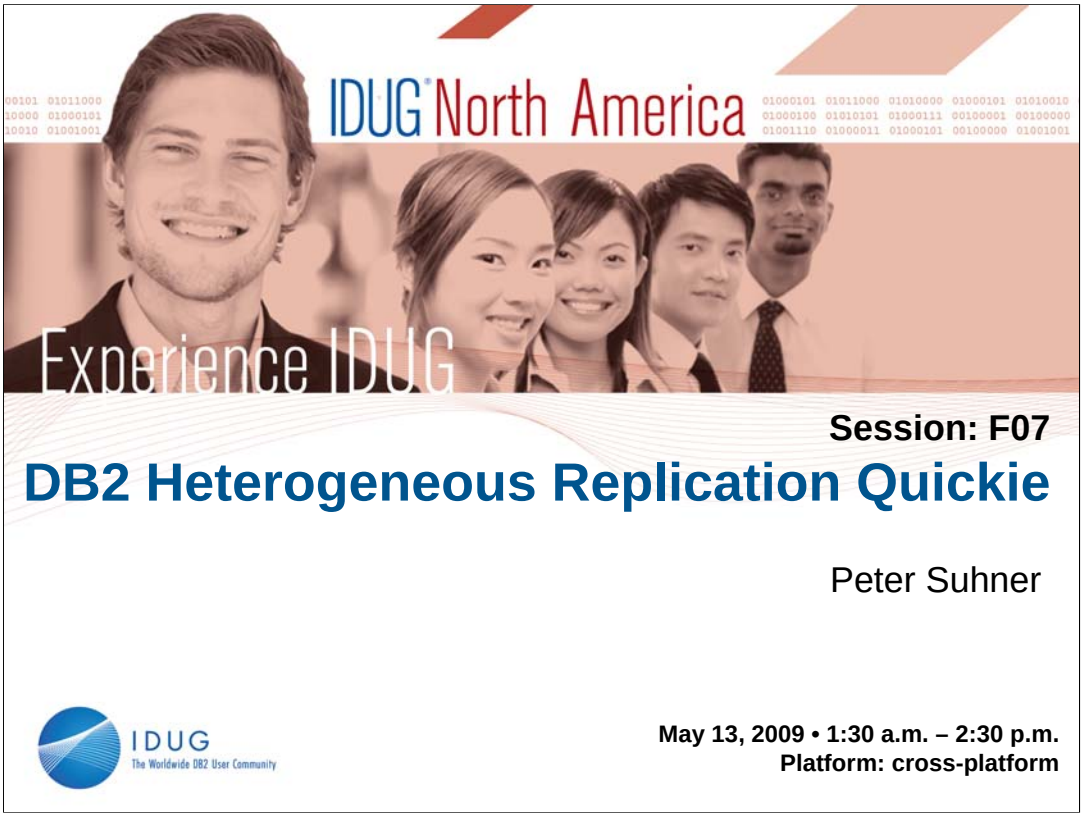


This presentation is a follow-up to "DB2 LUW Replication Quickie". It extends the overview of the DB2 Replication feature to a heterogeneous environment by adding information and examples towards DB2 z/OS and Oracle resources.

Starting with an overview of DataPropagator architecture and components, it will lead through the objects and definitions that are required to make heterogeneous replication work.

Basic information on performance and monitoring considerationas are mentioned as well as links to documentation and RedBooks.

Things are easy if you know…

- DB2 LUW Replication
  - Covered in first part of presentation
  - Short recap here
- DB2 LUW Federation
  - Quick overview follows here

To follow the contents of this presentation, it is advisable that you have a basic understanding of the mechanisms, technology and processes of the DB2 LUW Replication implementation.

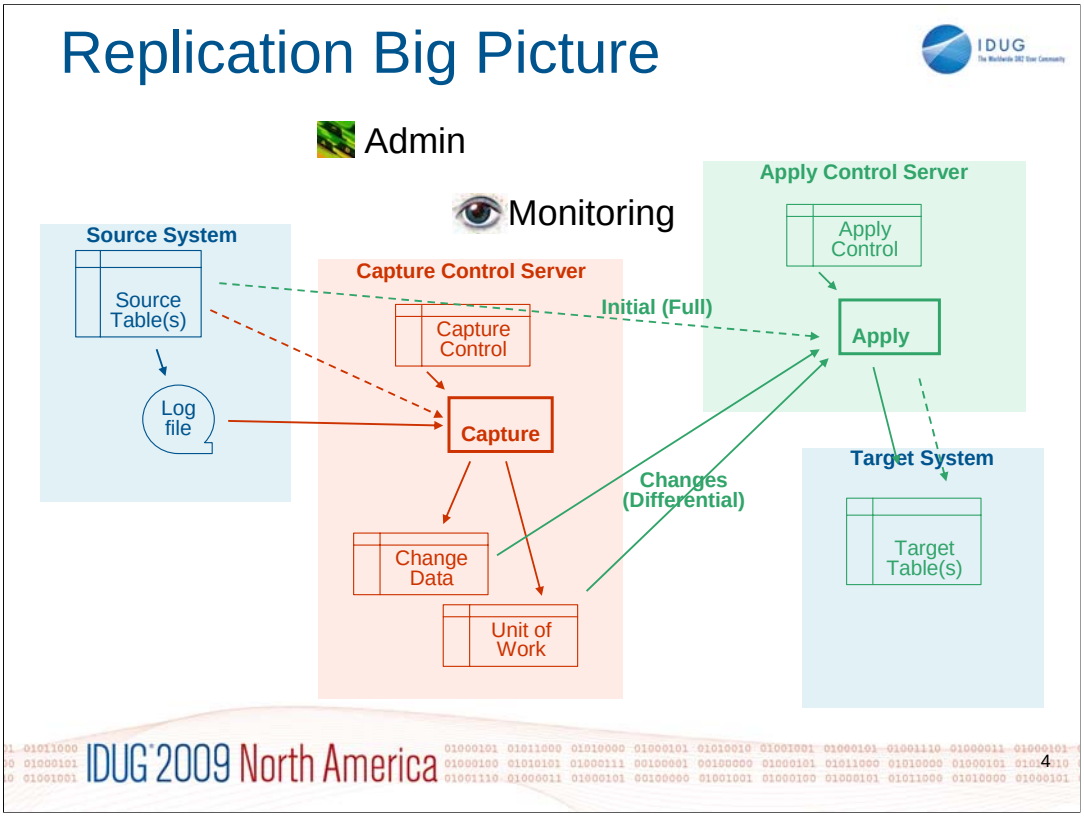
DB2 Federation technology is a requirement for Heterogeneous Replication, but only covered very superficially this presentation. It is expected that you have the required knowledge.

Agenda

- Replication Overview
- Federation Overview
- The z/OS view
- The Oracle view
- General issues and remarks
- Common Replication Scenarios

Listed on this slide are the major topics of this presentation.

Note: Some slides are probably not shown during the presentation, but are available in your proceedings. These slides supply additional information where we think this might be useful. *Hidden slides are not necessary to follow the sequence of information provided here.*



This slide shows the basic components of the replication mechanism in a very simplified way.

The blue part shows the relevant parts of your source and target databases: Any changes to the data tables will be written to a log file by the DB2 log writer processes.

The red part shows the capture mechanism. It basically consists of a log file reader which grabs all relevant data from the DB2 log file and – according to the definitions in the capture control tables – writes the data changes to respective change data and UOW tables (a set of intermediate DB2 tables). UOW information is stored to ensure that the captured data may be applied consistently later on.

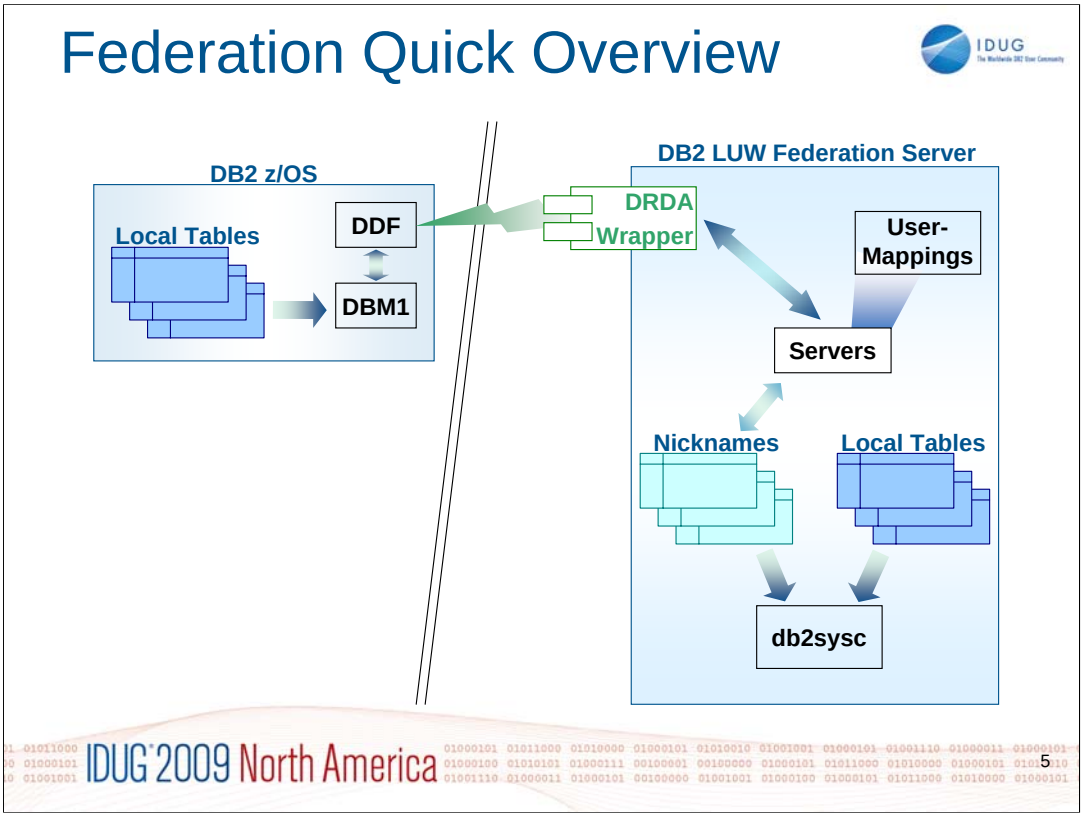
If the source database is a non-DB2 product, the Capture process usually can not refer to the source system's log file. In this case, data capture is implemented by other means – typically update triggers or some manually implemented simulation. But in any case, changes to the source table(s) are copied to a set of staging tables (Change Data) in a consistent manner.

