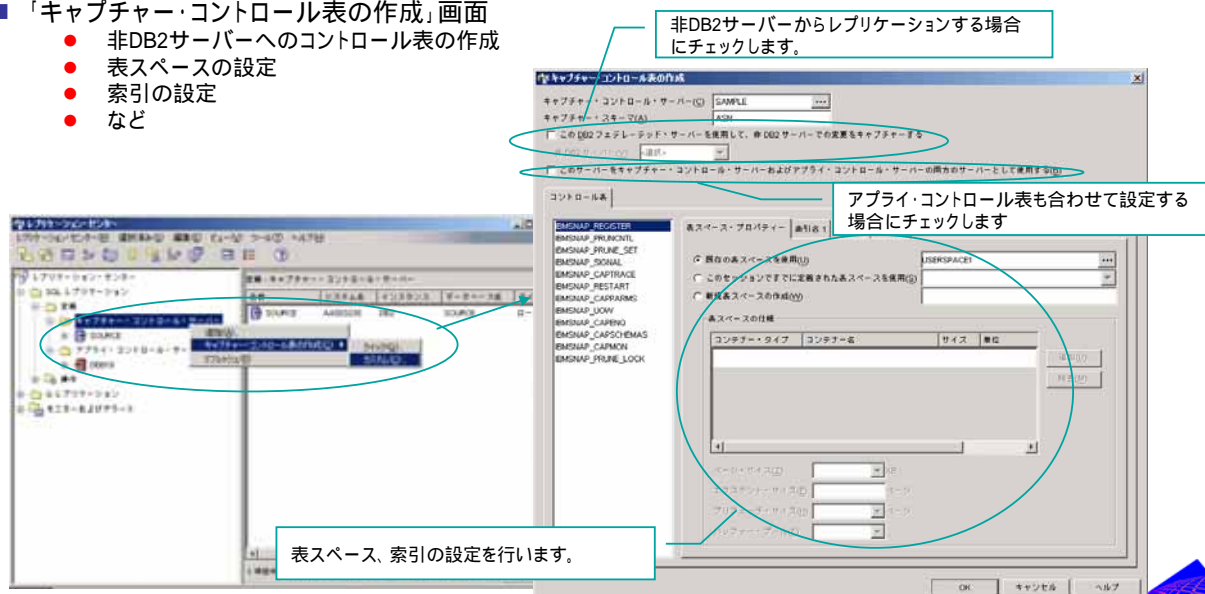
コントロール表

■ キャプチャー・コントロール表の作成

- レプリケーション・センターのツリーから【SQL レプリケーション】 【定義】 【キャプチャー・コントロール・サーバー】を右クリックし、【キャプチャー・コントロール表の作成】 【カスタム】を選択し、「キャプチャー・コントロール表の作成」を起動

■ 「キャプチャー・コントロール表の作成」画面

- 非DB2サーバーへのコントロール表の作成
- 表スペースの設定
- 索引の設定
- など



解説

- 「キャプチャー・コントロール表の作成」画面では、コントロール表を格納するための表スペースの属性の設定、コントロール表に付けられる索引の属性の設定が可能です。
 - 表スペースは、新規に作成も可能ですし、既存の表スペースを使用することも出来ます。
- 非DB2サーバーからレプリケーションを行う場合に、その非DB2サーバー上にキャプチャー・コントロール表を作成する必要があります。
 - 「このDB2フェデレーテッド・サーバーを使用して、非DB2サーバーでの変更をキャプチャーする」にチェックをつけて、連合サーバーで定義されているサーバーオブジェクトを選択します。
- 選択しているサーバーにキャプチャー・コントロール表のみではなく、アプライ・コントロール表も合わせて作成することができます。
 - 「このサーバーをキャプチャー・コントロール・サーバーおよびアプライ・コントロール・サーバーの両方のサーバーとして使用する」にチェックをします。
- キャプチャー・コントロール表の作成により、以下の12個の表が作成されます。
 - IBMSNAP_CAPSCHEMAS
 - IBMSNAP_CAPENQ
 - IBMSNAP_CAPMON
 - IBMSNAP_CAPPARMS
 - IBMSNAP_CAPTRACE
 - IBMSNAP_PRUNCNTL
 - IBMSNAP_PRUNE_LOCK
 - IBMSNAP_PRUNE_SET
 - IBMSNAP_REGISTER
 - IBMSNAP_RESTART
 - IBMSNAP_SIGNAL
 - IBMSNAP_UOW

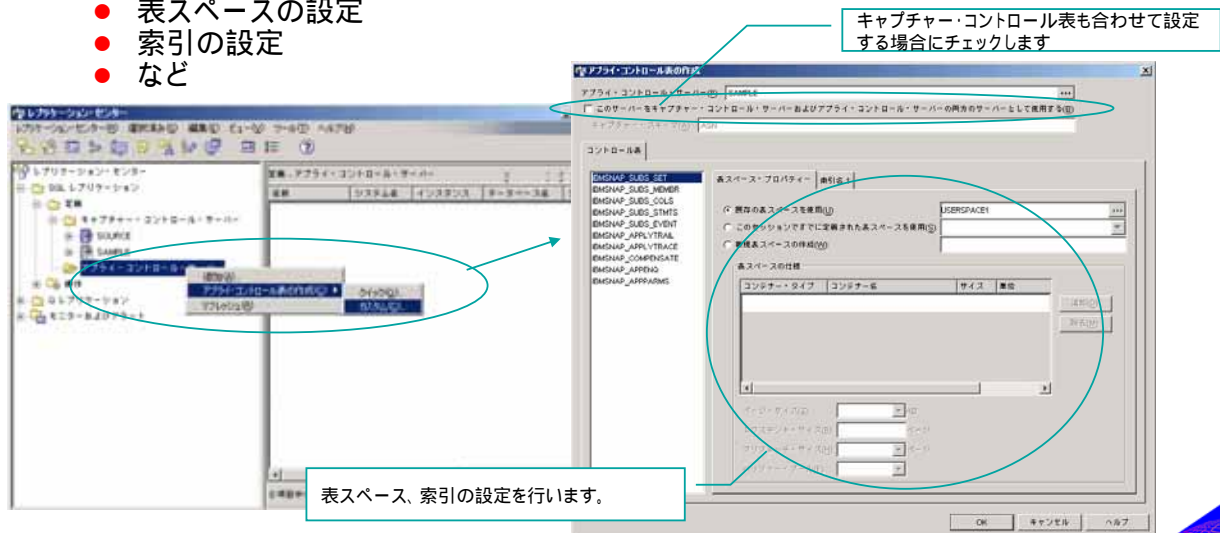
コントロール表

■ アプライ・コントロール表の作成

- レプリケーション・センターのツリーから【SQL レプリケーション】 【定義】 【アプライ・コントロール・サーバー】を右クリックし、【アプライ・コントロール表の作成】 【カスタム】を選択し、「アプライ・コントロール表の作成」画面を起動

■ 「アプライ・コントロール表の作成」画面

- 表スペースの設定
- 索引の設定
- など



解説

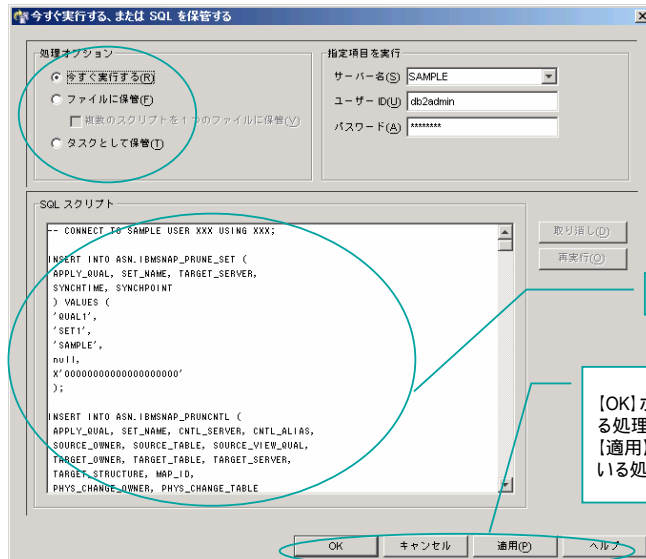
- 「アプライ・コントロール表の作成」画面では、コントロール表を格納するための表スペースの属性の設定、コントロール表に付けられる索引の属性の設定が可能です。
 - 表スペースは、新規に作成も可能ですし、既存の表スペースを使用することも出来ます。
- 選択しているサーバーにアプライ・コントロール表のみではなく、キャプチャー・コントロール表も合わせて作成することができます。
 - 「このサーバーをキャプチャー・コントロール・サーバーおよびアプライ・コントロール・サーバーの両方のサーバーとして使用する」にチェックをします。
- アプライ・コントロール表の作成により、以下の10個の表が作成されます。
 - IBMSNAP_SUBS_SET
 - IBMSNAP_SUBS_MEMBER
 - IBMSNAP_SUBS_COLS
 - IBMSNAP_SUBS_STMTS
 - IBMSNAP_SUBS_EVENT
 - IBMSNAP_APPLYTRAIL
 - IBMSNAP_APPLYTRACE
 - IBMSNAP_COMPENSATE
 - IBMSNAP_APPENQ
 - IBMSNAP_APPPARMS



SQLスクリプトの実行、保管

■ 「今すぐ実行する、またはSQLを保管する」画面

- スクリプトの生成後に表示され、生成されたスクリプトに対して以下の3つの作業が可能
 - 今すぐ実行
 - ファイルに保管
 - タスクとして保管



生成されたSQLが表示されます。

【OK】ボタンを押すと【処理オプション】で指定されている処理が実行され、ウィンドウが閉じられます。
【適用】ボタンを押すと【処理オプション】で指定されている処理が実行されます。



解説

- 「今すぐ実行する、またはSQLを保管する」画面は、スクリプトの生成後に表示されます。
- 生成されたスクリプトが表示されます。そのスクリプトを編集することもできます。
- 生成されたスクリプトをその場で実行することが可能です。ファイルに保管して、後で実行することも可能です。タスク・センターが構成されていれば、タスクとして保管することも可能です。
- 【OK】ボタンを押すと【処理オプション】で指定されている処理が実行され、ウィンドウが閉じられます。【適用】ボタンを押すと【処理オプション】で指定されている処理が実行されます。ファイルに保管、今すぐ実行など続けて処理が必要な場合には【適用】ボタンを使用します。



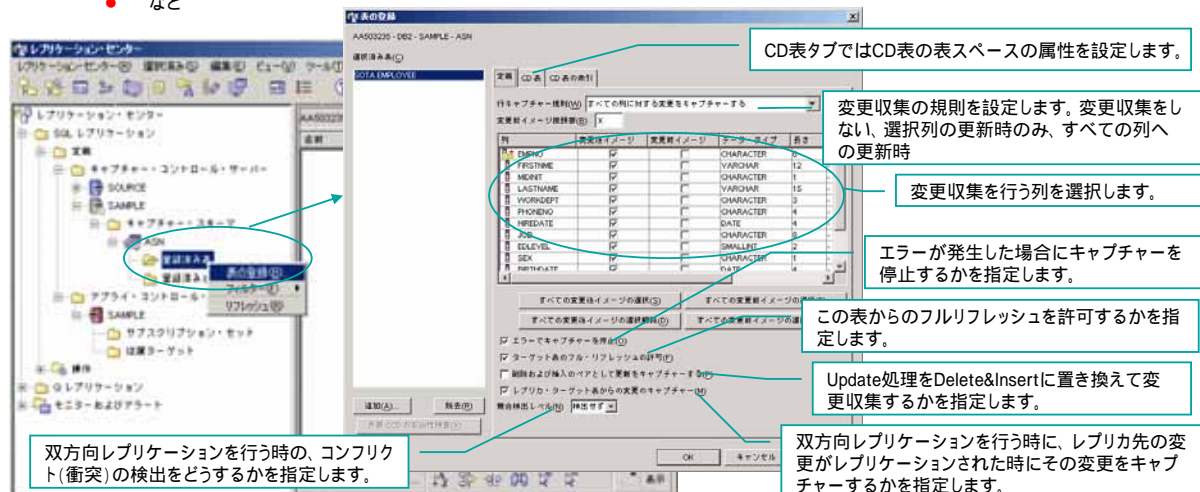
表の登録

■ 表の登録

- レプリケーション・センターのツリーから【SQL レプリケーション】 【定義】 【キャプチャー・コントロール・サーバー】 【データベース名】 【キャプチャー・スキーマ】 【ASN】 【登録済み表】を右クリックし、【表の登録】を選択し、「表の登録」画面を起動

■ 「表の登録」画面

- レプリケーションを行う列の選択
- CD表の表スペースの設定
- エラー時の動作
- フルリフレッシュの許可
- Update処理のDelete&Insertへの変換
- など