The green part shows the apply mechanism. This process will apply data changes to the defined target tables (again according to the definitions in the apply control tables). Initial (full) replication will directly read the current contents of the source table whereas any subsequent changes will be applied from the change data tables of the capture system. The apply process will do this along the units of work as specified by the capture process.

As the target tables are also part of a (DB2) database, the changes of the apply process will be treated, controlled and logged as any other DML requests.

Capture and Apply processes are independent of each other. They may run continuously or triggered via schedule or events.

If source and/or target database are non-DB2 products, the connectivity to these resources is implemented through federation. Apply (and some parts of Capture) will then run against Nicknames.



This information is spread over two slides to provide enough space for additional notes on all the objects that exist for federation.

To set up Heterogeneous Replication (i.e. replication from and to data sources other than DB2 LUW), the concept of Federation is inevitable. Federation will not be covered in detail in this presentation. However, the following two slides will give you an overview on how the whole mechanism of Federation allows you to unite data from different locations, products and types of data and access them on the level of SQL statements.

Enabling Federation

Federation support comes with certain DB2 Server versions. To enable it, certain specific objects need to be created to support connections to other systems like DB2 z/OS and Oracle. Before any of these objects can be created, ensure you have set the DBM parameter "FEDERATION=YES" (non-dynamic, restart instance after altering). As a next step, catalog the remote node and database. Example:

db2> catalog tcpip node MF1TST remote DNS-ALIAS server PORT

db2> catalog database MF1TST as MF1TST at node MF1TST [authentication DCS]

db2> catalog dcs database MF1TST as MF1TST_01_000 (required if target DB name longer than 8 characters)

Wrapper

Wrappers are not only objects defined within the DB2 Catalog; they refer to specific binaries which will take care of adapting requests to the target system's transport layer and SQL dialect. A whole variety of wrappers for different products exist, including not only any RDBMS of a certain relevance (Oracle, MS SQL Server, Informix, mySQL, etc.) also non-relational data like IMS DB, XML, Excel spreadsheets and many more.

The DRDA wrapper comes with DB2 ESE. It can be made available with a simple SQL statement: CREATE WRAPPER "DRDA". All other relational and non-relational wrappers are part of Infosphere Information Integration (former WebSphere, former RelationalConnect/DataJoiner, etc.). This product is available at separate cost. Examples for Wrapper creation:

db2> create wrapper DRDA

db2> create wrapper ORAC1 library 'libdb2net8.so'

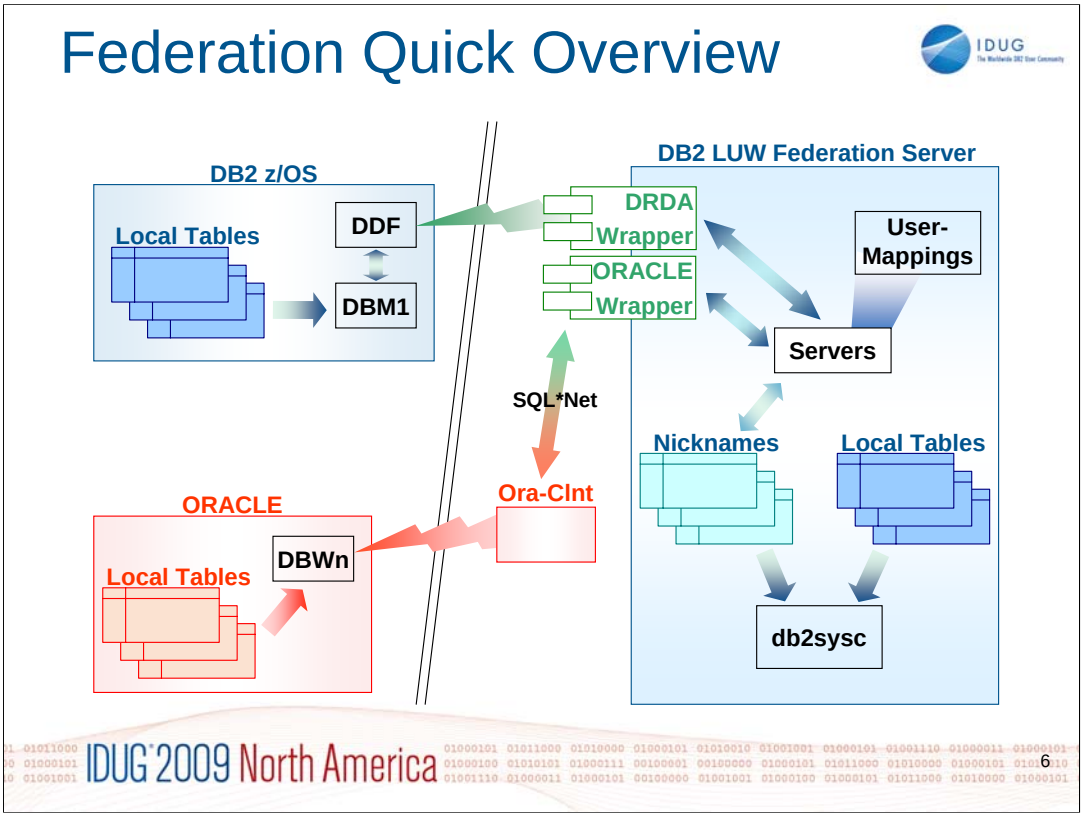
Server

Each remote data server requires a respective server definition. These definitions are stored in the catalog and reference the wrapper type that is required for the target server data source. Example for Server creation:

db2> create server "MF1TST" type DB2/ZOS version '9' wrapper "DRDA" authorization "myUser" password "myPWD" options (dbname 'MF1TST', node 'MF1TST')

db2> create server "ORACSRV1" TYPE ORACLE Version '10.2.0.4' WRAPPER "ORAC1" OPTIONS (COLLATING_SEQUENCE 'N', COMM_RATE '2', CPU_RATIO '1.000000', NODE 'ETDB1')

Notes continue on next slide...



Continuation from previous slide.

User Mapping

Mappings from the local to the remote user can be defined with User Mappings. They are per-data-server definitions of local and remote user id's including the remote password. This allows for completely transparent access to the remote data source. Passwords are automatically encrypted before they are stored in the catalog. Passwords can be updated, but there is no way to decipher and display the password once it has been stored. Example for User Mapping creation:

db2> create user mapping for LOCALUSER server "MF1TST" options (remote_authid 'REMOTEUSER' ,remote_password 'REMOTEPWD')

db2> create user mapping for db2inst3 server "ORACSRV1" OPTIONS (remote_authid 'donpedro', remote_password 'donpedro')

Nicknames

Nicknames can be defined for remote resources once the Wrapper, Server and User Mapping objects exist. They represent a DB2 version of the remote data source in a way that allows you to refer to it like to a local table. In terms of SQL DML, you can use them like any local table.

Nicknames typically use three-part names for the source definitions: SERVER.SCHEMA.TABLE (remember: "SERVER" depicts the target database -> see Server definitions above). Example for Nickname creation:

db2> create nickname ZOS_TST1 for MFTST1.MFTST1.TST1

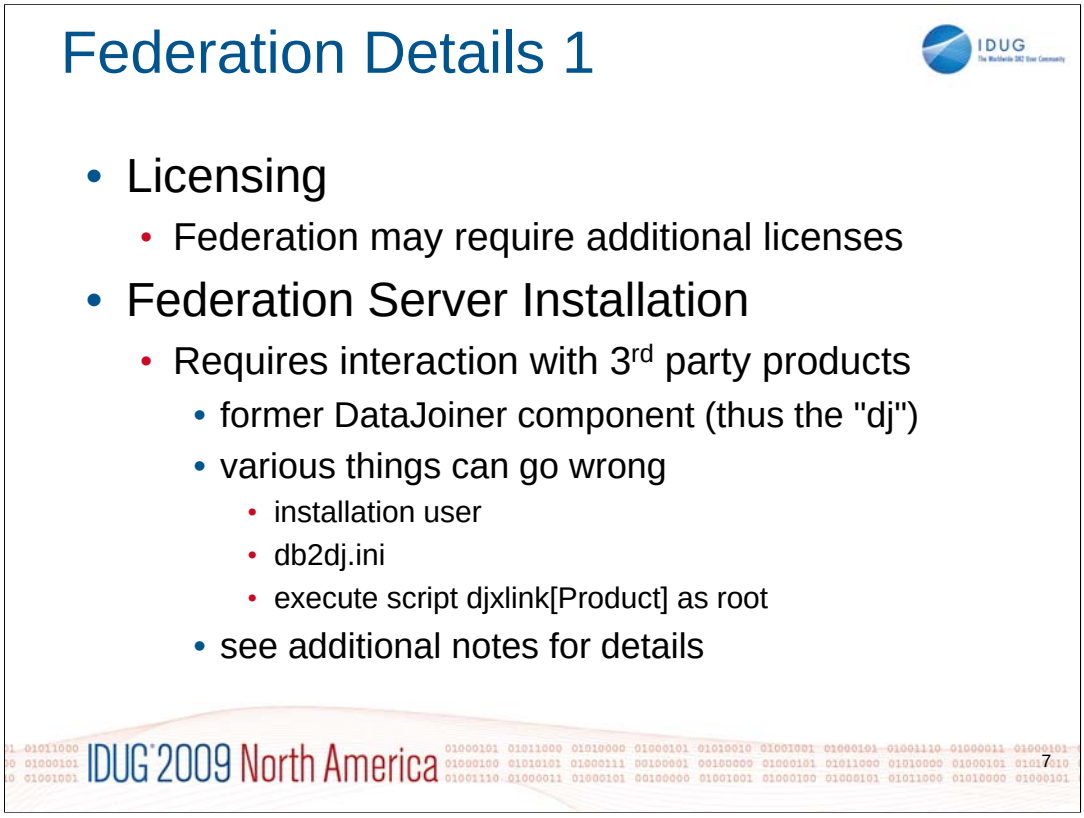
db2> create nickname ORA_TST1 for ORACSRV1.DONPEDRO.TST1

Product specific client software

As DB2 LUW server versions come with a local client, direct access to remote DB2 resources needs no further installations. For products other than DB2, the respective client software needs to be installed on the server where the DB2 Federation Server resides. Relevant libraries must be made available to the DB2 Federation Server by means of extending the respective path variables.

Disclaimer

Please note that IBM repackages it's software from time to time. As a result, some of the product options (like Federation, Compression, Encryption, Partitioning – just to have mentioned a few of them) may become part of the base product or may require separate licensing, depending on the product version.



Federation Server Installation

Federation requires interaction with 3rd party products. The respective definitions and product libraries must be made known to DB2.

Things that have gone wrong with DB2 9.5 under Solaris:

- Installation User should be root
  - Ensure the environment variable "USER=root" is set
    - Otherwise, the installation routine "iisetup" will not know the user and will fail
- db2dj.ini is used in Unix environments (not for Windows). It contains environment variables for 3rd party tools like ODBC- and Oracle-Client, etc.
  - The file is typically **not created** initially unless the installation user's environment has been set up completely for all of these 3rd party tools
  - The file should be created under ${INSTANCE_USER_HOME}/sqllib/cfg
- db2 registry variables
  - You should set the location of db2dj.ini with "db2set DB2_DJ_INI=${INSTANCE_USER_HOME}/sqllib/cfg/db2dj.ini" (replace variable by physical path)
  - Optionally, you can set the variable DB2_DJ_COMM=[library-name] to preload the interface libraries for 3rd party tools during DB2 instance startup
    - This will save some time during the first access
- Link libraries for 3rd party products
  - For most 3rd party products (like Oracle, SQLServer, Informix, etc.), two libraries are already available and a third one needs to be link edited
    - library names are "libdb2[ConnectionType][Suffix].[extension]"
      - the two existing libraries have a) no suffix and b) a suffix of "U" (e.g. libdb2net8.so and libdb2net8U.so for Oracle under Solaris)
      - the library that must be created has a suffix of "F" (e.g. libdb2net8F.so for Oracle under Solaris)
    - The additional library is link edited by executing a program "djxlink[ProductName]" (e.g. "djxlinkOracle" – without any additional parameters)
      - you must be root
      - db2dj.ini must contain the required environment variables for the 3rd party product
      - the newly created library will be placed in DB2's "lib64/" subdirectory
- Restart DB2 Federation Server after installing additional 3rd party product interfaces

Federation Details 2

- Install desired Wrappers
  - relational
  - non-relational
  - Stop/Restart instances after installation
    - db2 library path is updated
- Security Definitions on target server
  - Users in User Mappings need proper authority
  - Typically technical UserIDs
  - z/OS: check remote connection authorisation

Install Wrappers

- Ensure you have the correct license to use the wrappers
- Download the wrappers
  - relational and non-relational wrappers come in different packages
  - they may come with your Federation Server product (IBM Infosphere Information Integration)
  - they may be provided as separately downloadable add-ons for your DB2 ESE license
- Stop the db2 instance(s) after the installation (db2 library path needs to be updated during installation)
- Install the wrappers (db2_install for each of the wrapper packages)
- Restart the db2 instance(s)

Security Definitions

You must ensure that the userid/password combination used to access the remote DB2 (or non-DB2 system) is granted the required access rights on the target system.

Such definitions are possible on certain types of wrappers as well as with the UserMappings. You would typically prefer technical userIDs with non-expiring passwords over personal user accounts.

For z/OS also ensure that the specified users are allowed to connect to the system from remote.

Now that we know DB2 LUW Replication and just have gone through a Federation overview, the it's time for some details on Capture in a heterogeneous world.

Third Party Capture Basics

- **NO** Log Reader (obviously)
  - True for any non-DB2 Capture
- Alternatives:
  - **Capture Triggers**
    - DB2 Capture can generate them
    - Fired on any modification of source table data
  - **Capture Simulation** (not covered here)
    - populate staging tables with your own process
    - DB2 expects consistent data in CCD tables
    - No automatic pruning available

While DB2 Capture is based on a process that reads the source system's log files, this is not as easily achieved with 3rd party products. Of course, any RDBMS system writes log files, but their format and the internal log file processing varies heavily. Therefore an alternative is obviously inevitable.

Reading the log files for a Capture process is by far the most resource-saving way of capturing data. Any alternative is less efficient, but a variety of possibilities is available.

Capture Triggers can be generated automatically by DB2 primarily for 3rd party RDBMS. This solution implements triggers on the source table, which are fired on any update and will provide the functionality of reflecting the update in a different table which then serves as staging table.

But you can also – quite easily – **simulate the capture process** and provide the staging data by any other means. This could be Load/Insert from a file which was exported on the source side or any similar mechanism. Whenever you provide updates to the staging table(s), your Capture Simulation process should also update the respective Capture Control tables to let the Apply process know that new data is ready.

It is important to know that DB2 Data Propagation always expects to find consistent data in such staging tables. This is why such tables are called CCD (Consistent Changed Data). The Apply Process running on CCD tables is slightly different from what it looks like on standard CD tables (which were populated by DB2 Capture).

Also note that the responsibility for pruning of the CCD tables is entirely in your own responsibility. If you simulate DB2 Capture, the Capture process is not running and as a result, automatic pruning can not be provided.

Third Party Apply Basics

- DB2 Apply identical to LUW
  - see first part of this presentation
- Apply works on Nicknames
  - Real source tables are transparent to Apply
  - Apply **from** 3rd party resources
    - Full refresh refers to Source Table Nickname
    - Differential refresh refers to CCD Table Nickname
  - Apply **to** 3rd party resources
    - Apply refers to Target Table Nickname

Once the captured data is available in a set of staging tables within DB2, the Apply process runs about identical to a purely DB2-internal replication environment. For these details, please refer to the first part of this presentation ("DB2 LUW Replication Quickie").

Instead of referring to the physical source and target tables, all Apply definitions are defined towards Nicknames.

Nicknames for Source and Target tables must be defined manually.

Nicknames for CCD tables will be created automatically during Capture definition.

Let's now explore replication in the z/OS world.

DB2 z/OS DPropR 1

- **DataPropagator** for Replication
  - DPropR, aka Replication Server for z/OS, etc.
  - separate product, additional license fee
- z/OS **USS** application (as of v8.1)
  - run in USS or as Started Tasks in z/OS via JCL
    - Output is written to SYSPRINT
- Administration possible with db2rc GUI
  - DAS required

First of all, DB2 z/OS base product does not provide any replication feature. Instead, Replication functionality is implemented through a separately distributed (and licensed) product called "DataPropagator", or "DPropR" in short. The terms "Replication Server for z/OS" are also found, as well as "WebSphere Information Integration Change Data Capture for z/OS" and probably a few more. Product names seem to vary in this area but it looks like it was all pretty much the same.

DataPropagator for DB2 z/OS is an application which runs on the USS side of z/OS. Nevertheless, the processes can also be run as z/OS Started Tasks via JCL. Several preconditions must be met, path and user settings on the USS side are required. For details on this, refer to the book listed at the end of this page.

If run as Started Tasks in native z/OS, Capture and Apply write their output to SYSPRINT. The format of the output is identical to what you know from DB2 LUW Replication.

You can administrate DataPropagator on z/OS with the respective GUI (db2rc) that comes with the DB2 client. As with DB2 LUW, a DAS server must be running to allow the client to connect to the mainframe and find the targeted DB2 subsystem(s).

DB2 z/OS DPropR 2

- Similar to DB2 LUW Replication
  - Capture, Apply, Monitor processes and Trace
  - Systems Automation in analogy to DB2 LUW
- Data Sharing Group context
  - One Capture per Group and Control Table Schema
    - Multiple Capture processes with multiple Schemata
- Capture needs APF authorized library
  - ASNAPLX and ASNCAP read DB2 Log IFI

The product itself is quite similar to the replication feature of DB2 LUW and supplies about the same processes. As a result, the handling of the environment to what concerns performance issues, automation, etc. can be adapted from what was presented in the "DB2 LUW Replication Quickie".

If you run a Data Sharing system, ensure that Capture and Apply are installed on all systems of the data sharing group. The DPropR processes run in the context of the Data Sharing Group, not of a single member. You can run more than one Capture process in a Data Sharing Group – they can read the log files in parallel (even though it might slow down the read throughput). But every Capture process runs on it's own schema for the Capture Control Tables and you therefore need to multiply these Control Tables within different schemata.

Also remember that Capture reads DB2's log files. As a result, you need to install DataPropagator on all systems of a data sharing group. Plus two binaries of Capture (ASNAPLX and ASNCAP) must be placed in an APF authorized library. ASNAPLX is only required in a Sysplex where you run DB2 in Data Sharing mode.

For more details on running Capture and Apply on z/OS, see "A Practical Guide to DB2 UDB Data Replication V8" (SG24-6828-00), pg. 256ff and 275ff

DB2 z/OS Capture

- DB2 z/OS Capture threads
  - HOLDL – schema enqueue
  - ADMIN – statistics and traces
  - PRUNE – data cleanup
  - WORKER – log reader/table writer
  - A control thread (starts the other ones)
- Specific privileges to run Capture, on
  - Data (replication tables)
  - DB2 Subsys (TRACE, MONITOR1, MONITOR2)

Capture on z/OS has five threads:

HOLDL Ensures that only one Capture is running for a specific schema by locking the IBMSNAP_CAPENQ table

ADMIN Writes statistics to IBMSNAP_CAPMON and messages to the IBMSNAP_CAPTRACE tables

PRUNE Stays in "rest" state and is activated on *prune_interval* (with autoprune) or when prune is activated manually. The thread prunes the CD table(s) plus UOW, SIGNAL, CAPTRACE and CAPMON tables

WORKER Read the DB2 logs and writes changes into memory. Then writes committed changes from memory to CD tables and updates UOW table with commit information

The Control thread controls the other four threads, i.e. it wakes them up when it's time to become active, etc.