解説

- 表の登録では、レプリケーションを行う表をキャプチャー・コントロール表へ登録します。このときCD表も作成されます。

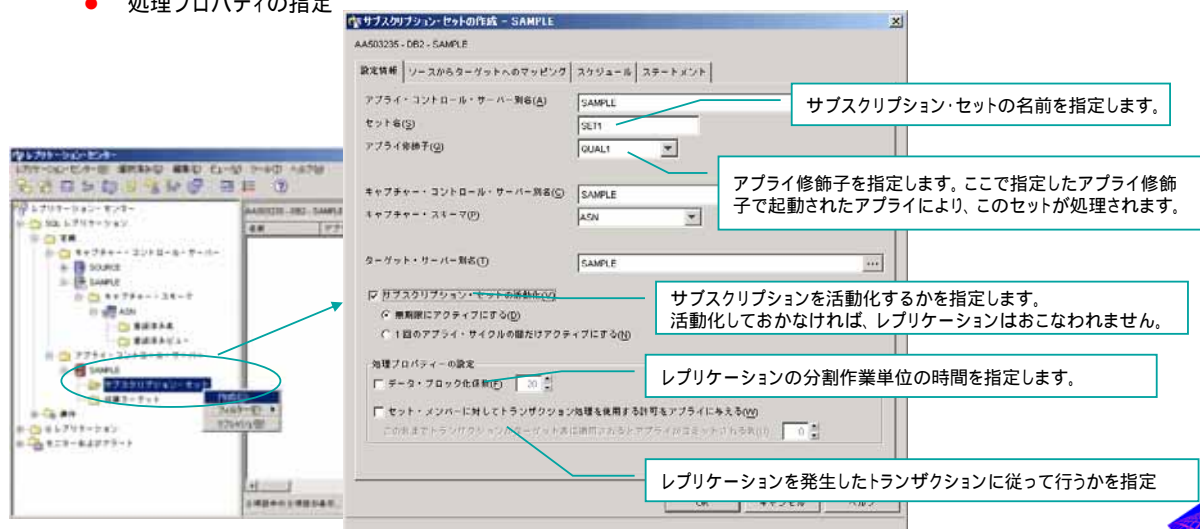
- 「表の登録」画面では、変更収集を行う列の指定、CD表の表スペースの設定、変更収集の属性の設定が可能です。

- レプリケーション・キー列のUpdateがある場合には、ここの設定に注意してください。

- レプリケーション・キー列のUpdateをサポートするためには2つの方法があります。
 1. 変更前イメージを取得しておき、キー列の変更前イメージで行と特定しUpdateする方法
 - レプリケーション・キー列の変更前イメージを選択しておき、このあとサブスクリプション・メンバーの設定で更新前イメージを使用させるように設定します。
 2. Update処理をDelete&Insert処理として扱う方法
 - 「削除および挿入のペアとして更新をキャプチャーする」にチェックをします。

サブスクリプション・セットの作成

- サブスクリプション・セットの作成
 - レプリケーション・センターのツリーから【SQL レプリケーション】、【アプリ・コントロール・サーバー】、【データベース名】、【サブスクリプション・セット】を右クリックし、【作成】を選択し、「サブスクリプション・セットの作成」画面を開きます。
- 「サブスクリプション・セットの作成」画面の「設定情報」タブ
 - 使用するアプリおよびキャプチャー・コントロール表の指定
 - ターゲット・サーバーの指定
 - セットの活動化の指定
 - 処理プロパティの指定



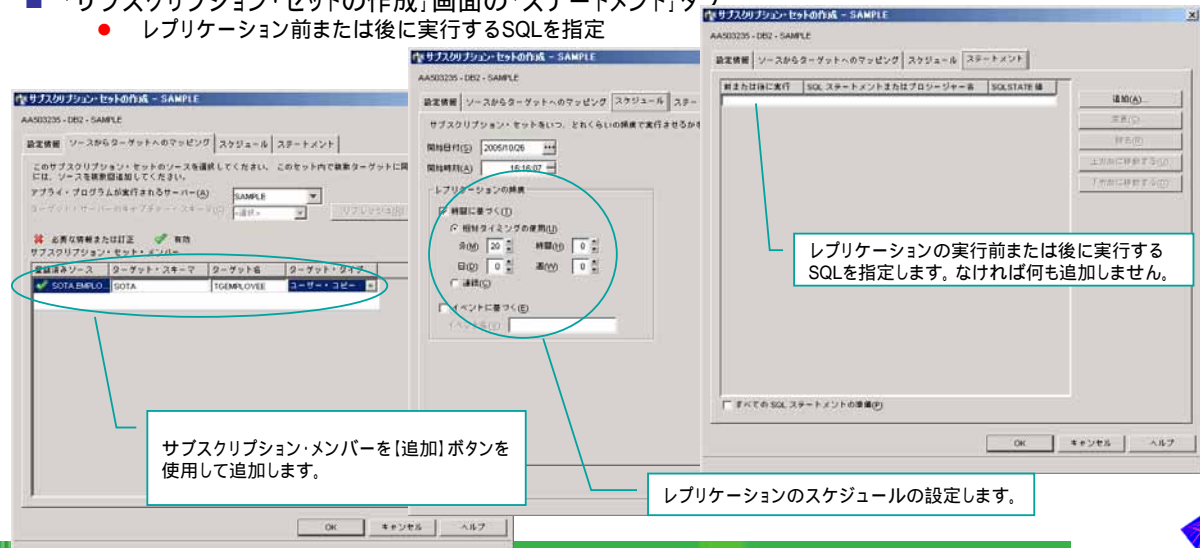
解説

- 「サブスクリプション・セットの作成」画面の「設定情報」タブでは、キャプチャー・コントロールサーバーの指定、アプリ・コントロール・サーバーの指定を行います。
- サブスクリプション・セットの名前も指定します。サブスクリプション・セット名はアプリ・コントロール・サーバーで一意になるように指定します。
- アプリ修飾子の指定も行います。アプリ修飾子はアプリの起動単位に設定される名前です。ここで指定されたアプリによりこのセットは処理されます。
- サブスクリプションの活動化の設定も行います。デフォルトでは、非活動化の設定になっているので活動化するように設定します。非活動化となっている場合にはアプリにより、そのサブスクリプション・セットは処理されません。
- 処理プロパティの設定では、ターゲットへの反映処理の仕方について設定します。デフォルトでは、1つのサブスクリプションは1つのUOWで反映されます。そして反映順序は、ターゲットの表名の順になります。
- 「データブロック化係数」が指定された場合には、そこに指定した時間単位でレプリケーションの反映の単位が分割されます。
- 「セット・メンバーに対してトランザクション処理を使用する許可をアプリに与える」が指定された場合には、レプリケーションの反映順序がソース表が更新されたトランザクション順序になります。「この数までトランザクションがターゲット表に適用されるとアプリがコミットされる数」は、指定された数のトランザクションが反映されたタイミングでコミットされます。



サブスクリプション・セットの作成

- 「サブスクリプション・セットの作成」画面の「ソースからターゲットへのマッピング」タブ
 - サブスクリプション・メンバーの指定
 - ソース表、ターゲット表のマッピングを指定
 - ターゲットタイプの指定
- 「サブスクリプション・セットの作成」画面の「スケジュール」タブ
 - レプリケーションの起動タイミングを指定
- 「サブスクリプション・セットの作成」画面の「ステートメント」タブ
 - レプリケーション前または後に実行するSQLを指定