The user under which the Capture process runs, requires various privileges to ensure all resources may be accessed correctly. For DB2 z/OS this includes:

- SELECT, UPDATE, INSERT, DELETE for all replication tables
- SELECT on Catalog
- EXECUTE on capture packages
- TRACE privilege
- MONITOR1 and MONITOR2 privileges
- Write access to capture path directory (for diagnostic files)

DB2 z/OS Capture Job


```
//P1CAPTUR JOB (DB2,M1S),'14 DB2/SYS',MSGCLASS=S,MSGLEVEL=(1,1),
//     PRTY=12,USER=MYUSER,REGION=400M,CLASS=J,SCHENV=SCHENV12
//*******************************************************
//ASNCAP   EXEC PGM=ASNCAP,
// PARM='capture_server=P0DB capture_schema=ASN startmode=warmns'
//STEPLIB  DD  DISP=SHR,DSN=SYS1.SCEERUN
//         DD  DISP=SHR,DSN=SYS1.SCLBDLL
//         DD  DISP=SHR,DSN=SYS1.DSNP0DB.SDSNEXIT
//CAPSPILL DD  DSN=&&CAPSPL,DISP=(NEW,DELETE,DELETE),
//       UNIT=VIO,SPACE=(CYL,(50,70)),DCB=(RECFM=VB,BLKSIZE=6404)
//MSGS  DD PATH='/usr/lpp/db2repl_09_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD   DUMMY
//SYSTERM  DD   SYSOUT=*
//SYSUDUMP DD   DUMMY
//SYSPRINT DD   SYSOUT=*
```

Capspill: if TRX size exceeds memory

Example of running V8 Capture:

Start Capture COLD in subsystem P0DB for Capture schema ASN.

Capture_server is a mandatory parameter for Capture. The capture_server parameter can be specified as either a positional parameter or a key word parameter. Positional parameters are the old V7 syntax. Key word parameters are the new V8 syntax. Positional parameters are for compatibility only. Key word parameters are encouraged.

The capture_schema parameter is also mandatory if your capture_schema is not ASN. The capture_schema parameter must be specified as a key word parameter.

Additional Capture key word parameters are: logreuse,logstdout,autostop - all these can be specified in the asn.ibmsnap_capparms table.

The CAPTURE_PATH=//MYPATH keyword parameter can be specifed in the asn.ibmsnap_capparms table. This example shows how to specify the Capture_path in JCL. The Capture log file will be written to the z/OS dataset MYUSER.MYPATH.T0DB.ASN.CAP.LOG where 'MYUSER' is the user who submits the Capture JOB.

If you want the Capture log z/OS dataset to have a different high level qualifier, you can specify CAPTURE_PATH=//"OEUSR01' and Capture will write its log file to OEUSR01.P0DB.ASN.CAP.LOG. Although the Capture program can be run from JCL, it is a USS program and uses HFS as its default file system. If you do not specify the Capture_path parameter, the Capture program will write its log file to the home directory of the user who submits the Capture JOB.

You can include a MSGS DD statement that identifies the message catalog HFS path in its PATH parameter. If you do not include a MSGS DD statement, the message catalog location is determined from the NLSPATH environment variable. Your NLSPATH environment variable must include the %N special variable, which holds the name of the message catalog. It can also optionally include the %L special variable (which holds the locale-specific directory that contains the message catalog). NLSPATH example: NLSPATH=/u/OEUSR01/usr/lpp/db2repl_08_01/msg/%L/%N

The Capture program gets its time zone information from the TZ environment variable. The NLSPATH and TZ values should be set in either the .profile file located in the user's home directory (user who submits the Capture JOB) or in the /etc/profile file (system-wide profile for shell users). If an NLSPATH value is not set in either the user's .profile file or the /etc/profile file, the default NLSPATH is "/usr/lib/nls/msg/%L/%N".

When Capture exceeds the memory_limit for transactions (as specified in IBMSNAP_CAPPARMS table or in PARM in ASNCAP EXEC statement), it will spill to VIO with a default value. You can override the default value with the CAPSPILL DD statement. Please allocate sufficient space for the largest transaction. You can use UNIT=SYSDA instead of VIO.

DB2 z/OS Apply

- DB2 z/OS Apply threads
  - HOLDL – enqueue on apply qualifier
  - WORKER – table reader, spill and table writer
  - A Control thread (starts the other ones)
- Spill file
  - ASNASPL DD ..., write to SYSDA
- Apply cycle is identical to DB2 LUW

Apply on z/OS has three threads:

HOLDL Serialization: Ensures that only one Apply is running for a specific apply qualifier

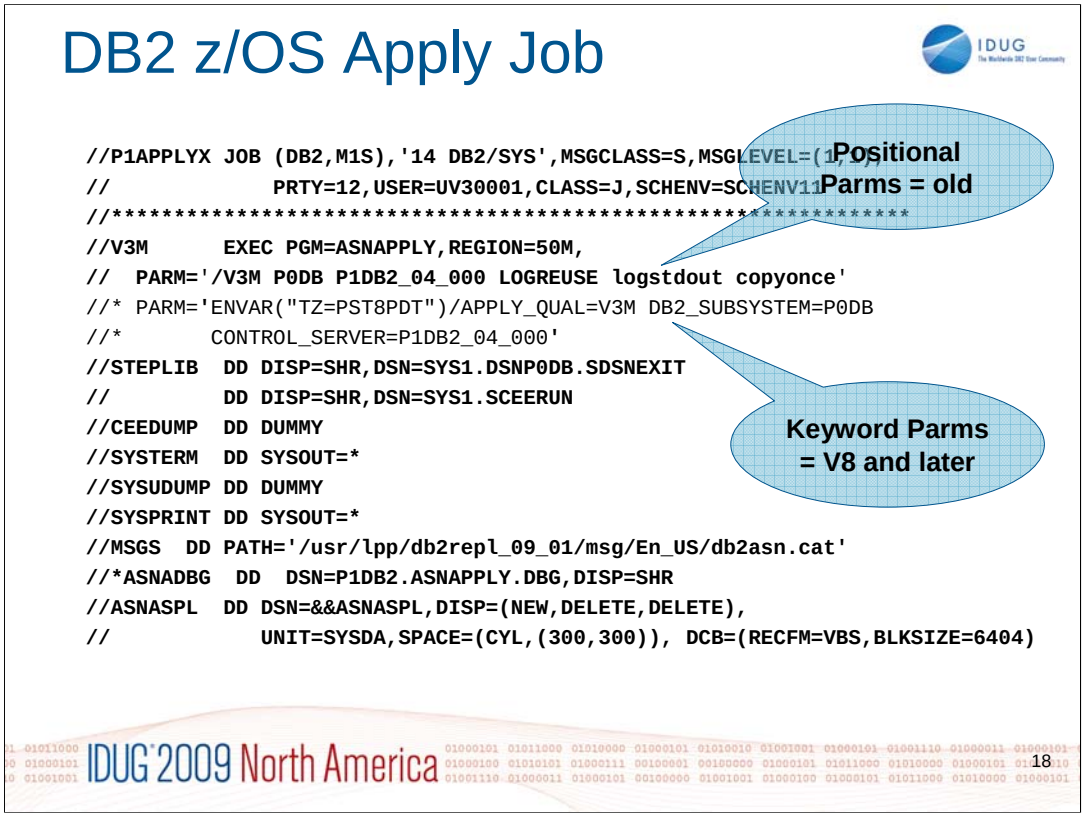
WORKER Read the CD and UOW tables and applies changes to target tables. Updates control tables of both, Capture and Apply Control Servers (for pruning, statistics, etc.).

The Admin thread controls the other two, i.e. it wakes them up when it's time to become active, etc.

Before Apply writes to the target table(s), it will need to create a spill file of the data it reads from the staging tables. For details on this, refer to the first part of this presentation. When started from JCL, the spill file should be defined as SYSDA using DD statement "ASNASPL". See example on next slide.

Anything else is about identical to DB2 LUW Replication.

DB2 z/OS Apply Job



This is an example of running V8/V9 Apply:

3 parameters are mandatory for Apply: Apply qualifier, DB2 subsystem, and Control server location. These mandatory parameters can be specified as either positional parameters or key word parameters (Positional parameters are the old V7 syntax - keyword parameters are encouraged).

Positional parameters are the old V7 syntax. Key word parameters are the new V8 syntax. In this example, the positional parameters are Apply qualifier (V3M), Subsystem (P0DB), and Control server location (P0DB).

Key word parameters are: APPLY_QUAL=V3M DB2_SUBSYSTEM=P0DB control_server=P1DB2_04_000

Additional Apply key word parameters are: logreuse,logstdout,copyonce (all of these can be specified in asn.ibmsnap_appparms table)

If you want the Apply log z/OS dataset to have a different high level qualifier, you have to specify APPLY_PATH=//"MYHLQ' and Apply will write its log file to MYHLQ.[DB2_SUBSYSTEM].[APPLY_QUAL].APP.LOG. (which will resolve into MYHLQ.P0DB.V3M.APP.LOG in this example).

Although the Apply program can be run from JCL, it is a USS program and uses HFS as its default file system. If you do not specify the Apply_path parameter, the Apply program will write its log file to the home directory of the user who submits the Apply JOB.

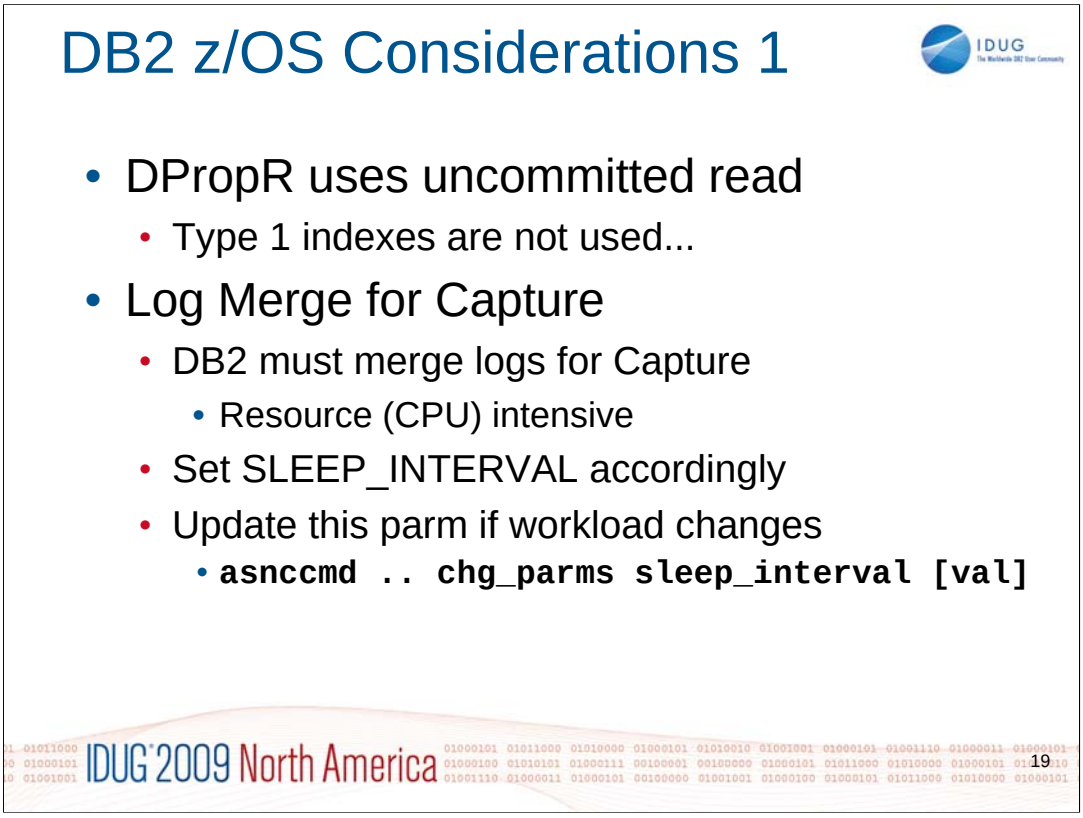
You can include a MSGS DD statement that identifies the message catalog HFS path in its PATH parameter. If you do not include a MSGS DD statement, the message catalog location is determined from the NLSPATH environment variable. For more details, see the Capture Job example.

For debugging, you can add a ASNADBG DD statement.

Also note the spill file which the Apply process requires to write the captured data to before it is inserted into the target table.

Output is written to SYSPRINT and will look similar to what Apply produces on the LUW platform. Example:

```
2009-02-26-23.31.50.460193 ASN0552E  "Apply" :  "V3I" : "WorkerThread" : The program encountered an
   SQL error. The server name is "UC000550". The SQL request is "CONNECT". The table name is "N/A".
   The SQLCODE is "-1040". The SQLSTATE is "57030". The SQLERRMC is "". The SQLERRP is "SQLESUBC".
2009-02-26-23.31.50.460193 ASN8999D  "Apply" : "V3I" : "WorkerThread" :
  DSNT408I SQLCODE =  -1040, SQLSTATE = 57030, RESOURCE NOT AVAILABLE OR
           OPERATOR INTERVENTION FROM DB2 UDB for AIX, Linux, HP-UX, Sun,
           and Windows
2009-02-26-23.31.50.474796 ASN0530E  "Apply" :  "V3I" : "WorkerThread" : The program could not
   connect to database "UC000550" with USERID "N/A" . The SQLCODE is "-1040".
```



Uncommitted read for DataPropagator

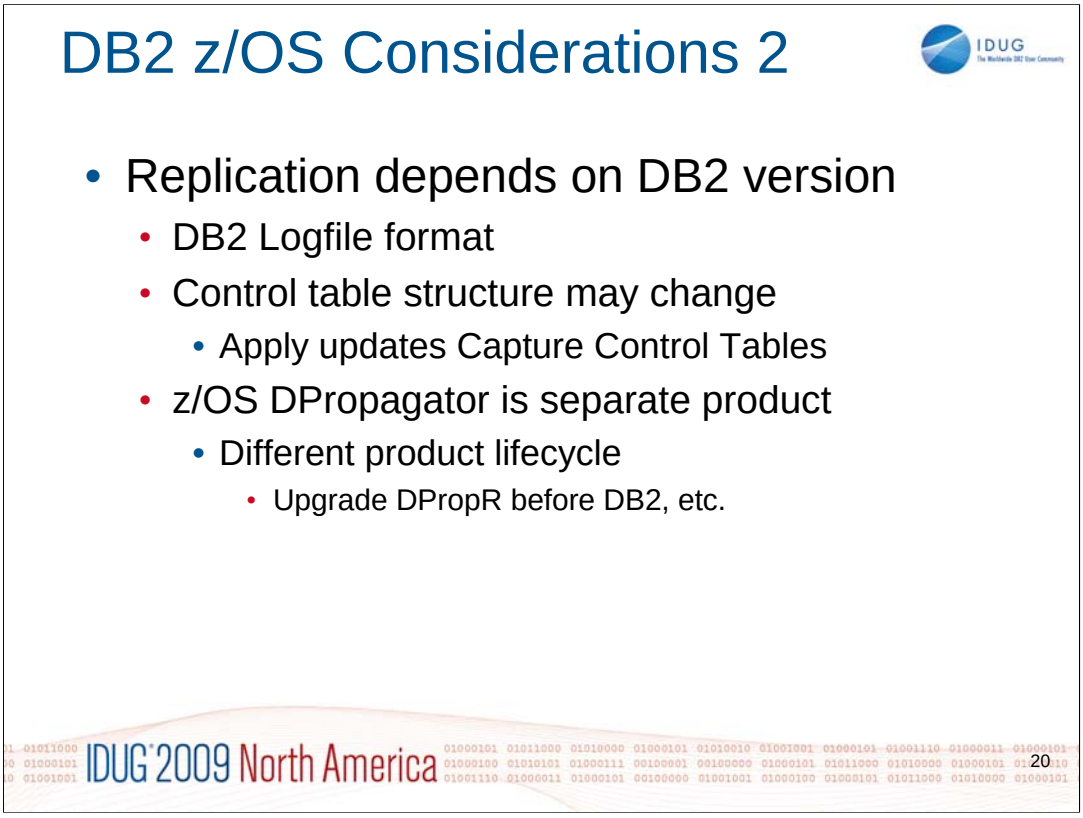
It is good to know that DataPropagator on z/OS makes use of uncommitted read. Apart from the fact that this is important information for full refresh in terms of data consistency, please ensure that all of your indexes on Replication candidate tables are Type 2 indexes.

Log merge

With low update activity on the captured tables, the Capture process can become current and goes to sleep. When it becomes active again and requests logs just to see whether there is new data to be processed, DB2 must merge the logs of all members of the data sharing group. This task is resource (mainly CPU) intensive.

You can optimize this resource consumption if you check hte activity and set the SLEEP_INTERVAL for Capture accordingly (either as a start parameter or through the respective value in IBMSNAP_CAPPARMS).

If your workload changes (e.g. for your batch window), you might want to update this parameter. This can be done dynamically by the aid of "asnccmd" command.



DB2 Version changes

With DB2 LUW, the replication feature is part of the product and therefore version dependencies (e.g. control table format or the log reader, which depends on the internal format of the log file contents) are not a big issue. Nevertheless, you need to be aware of possible issues when you start using heterogeneous replication or/and in case you use advanced features like Capture Simulation or Capture/Apply control from external 3rd party or self-written applications.

For any replication that involves different versions of DB2 (be it all LUW or including z/OS and/or iSeries versions), it is inevitable that you check the product versions for compatibility. Major issues are

- Difference between DB2 z/OS and DPropR (DB2 log format changes)
- Differences between Capture/Apply process versions and their Control Tables
    - Remember: Apply updates Capture Control Tables as well
        - e.g. Capture could be on z/OS with a different version than the various Apply processes you run in your LUW environment(s). Or vice versa.

Time for a refreshing sea breeze sponsored by Oracle...

Oracle Setup

- Set up Oracle client
  - Use the Oracle **Runtime** Client
    - Instant Client is missing library "xaondy.o"
- Test Oracle access for DB2 instance user
- Set Oracle environment variables for DB2
  - setup sqllib/cfg/db2dj.ini
  - set .profile stuff
  - run djxlinkOracle as root: creates libdb2net8F.so
- Test: try creating the Oracle Wrapper

Set up Oracle client

A fully fledged "Oracle Runtime Client" is required on the local server where the Federation Server is installed. The existing "Oracle Instant Client" will not suffice - it is missing specific libraries (e.g. "xaondy.o") which are required to link DB2's interface libraries.

Test Oracle access for DB2 instance user

Ensure that the DB2 Federation instance user is allowed to use the Oracle client. Connect to an existing Oracle database through the Oracle Client's own interface:

$ su – [DB2InstanceUser]

$ cd ${ORACLE_HOME}

$ sqlplus [UserId]/[Password]@[OracleDBName]

$ select count(*) from dual;

$ exit;

Set Oracle environment variables for DB2

- File db2dj.ini should exist in ${DB2InstanceOwnerHome}/sqllib/cfg
  - Create it if it does not exist
  - provide the required environment variables as listed in the example below
- run djxlinkOracle as root to ensure libdb2net8F.so is created
  - The script "djxlinkOracle" will create the required interface library "libdb2net8F.so"
  - Ensure that the created library is located in the correct path (DB2's lib64/ directory) and has the correct ownership to be accessed by the DB2 Instance User

You will recognize whether or not the interaction of DB2 with the Oracle client is possible if you try to create the Oracle Wrapper.