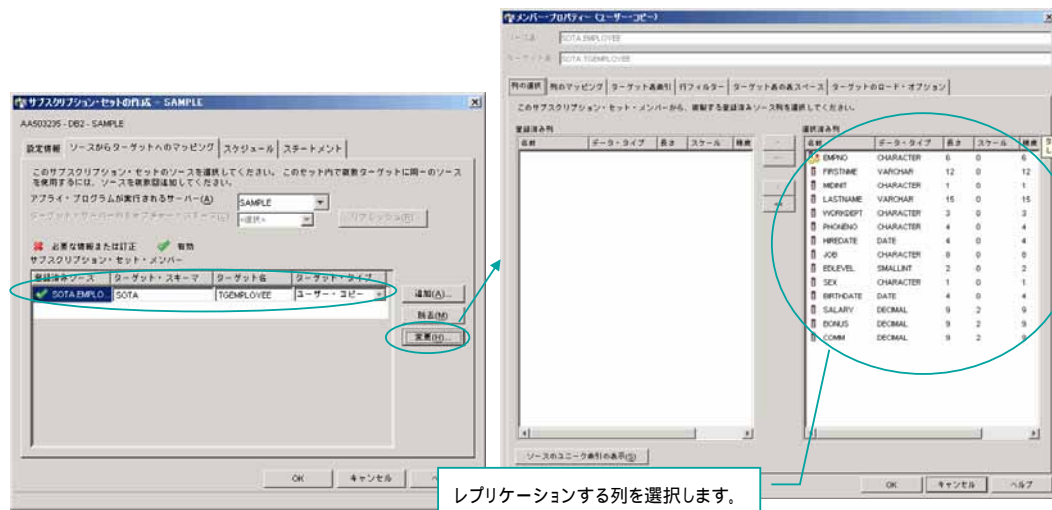


解説

- 「ソースからターゲットへのマッピング」タブでは、[追加] ボタンにより、サブスクリプション・メンバーを追加します。
 - [追加] ボタンを押すと登録された表を選択します。デフォルトでは、プロファイルに設定されたターゲット表が設定されています。
 - ターゲット表名を既存表に設定することも可能です。指定されたターゲット表名が既存表でなければ、新規にターゲットを作成するスクリプトが生成されます。
 - ターゲット・タイプは、以下の6つから選択できます。
 - ユーザー・コピー
 - 基礎集約
 - 変更集約
 - ポイント・イン・タイム
 - レプリカ
 - 整合変更データ(CCD)
- 「スケジュール」タブでは、サブスクリプションのスケジューリングを行います。
 - 「時間に基づく」は、指定した時間間隔でレプリケーションを実施するように指定できます。
 - 「イベントに基づく」は、ASN、IBMSNAP、SUBS_EVENTにセットされるイベントをトリガーにレプリケーションを実施するように設定できます。
- 「ステートメント」タブでは、レプリケーション実行前または後に実行するSQLを指定します。必要なければ、何も入力しません。

サブスクリプション・セットの作成

- 「メンバー・プロパティ」画面
 - 「サブスクリプション・セットの作成」画面の「ソースからターゲットへのマッピング」タブでサブスクリプション・メンバーを選択し、【変更】ボタンを押すと表示
- 「列の選択」タブ
 - レプリケーションする列を選択



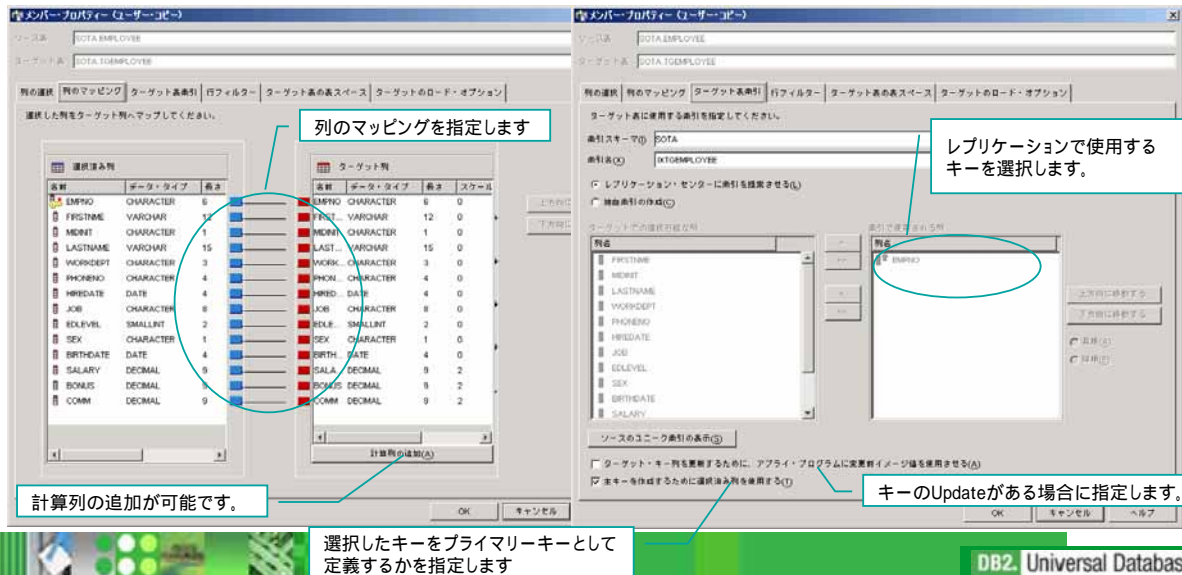
解説

- 「メンバー・プロパティ」画面は、メンバーで指定されているターゲット・タイプ毎に異なります。
 - 上の図は、ターゲット・タイプがユーザー・コピーのもので。
- 「列の選択」タブでは、レプリケーションに必要な列を選択します。
 - 計算列の追加は「列のマッピング」タブで行います。



サブスクリプション・セットの作成

- 「列のマッピング」タブ
 - ソース表の列とターゲットの列のマッピングを指定
 - 計算列の追加が可能
- 「ターゲット表索引」タブ
 - レプリケーション・キーの指定

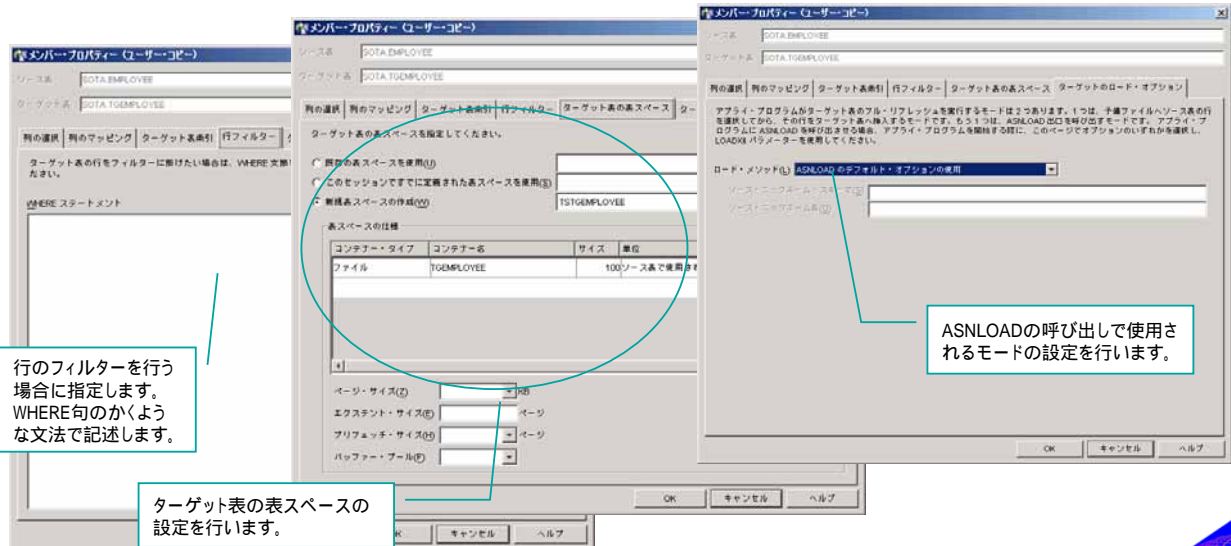


解説

- 「列のマッピング」画面では、ソース表とターゲット表の列のマッピングを指定します。計算列の追加も可能です。
- 「ターゲット表索引」タブでは、レプリケーションで使用するキーの設定を行います。
 - レプリケーションでは、レプリケーション対象の行を識別するためのキーが必要です。このキーを設定します。
- ここで指定したキーに対してUpdateがある場合には、「ターゲット・キー列を更新するために、アプライ・プログラムに更新前イメージを使用させる」を指定します。
 - この設定を行う場合には、表の登録の設定の時に更新前イメージが取得されていなければなりません。
- 選択したキーをプライマリー・キーとして作成する場合には、「主キーを作成するために選択済み列を使用する」を指定します。
 - これを指定しない場合には、その列をキーにしたユニーク・インデックスを作成するスクリプトが生成されます。

サブスクリプション・セットの作成

- 「行フィルター」タブ
 - フィルタリングの設定
- 「ターゲット表の表スペース」タブ
 - ターゲット表の表スペースの設定
- 「ターゲットのロード・オプション」タブ
 - ASNLOADの呼び出しで使用するモードの設定



解説

- レプリケーションするデータにフィルタリングが必要な場合に「行フィルター」タブに設定を行います。
 - 記述の仕方は、SQLのWHERE句の形式で記述します。
- 「ターゲット表の表スペース」タブでは、ターゲット表の表スペースの設定を行います。
 - 既存表が指定されている場合はこのタブは表示されません。
- 「ターゲットのロード・オプション」タブでは、ASNLOADの呼び出しのモードの指定を行います。
 - フルリフレッシュを行う時に、LOADのExitルーチンとしてASNLOADを呼び出すことが可能です。
 - 以下の設定が可能です。
 - 適切なユーティリティが選択されます。
 - ASNLOADを呼び出しません。
 - ユーザーが提供するロジックを呼び出します。
 - カーソルを使用したロードを行います。(クロスローダー)
 - EXPORTとLOADユーティリティを使用します。
 - EXPORTとIMPORTユーティリティを使用します。



ASNCLP

- レプリケーションの定義
- セットアップ
- ASNCLPとレプリケーション・センター
- ASNCLP使い方
- ASNCLPサンプル



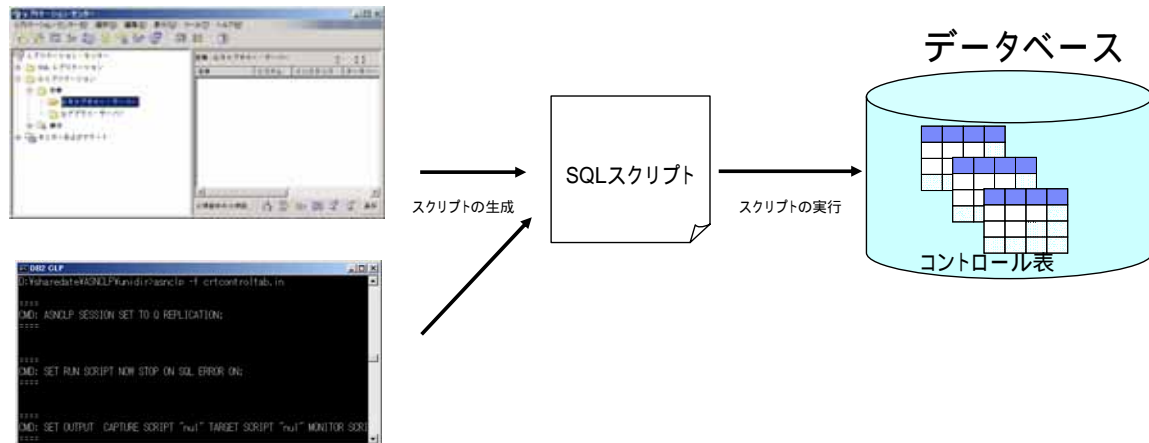
blank page



レプリケーションの定義

■ レプリケーションの定義

- レプリケーション・センター
- ASNCPL



解説

- レプリケーション定義は、これまでレプリケーションセンターと呼ばれるGUIベースのツールで行うことが可能です。
 - V8.1からレプリケーション・センターが提供されています。
 - それより、以前はコントロール・センター、DJRAがレプリケーション定義のためのツールとして提供されていました。
- V8.2より、ASNCPLが提供され、このツールはコマンド・ラインからレプリケーション定義を可能にするツールです。
- レプリケーション定義は、SQLスクリプトの生成、実行により行われます。
 - このSQLスクリプトには、コントロール表の作成や、その表のデータの更新などが含まれます。

レプリケーション・センター



ASNCPL

```
ASNCPL SESSION SET TO Q REPLICATION;
SET QMANAGER QMGR FOR CAPTURE SCHEMA;
SET CAPTURE SCHEMA SOURCE TO DEFAULT;
SET SERVER CAPTURE TO DB SAMPLE id db2admin password "db2admin";
CREATE CONTROL TABLES FOR APPLY SERVER;
CREATE CONTROL TABLES FOR CAPTURE SERVER USING RESTARTQ "SAMPLE_RESTARTQ" ADMINQ "SAMPLE_ADMINQ";
```