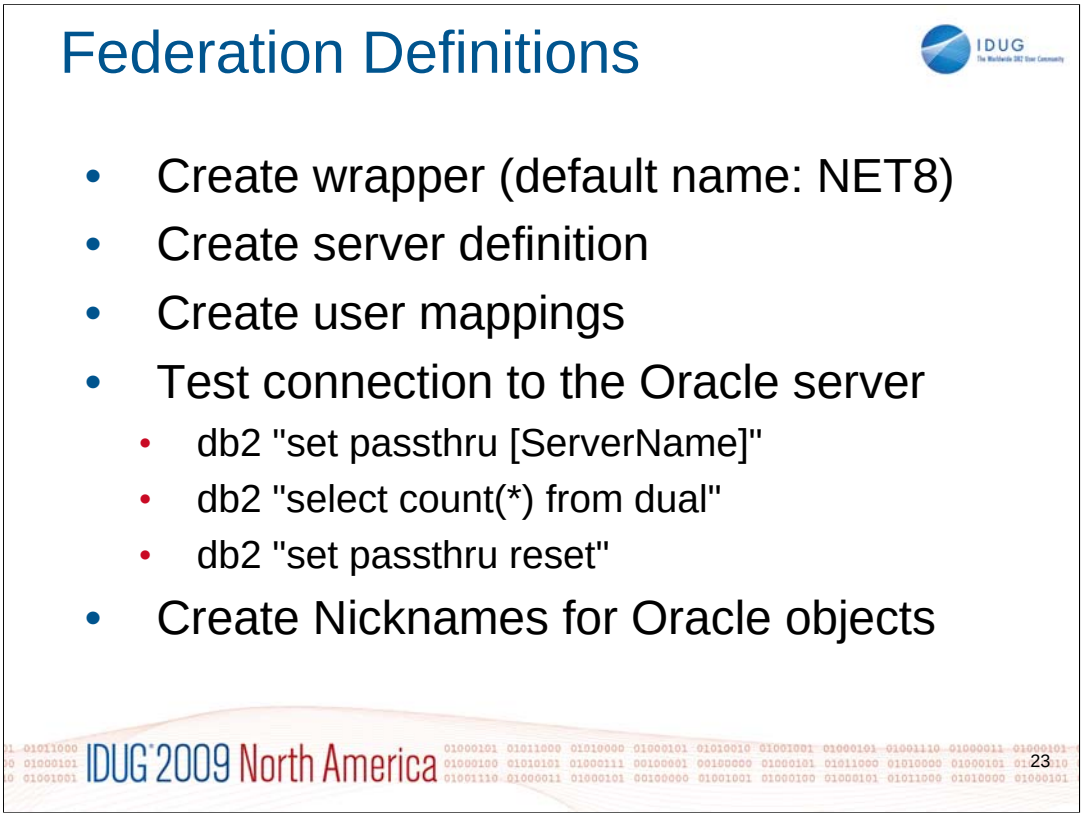
- Connect to your DB2 database and issue db2 "create wrapper NET8" (this is the default Oracle Wrapper)
  - If the Wrapper is created, the Oracle client is accessible from DB2
  - If the Wrapper can not be created, the error message typically says that the library "libdb2net8F.*" could not be found
    - This error message is also issued if the library is available, but something further down the road towards the Oracle Client went wrong
      - At this point: **Check db2diag.log for details!**
- If the Wrapper creation fails without meaningful messages in db2diag.log, extend the DB2 Instance User's .profile according to the example listed further down, then:
  - Stop the DB2 Instance
  - Source the updated .profile file or Logoff/Logon as Instance User
  - Start DB2 Instance
  - Try again

Example db2dj.ini for Oracle

```
ORACLE_HOME=/app/ora10/product/10.2.0.3.0_RT_Client
LIBPATH=/usr/db2i1/home/sqllib/lib:/usr/lib:/lib
TNS_ADMIN=/app/ora10/network/admin
ORACLE_OWNER=ora10
NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1
```

Example DB2 Instance User's .profile

```
export ORACLE_OWNER=ora10
export ORACLE_HOME=/app/ora10/product/10.2.0.3.0_RTCl
export ORACLE_SID=ORADB1
export TNS_ADMIN=/app/ora10/network/admin
export PATH=${PATH}:${ORACLE_HOME}/bin
```



Once you have created the wrapper (as described in the previous slide), you can continue with the other required federation objects.

Let's first see the simplest way:

db2> create wrapper NET8

And here's an example for a more detailed version of an Oracle Wrapper definition. For other third party products, you would use about the same command and just change the wrapper name and the library providing the required functionality for the interaction between DB2 and the third party product:

db2> create wrapper ORAC1 library 'libdb2net8.so'

Continue with the server definition:

db2> create server "ORACSRV1" TYPE ORACLE Version '10.2.0.4' WRAPPER "ORAC1" OPTIONS (COLLATING_SEQUENCE 'N', COMM_RATE '2', CPU_RATIO '1.000000', NODE 'ETDB1')

Now it is time for the User Mappings:

db2> create user mapping for db2inst3 server "ORACSRV1" OPTIONS (remote_authid 'donpedro', remote_password 'donpedro')

Important: Ensure that the userID/password combination used here has the expected access rights on the Oracle database

At this point we do have all required information to basically connect to the Oracle database for a primary test. DB2 Federation Server provides a mode called "PASSTHRU". When you set the current session to passthru mode for a given ServerName, your SQL statements are directly routed to the defined database in the target RDBMS, executed there and the result is passed back to your DB2 session.

An SQL statement you can use against every Oracle database is "select count(*) from dual". "DUAL" exists in every Oracle database, it is the equivalent to DB2's "SYSIBM.SYSDUMMY1". This test should return the value "1" (because DUAL also contains exactly one row). It ensures interconnectivity, basic access authority, ability to use scalar functions, etc.

Don't forget to reset the passthru mode after your test.

When you have received the correct result, you can start registering Nicknames for your Oracle datasources.

$> db2 "create nickname nick_oratab1 for ORASRV1.ORASCHEMA.ORATAB1"

Oracle Data Types

- DB2 ensures source/target **compatibility**
  - Length, Nullability, PK/Index attributes, etc
  - **Not necessarily optimal** mapping!
    - Issue “ALTER NICKNAME” to adapt data types
- Be careful with
  - Different Numeric and Date/Time formats
  - Certain data types not supported, e.g.
    - LONG (deprecated in Oracle 10g)
  - UDF (User defined data types)
  - Field procedures

-408 (not compatible)
-180 (string representation)

Data Types

Basically, Oracle has a more relaxed approach towards data types. The number of different data types is smaller than in DB2 and Oracle is able to treat and cast values in a more relaxed way than DB2 does (this is also true for character and numeric data conversions – Oracle allows you to compute with character strings if they are figures, etc.). There is no general "better" or "less sophisticated" or whatever. Both approaches have their pros and cons and it's just important to be aware of the differences.

Note also that there is no possibility to define data types or column names during Nickname creation for relational resources (however this is allowed for non-relational resources). A relational Nickname is simply created as a DB2 version of the object – attempting to define it as close as possible to the original non-DB2 data source. db2rc focuses on **compatibility** rather than optical presentation of the original data within DB2. DB2 will select a data type that is able to correctly store the source data type in terms of length, nullability, attributes for primary keys and unique indexes, etc.

This might not necessarily be the data type you expect to work with in DB2. Therefore it is good to know that you can alter an existing Nickname to change column attributes. This is particularly relevant – and sometimes required – for numeric and date/time formats.

Particular care needs to be taken with numeric and date formats, but also with user defined data types. There are certain data types that are not supported.

Also don’t forget about such “nice” little features like field procedures, etc.

Hint: DB2 LUW can display a table structure if you issue the command “describe table <table_name>”. Oracle knows exactly the same feature, but the command is a bit simpler: “describe <table_name>” – there you go.

Few more details on data types can be found in “Websphere Information Integration 9, SQL Replication Guide and Reference” (SC19-1030-00)

Oracle as a Source 1

- Defining Capture (with db2rc) creates:
  - DB2 Federation side
    - Control table (ASN.IBMSNAP_CAPSCHEMAS)
    - Nicknames
  - Oracle side
    - Control tables
    - Staging tables in Oracle database (CCD Tables)
  - Capture mechanism
    - Several Oracle objects (see next slide)
    - Generated by DB2

When you define Oracle as a source for data replication, a capturing mechanism must be implemented on the Oracle source database. DB2 is able to create all of the necessary Capture Control Tables; most of them reside on the Oracle database.

Please note that for Oracle, the terms "schema" and a "user" are synonymous. Therefore, if you want to use a separate schema for the Capture Control and Staging Tables, you must ensure that a respective user exists. It might make sense to implement a naming convention for such special schemata.

Apart from the Control Tables, a mechanism must be implemented to capture changes to the source tables. The standard approach for capturing data at third party relational data sources is the use of triggers which are fired when data modifications occur on the source table.

There are rumours that IBM has decided to support the same mechanism for Oracle as for DB2: with a certain level of DB2 LUW 9.5, an Oracle LogReader is said to become available. This approach would of course provide much better performance and less overhead. If this feature arrives has proven to run stable, I would clearly favour it over triggers.

The capture mechanism is described in more detail on the next slide.

Oracle as a Source 2

- Trigger based Capture mechanism
  - CCD table pruning supported
  - SIGNAL table available for Apply process
  - Various objects in Oracle database
    - Control Tables
    - CCD Table(s)
    - Triggers
    - Sequences

Capture Control Objects (General)

IBMSNAP_REGISTER Table with 3 indexes
IBMSNAP_REG_SYNCH Table with 1 index
IBMSNAP_PRUNCNTL Table with 4 indexes
IBMSNAP_PRUNE_SET Table with 1 index
IBMSNAP_SIGNAL Table (without indexes)
PRUNCNTL_TRIGGER Trigger for Prune Control
REG_SYNCH_TRIGGER Trigger for Synchronisation of Registrations
SIGNAL_TRIGGER Trigger for Signal table (used by Apply)

SGENERATOR001 Sequence for SyncValue generation of general objects

Capture Registration Objects (per Table)

CCDtab_name Staging table with structure of source table plus some additional columns, with one index on primary key
DCCDtab_name Trigger for Delete activities, will act on updates to source table and write record(s) to Staging table
ICCDtab_name Trigger for Insert activities, will act on updates to source table and write record(s) to Staging table
UCCDtab_name Trigger for Update activities, will act on updates to source table and write record(s) to Staging table
UPCCDtab_name Trigger to update CommitSequence column in CCD table

SGENERATOR002 Sequence for SyncValue generation of the Delete/Update/Insert trigger of the source table. For additional source tables, more Sequence Objects are created as required.