SQLスクリプト



セットアップ

- ASNCLPはJavaアプリケーションのため以下の準備が必要です。

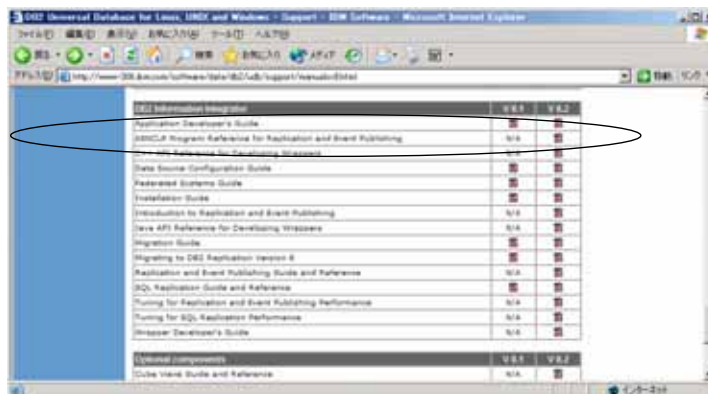
- Javaランタイム 1.3.1以上
 - java -versionで確認
 - 64bitインスタンス上で動作させるには、64bitのJVMを使用する
- CLASSPATHに以下の5つのファイルを含めます。
 - `INSTDIR\sqlib\java\Common.jar`
 - `INSTDIR\sqlib\tools\db2cmn.jar`
 - `INSTDIR\sqlib\tools\db2replapis.jar`
 - `INSTDIR\sqlib\tools\db2qreplapis.jar`
 - `INSTDIR\sqlib\tools\jt400.jar`
- 例: Windows
 - `set CLASSPATH=%CLASSPATH%;c:\sqlib\java\Common.jar;c:\sqlib\tools\db2cmn.jar;c:\sqlib\tools\db2replapis.jar;c:\sqlib\tools\db2qreplapis.jar;c:\sqlib\tools\jt400.jar;`
- 例: UNIX
 - `export CLASSPATH=$CLASSPATH:/home/db2inst/sqlib/java/Common.jar:/home/db2inst/sqlib/tools/db2cmn.jar:/home/db2inst/sqlib/tools/db2replapis.jar:/home/db2inst/sqlib/tools/db2qreplapis.jar:/home/db2inst/sqlib/tools/jt400.jar;`



DB2 Universal Database

解説

- ASNCLPは、Javaのアプリケーションとして提供されています。実行に必要なjarファイルへのクラスパスの設定は、行われていません。マニュアルに従ってクラスパスの設定を行ってください。
- マニュアル(英語)は以下のサイトにあります。
 - <http://www-306.ibm.com/software/data/db2/udb/support/manualsv8.html>



- サンプルも提供されています。
 - サンプルはDB2のインストールディレクトリ\samples\repl\asnclpにあります。



DB2 Universal Database

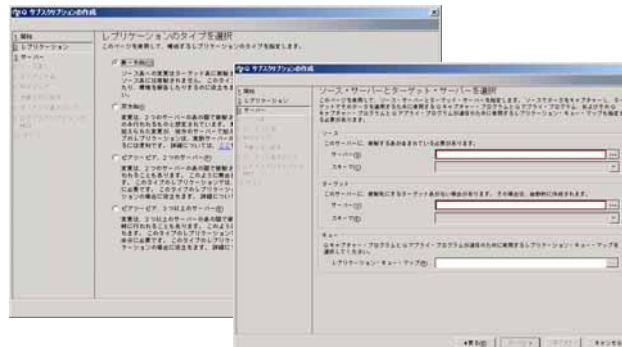
ASNCLPとレプリケーション・センター

■ ASNCLP

- コマンドをスクリプトとしてまとめることによりバッチ的な実行が可能
 - 同じようなレプリケーション定義の作成が必要な場合に有効

■ レプリケーション・センター

- ウィザードに従い入力を行うことで、比較的容易にレプリケーションの定義を行うことができる



解説

- レプリケーション・センターとASNCLPは、内部的に同じAPIを使用してSQLスクリプトを生成します。
 - 同じ設定値で生成されるSQLスクリプトは同じです。
- レプリケーション・センターとASNCLPは、以下に挙げるような作業が可能です。
 - Create/Drop Control Tables
 - Create/Alter/Drop registration
 - Create/Alter/Drop subscription set
 - Create/Alter/Drop member
 - など
- ASNCLPは、ファイルに書かれたASNCLPコマンドを読み込み実行することが可能です。
 - 多数の表のレプリケーションの定義を行う場合には、定義を行うコマンドをまとめておき、読み込ませることで効率的にレプリケーション定義が可能です。



ASNCLP使い方

- 以下の2つのモードでコマンドの実行が可能です。

対話式

```
DB2 CLP - asncpl
D:\work\ASNCLP\unidir>asncpl
Repl > asncpl session set to q replication

****
CMD: asncpl session set to q replication;
****

Repl > set run script now stop on sql error on

****
CMD: set run script now stop on sql error on;
****

Repl > set server capture to db sample id db2admin password "db2admin"
```

asncplを実行すると“Repl>”プロンプトが現れるのでここにコマンドを入力し、対話式で処理を行います。

ファイル読込

```
DB2 CLP
D:\work\ASNCLP\unidir>asncpl -f c:\asncpl.in

****
CMD: ASNCLP SESSION SET TO Q REPLICATION;
****

****
CMD: SET RUN SCRIPT NOW STOP ON SQL ERROR ON;
****

****
CMD: SET OUTPUT CAPTURE SCRIPT "nul" TARGET SCRIPT "nul" MONITOR SCRIPT "nul";
****
```

asncpl f ファイル名でファイルからコマンドを読み取り実行することが出来ます。



解説

- asncpl ? を実行すればヘルプを確認できます。

```
DB2 CLP
D:\work\ASNCLP\unidir>asncpl ?
IBM DPROPR Admin Setup Utility

Usage:
To view this help message:
asncpl ?

To invoke with interactive mode:
asncpl

To exit out of interactive mode:
Use the {QUIT} command

To invoke with input-file mode:
asncpl [-f] [input-filename]

Note:
* The following commands can either be specified individually (interactive mode) or
  together as a batch file (input-file mode).
* For detailed command syntax, contact IBM Software Support.

To set the environment session:
  Use the {ASNCLP SESSION SET TO} command to set the replication session

To setup the admin environment:
  Use the {SET SERVER} command to set replication server options
  Use the {SET PROFILE} command to set DDL profile options
  Use the {SET DROP} command to set table/tablespace drop options
  Use the {SET OUTPUT} command to set sql output script options
  Use the {SET RUN SCRIPT} command to set script execution options
  Use the {SET LOG} command to set logging options
```



サンプル

```
set run script now stop on sql error off;
```

スクリプト生成後、その場で実行されます。

```
set server capture to db srcdb id db2admin password "db2admin";
set server control to db tgt1db id db2admin password "db2admin";
set server target to db tgt1db id db2admin password "db2admin";
```

キャプチャー・サーバーをSRCDB、
アプライ・コントロールサーバーをTGT1DB、
ターゲットサーバーをTGT1DB
に設定しています

```
#drop control tables on capture server;
create control tables for capture server;
```

キャプチャー・コントロール表を作成します

```
#drop control tables on apply control server;
create control tables for apply control server;
```

アプライ・コントロール表を作成します

```
#drop registration (employee);
create registration (employee) differential refresh;
```

EMPLOYEE表をソース表として登録します

```
#drop subscription set setname emp_set applyqual emp;
create subscription set setname emp_set applyqual emp activate yes timing interval 2;
```

サブスクリプションEMP_SETを作成します。

```
#drop member from setname emp_set applyqual emp source employee target tmployee;
create member in setname emp_set applyqual emp activate yes source employee keys (empno +);
```