If you want to check the objects created by DB2 Capture, connect to your oracle database and issue "select * from user_objects;".

If you are interested in what actions the triggers perform, connect to your oracle database and issue "select trigger_name, trigger_body from user_triggers;". If you are in SQLPLUS, make sure you issue a "set long 2500" before to display all of the trigger definiton code.

This query can not be issued from within DB2 federation (through "set passthru [ServerName]") because the "long" datatype used by table column "user_triggers.trigger_body" is deprecated and not supported from within DB2.

Oracle as a Target

- Defining Subscription Sets
  - Oracle
    - Create target table in Oracle database
    - Define indexes for target table
  - DB2 Federation
    - Create Nickname for Oracle target table
  - DB2 Apply Control Server
    - Create Subscription Set based on Nickname

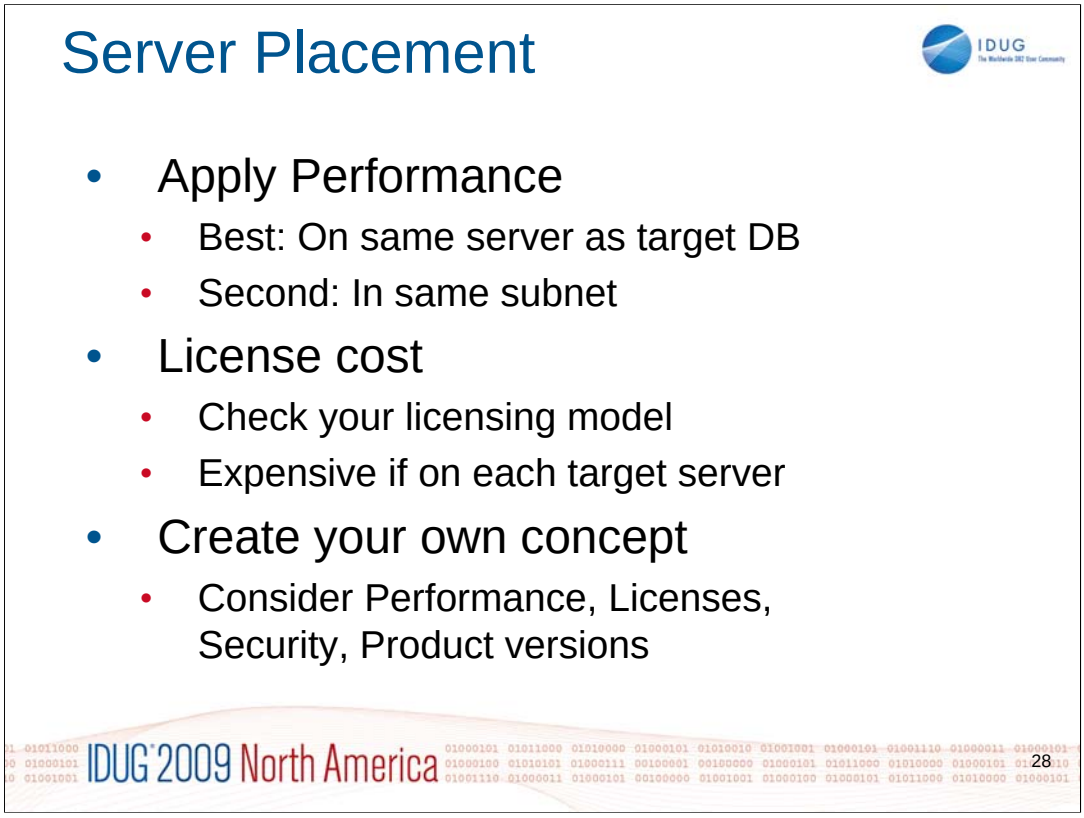
When you define Oracle as a target for data replication, all you need on the Oracle side is the target table. Also make sure that constraints and indexes are defined to suit your application's needs.

All of these resources should be defined before – as a next step – the Nickname is created on the Federation Server, because DB2 will consider these objects for it's decisions on the access path.

Defining the Subscription Set for the Oracle target table is identical to the definition of any DB2 target table and can be performed with 'db2rc' (DB2 Replication Center GUI Application, part of the DB2 Administration Client). The Nickname for the Oracle target table represents a DB2 facade which will can be used as any DB2 table.

In addition to the target Nickname, every Subscription Set definition requires a Registration, Capture and Apply Control Servers to act on.

You will also have to decide on where to run the Apply process – the next slide will cover these aspects.



Where to place the Apply process and the Federation Server

Remember that we are in a distributed environment. Remember also that you might have a server based licensing model for your DB2 products. Installing DB2 Federation on the same server as your Oracle database will give you optimal throughput. At the same time, it might cost you a fortune to have DB2 Federation installed on many target servers.

Example:

In our shop, we run Oracle on a set of centralized, dedicated database servers (actually virtualized Solaris zones). In such an environment, it is quite simple and cost effective to just add one virtual zone (per stage) which provides the necessary DB2 Federation services. Placed in the same subnet as the Oracle zones, this approach is feasible as long as your performance requirements are within the ordinary. In general, such database load is more network and I/O based than CPU intensive. Create your own concept on how to allocate instances and databases to your Oracle sources and targets. You might want to consider supporting various Oracle versions (-> different Wrapper definitions, possibly also different LIBPATH extensions and .profile, which would require difrerent DB2 instances) as well as security aspects here as well.

Oracle Considerations 1

- Trigger based capture mechanism
  - "Snail Factor": Updates slowed by factor 3 to 4
  - Increased log space requirements
  - Triggers are **NOT fired** with
    - TRUNCATE TABLE, LOAD, etc.
- Apply activity
  - Updates on source CCD tables (SYNCHPOINT)
    - Updated by Capture triggers also
    - Apply locks influence source table updates

Implications on Oracle DB

A capture mechanism that is built on triggers obviously increases the response time for updating transactions on the captured tables. The overall transaction workload for such systems is increased.

At the same time, the required log space is increased by a factor of about 3 for the captured tables. Remember: data is captured to a CCD table in the same transaction as the source table is updated. Every change to the source table (insert/update/delete) results in an insert of the changed row into the CCD table. Later on, a separate trigger-based pruning mechanism will delete superfluous data from the CCD table again.

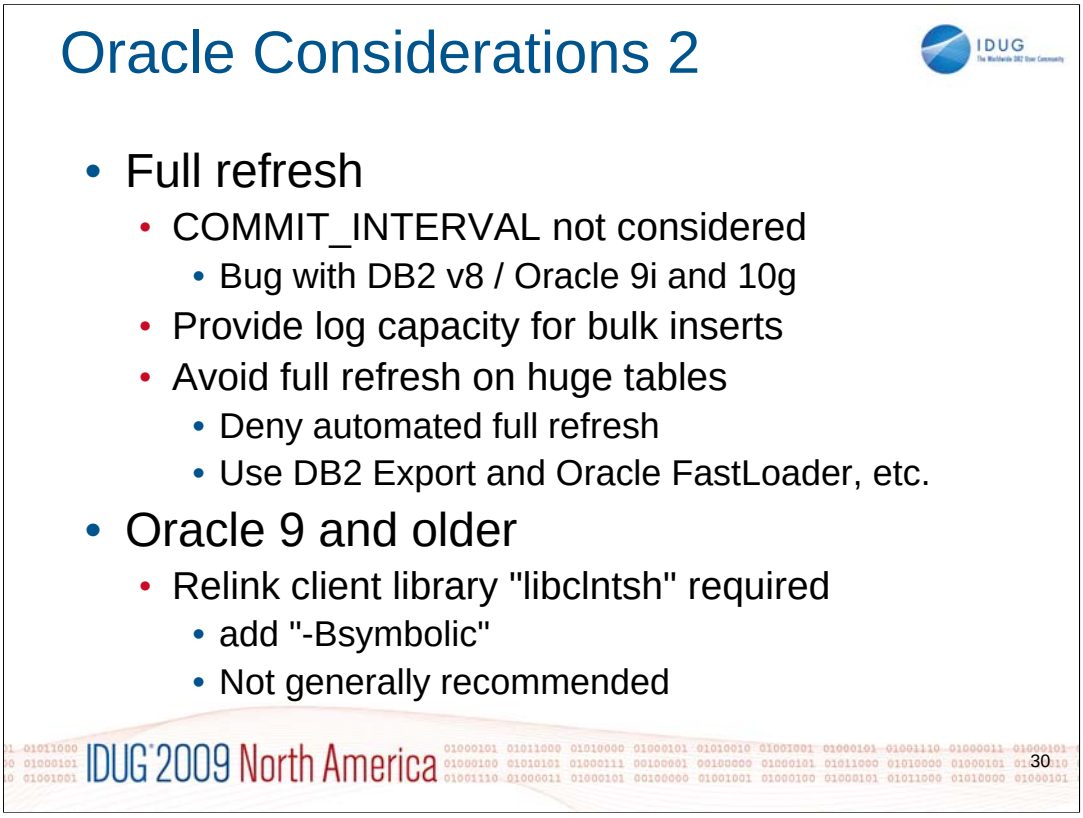
Results of a quick test: Insert of 1 Mio rows into a simple table with capture activated takes nearly 4 times longer than on the identical table without capture. The additional inserts on the CCD table add to the logging which increases (slow) pysical I/O.

Important: There are SQL statements which do not fire triggers, like e.g. TRUNCATE TABLE. In such a case, the trigger based mechanism fails to capture changes to the table. Consistency between source and target table is gone...

Locks on Oracle Source DB

Any application currently updating the Oracle source DB must finish before the Apply process can start applying data because Apply must lock the CCD table to set a synchpoint. These locks are not held through the whole Apply cycle, but until Apply is ready to set the synchpoint. During this time, any application updating the source tables must wait because the Capture triggers will need to update the the CCD table(s) in the same unit of work.