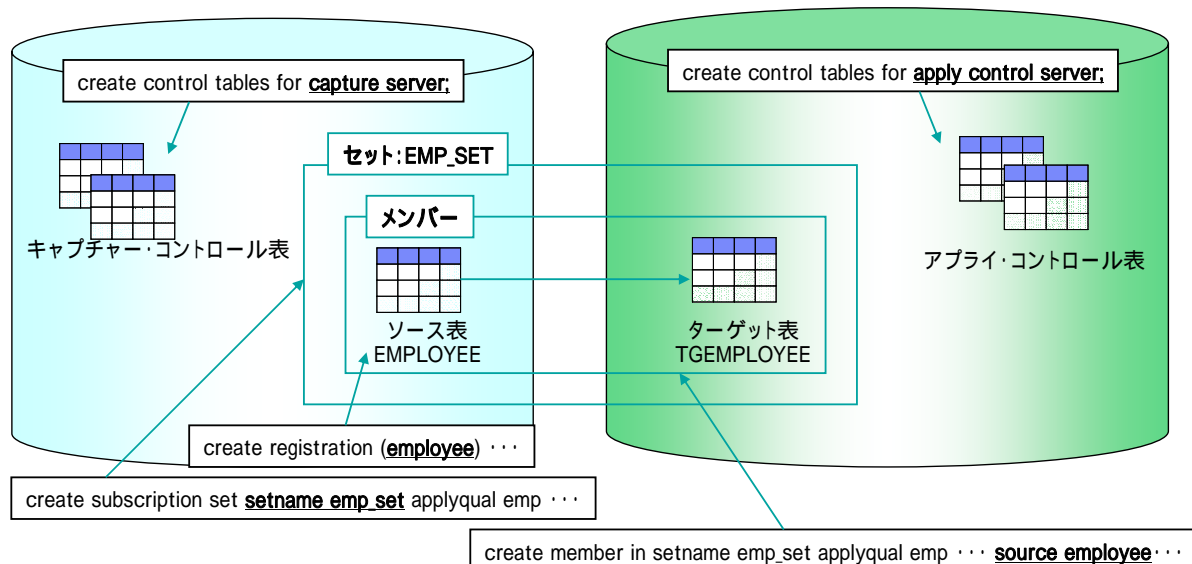
サブスクリプションEMP_SETにソース表EMPLOYEEのメンバーを追加します

解説

- 上記のコマンドで定義されるレプリケーション環境です。

ソース・サーバー
SRCDB

ターゲット・サーバー
TGT1DB



サンプル

```
set run_script now stop on sql error off;
```

```
set server capture to db srcdb id db2admin password "db2admin";
set server control to db tgt1db id db2admin password "db2admin";
set server target to db tgt1db id db2admin password "db2admin";
```

```
#drop control tables on capture server;
create control tables for capture server;
```

```
#drop control tables on apply control server;
create control tables for apply control server;
```

レプリケーションする表をREGISTRATIONしておきます。

```
#drop registration (employee);
create registration (employee) differential refresh;
create registration (department) differential refresh;
create registration (org) differential refresh;
```

```
#drop subscription set setname set1 applyqual qual1;
create subscription set setname set1 applyqual qual1 activate yes timing interval 2;
```

```
#drop member from setname set1 applyqual qual1 source employee target tgeemployee;
create member in setname set1 applyqual qual1 activate yes source employee keys (empno +);
create member in setname set1 applyqual qual1 activate yes source department keys (deptno +);
create member in setname set1 applyqual qual1 activate yes source org keys (deptnumb +);
```

複数のメンバーを作成する場合には、複数回 CREATE MEMBERを繰り返します。

DB2 Universal Database

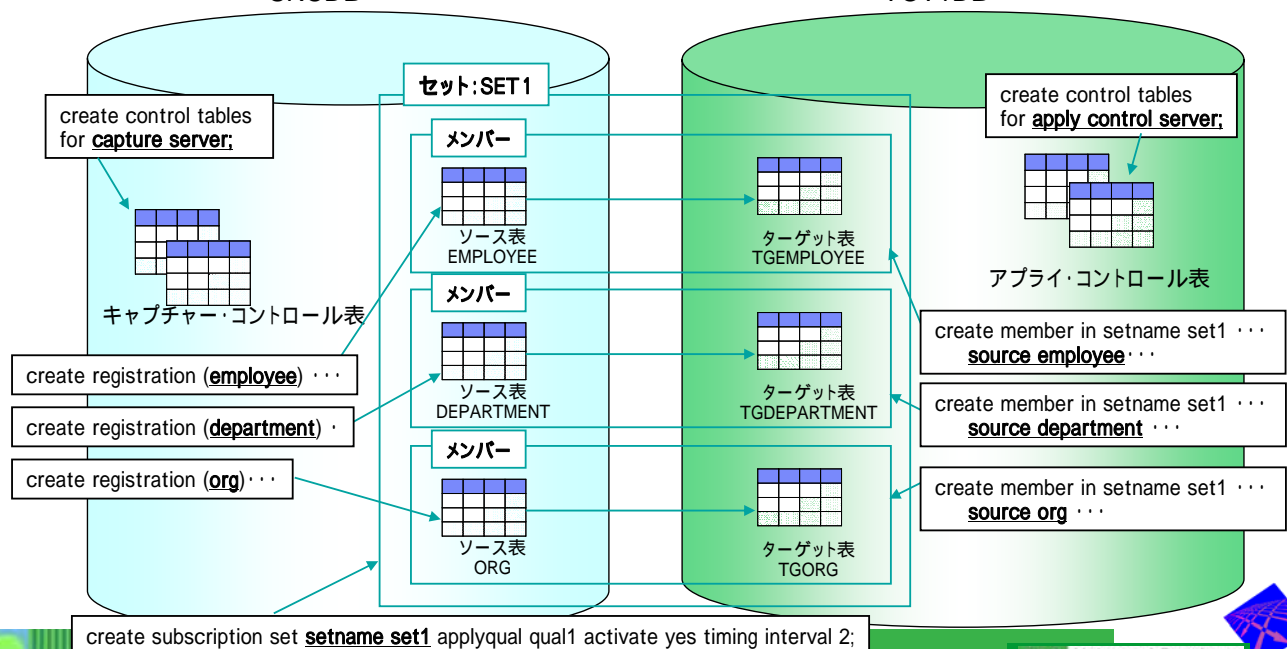


解説

- 上記のコマンドで定義されるレプリケーション環境です。

ソース・サーバー
SRCDB

ターゲット・サーバー
TGT1DB



DB2 Universal Database



サンプル

```
set run script now stop on sql error off;

set server capture to db srcdb id db2admin password "db2admin";
set server control to db tgt1db id db2admin password "db2admin";
set server target to db tgt1db nonibm server ora10g id db2admin password "db2admin";

#drop control tables on capture server;
create control tables for capture server;

#drop control tables on apply control server;
create control tables for apply control server;

#drop registration (employee);
create registration (employee) differential refresh;

#drop subscription set setname emp_set applyqual emp;
create subscription set setname emp set applyqual emp activate yes timing interval 2;

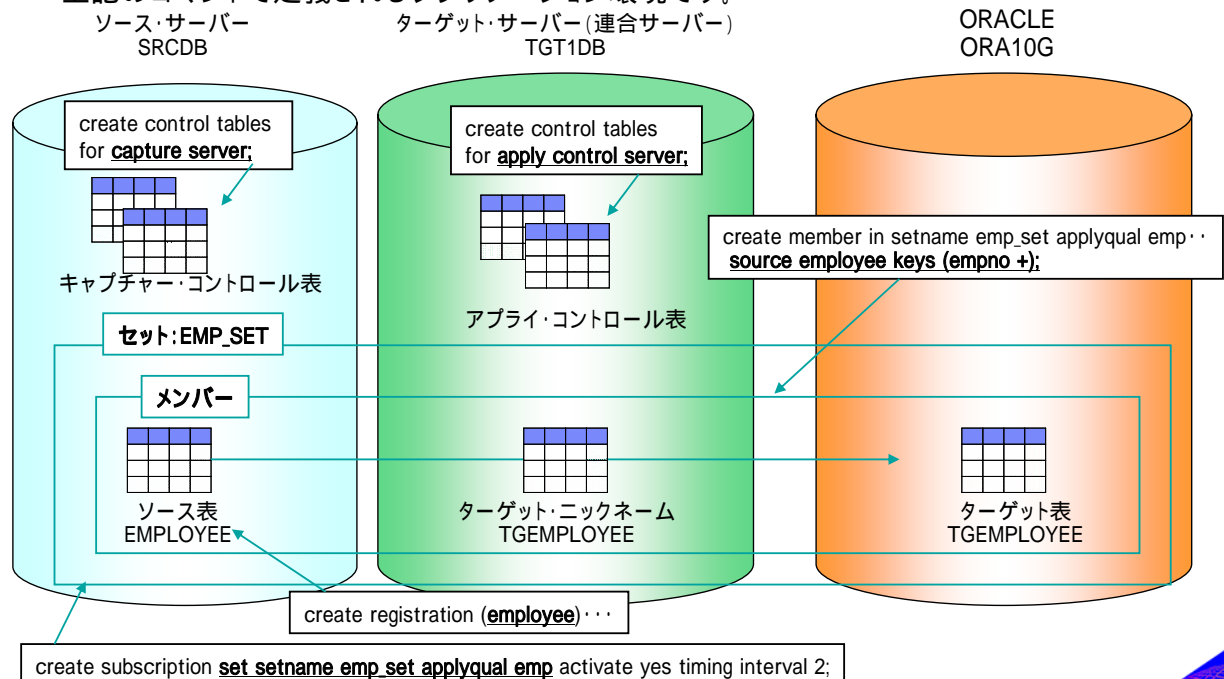
#drop member from setname emp_set applyqual emp source employee target tgemployee;
create member in setname emp_set applyqual emp activate yes source employee keys (empno +);
```

ターゲットサーバーにNONIBMを指定します。
ここで指定するのは、連合サーバーのサーバーオブジェクトです。



解説

- 上記のコマンドで定義されるレプリケーション環境です。



サンプル

```
set run_script now stop on sql error off;
```

ソース・サーバーにNONIBMを指定します。
ここでしているのは、連合サーバーのサーバーオブジェクトです。

```
set server capture to db srcdb nonibm server ora10g id db2admin password "db2admin";
set server control to db tgt1db id db2admin password "db2admin";
set server target to db tgt1db id db2admin password "db2admin";
```

```
#drop control tables on capture server nonibm schema scott;
create control tables for capture server in nonibm schema scott;
```

Oracle上に作成するコントロール表のスキーマを指定します。

```
drop control tables on apply control server;
create control tables for apply control server;
```

ソースサーバー(連合サーバー)に作成されたニックネームを指定します。

```
#drop registration (emp);
create registration (emp) differential refresh condensed off;
```

```
#drop subscription set setname emp_set applyqual emp;
create subscription set setname emp_set applyqual emp activate yes timing interval 2
nonibm source server ora10g;
```

ソースがora10g上にあることを指定します。

```
#drop member from setname emp_set applyqual emp source emp target ttemp;
create member in setname emp_set applyqual emp activate yes source emp keys (empno +);
```

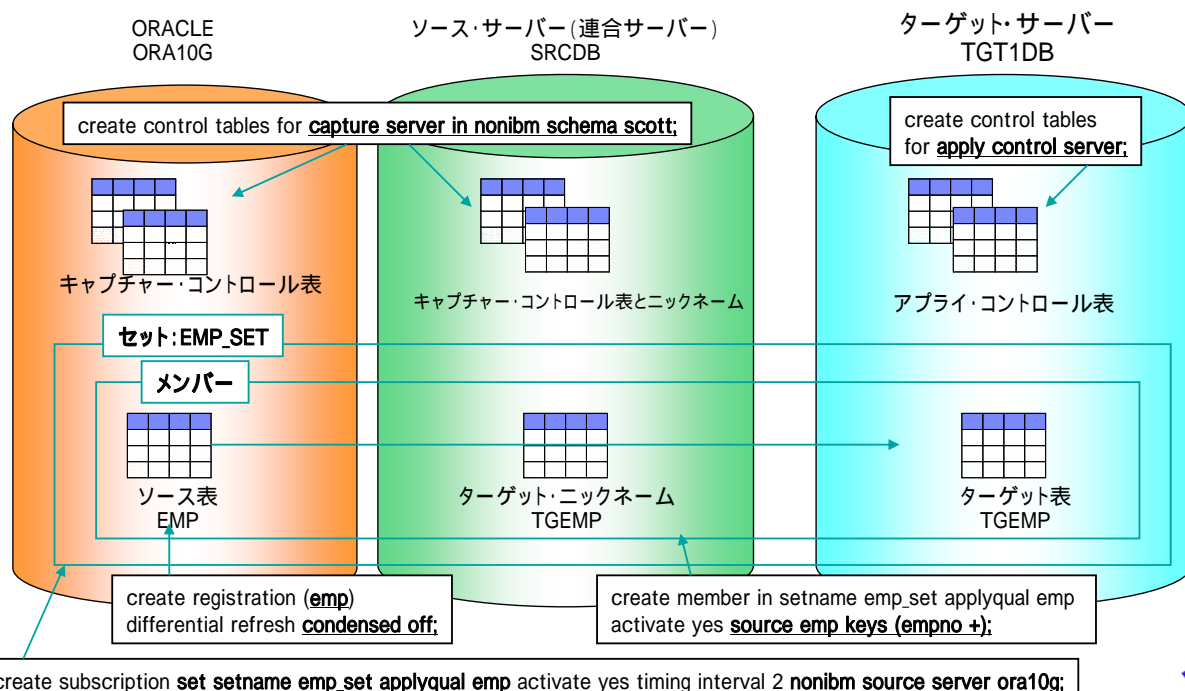
注意: IBMSNAP.SUBS.SET表のTGT_CAPTURE_SCHEMAにスキーマ名が入ります。
レプリケーションは成功します(動作上問題ありません)が、フルリフレッシュ時にエラーが出力されます。
気になる場合は定額後、この値をnullに更新します。



DB2 Universal Database

解説

- 上記のコマンドで定義されるレプリケーション環境です。



DB2 Universal Database