More details on data types can be found in “Websphere Information Integration 9, SQL Replication Guide and Reference” (SC19-1030-00).



Full refresh

We have encountered problems with full refresh for Oracle targets, running out of log space. It was easy to find out that the COMMIT_INTERVAL that we had set was not considered by Oracle, which resulted in requests for more log space than we had anticipated. But there was no way to overcome this limitation as it seemingly is a bug somewhere in the communication between DB2 and Oracle.

Therefore ensure you have enough log capacity for bulk inserts on the target. As an alternative, you might want to disallow full refresh on large tables anyway.

As an alternative, you can populate the target table by different means (like DB2 Export and Oracle Import or Load) and set the respective Flags in the Subscription Set Control Table to reflect a consistent state manually. DB2 Replication Monitoring Center (mentioned in the first part of this presentation) allows you to set this option with a simple mouse click (right click on Subscription Set or Member and select "Reset [Member] to Full Refreshed".

Oracle 9 and older:

Relink of Oracle client library "libclntsh" is required for Oracle versions before 10g

- Edit Oracle command file "genclntsh" and add '-Bsymbolic' to the link line (see below)
  - LD="ld -v -G -b +s -L$(OLIB) **-Bsymbolic**"
- Run genclntsh as user "root" to create a new main Oracle client library 'libclntsh'

Please note that with Oracle 10g and above, this step is not necessary anymore. Please only relink this Oracle client library if required; the "-Bsymbolic" option is not recommended to be used generally for executable modules.

Wow – we've done quite some stuff, right?

And we're slowly coming to an end. But - there are a few more topics worth mentioning because its just "good to know"!

Nickname Statistics

- DB2 uses statistics on Nicknames
  - Table and Column Cardinality
  - Existing indexes on remote table
  - Taken from target during Nickname creation
    - if available at target
- Updating Nickname statistics
  - Update Target Stats (use PASSTHRU mode)
  - Recreate Nicknames to reflect new Stats
    - Somewhat cumbersome...

DB2 keeps statistics on Nickname objects and their indexes. These values are considered for access path decision.

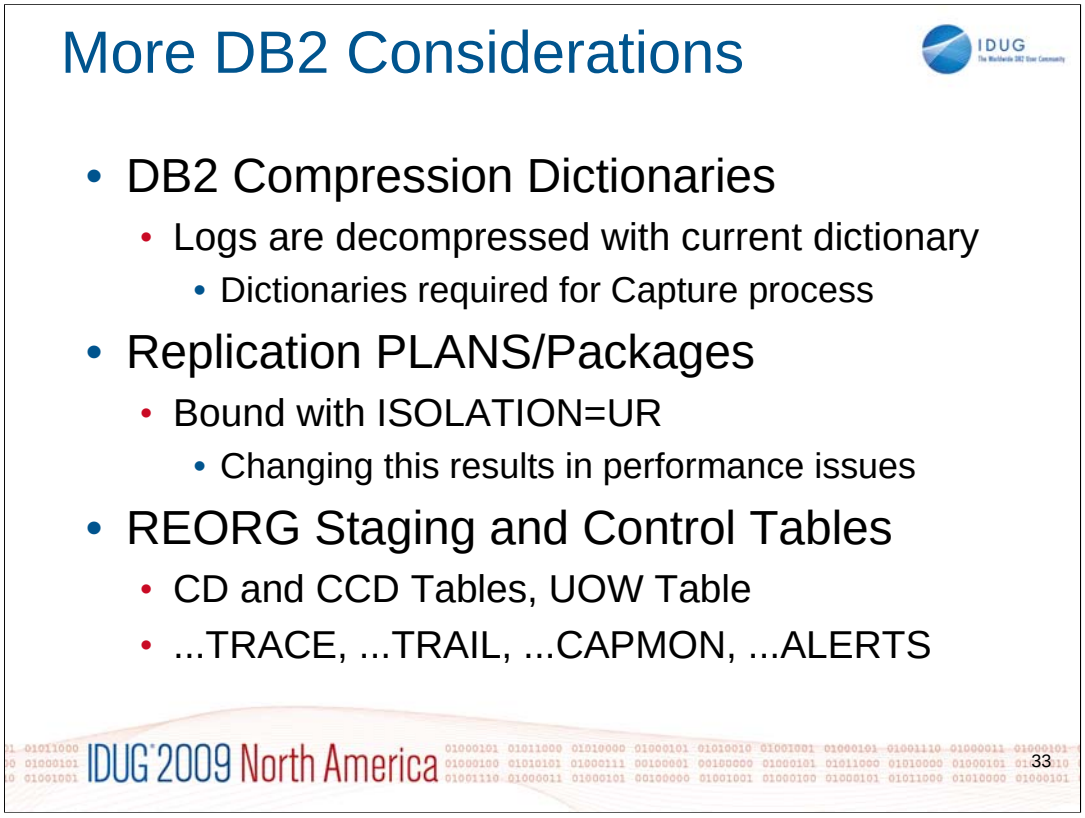
These statistics are taken from the target system at the time the Nickname is created. You can query the catalog for current values and to see what types of statistics are available for a specific nickname:

- select substr(tabschema,1,10), substr(tabname,1,12), card, npages, fpages, overflow, active_blocks from sysstat.tables where `tabschema = '[yourSchema]' and tabname = '[yourTable]'`
- select substr(tabschema,1,10), substr(tabname,1,12), substr(colname,1,12), colcard, substr(high2key,1,12), substr(low2key,1,12) from sysstat.columns `tabschema = '[yourSchema]' and tabname = '[yourTable]' select * from sysstat.columns where tabschema = '[yourSchema]' and tabname = '[yourTable]'`
- select substr(indschema,1,12), substr(indname,1,12), substr(colnames,1,40), nleaf, nlevels, firstkeycard, clusterratio from sysstats.indexes `where tabschema = '[yourSchema]' and tabname = '[yourTable]'`

There are a lot more columns in these stats tables and depending on what target system you use, they are (or are not) populated.

While this is basically a great feature, updating Nickname statistics is not really state of the art. Actually, there is no way to update them once a Nickname has been created, apart from:

- Ensure you have current and relevant statistics on the remote table
  - you can easily update them in PASSTHRU mode
    - typical command for ORACLE is "analyze table [table_name] compute statistics"
- Delete the Nickname
- Recreate the Nickname
  - This automatically updates the statistics
    - statistics are simply taken based on the current statistics that exist on the remote object
- If required: ALTER the Nickname to make it suit your needs
  - It's good to have the DDL statements for your Nicknames available



DB2 Compression Dictionaries

It is good to know that if compression is used in DB2, the log file contents are compressed as well. DB2 uses the current compression dictionary to compress the data contents within a log file.

When Capture is active, DB2 has to provide log files and uncompress their contents. Compression dictionaries of the associated tables are therefore required.

Replication PLANs and PACKAGEs

For performance reasons, all (or almost all) of the PLANs and PACKAGEs for Capture, Apply and Monitor are bound with uncommitted read isolation.

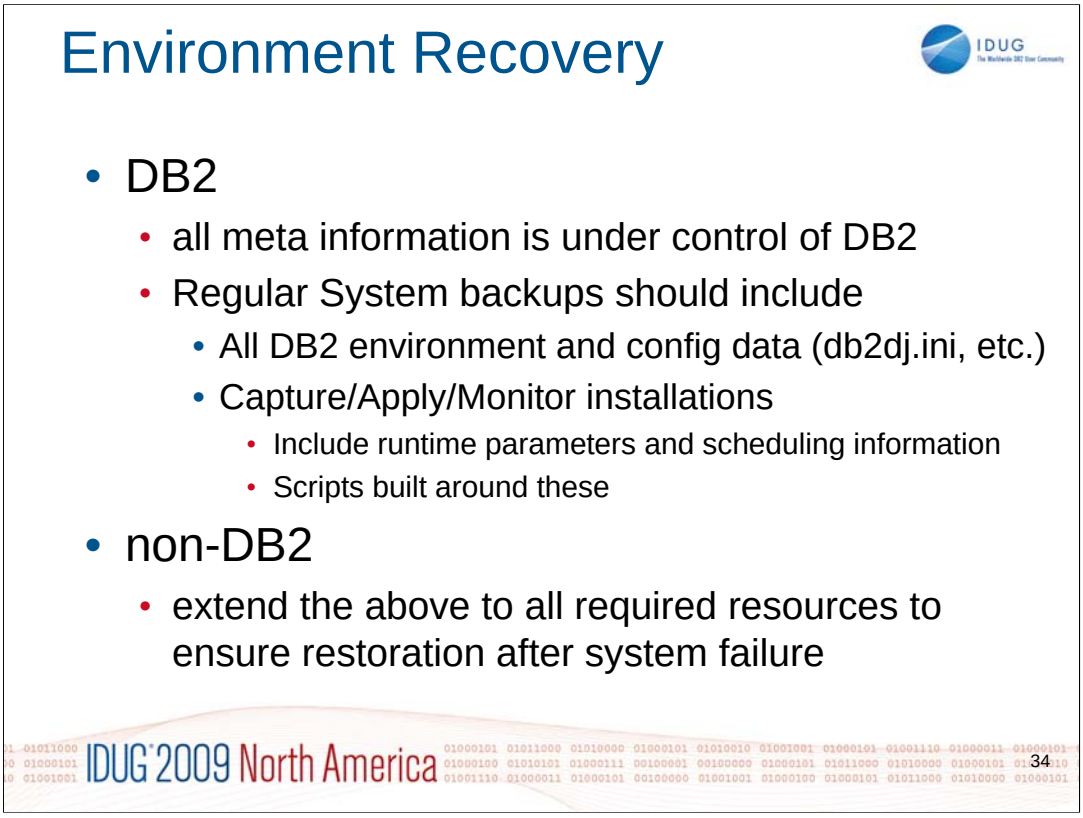
Rebinding these objects with other than ISOLATION=UR will most certainly result in performance degradation of the whole system.

REORG for Control Tables

Many of the Control Tables for a Replication environment encounter heavy insert activity and therefore are volatile in size. If created with recent versions of DB2, the respective "VOLATILE" option is generated in the DDL.

For a healthy environment, it is relevant to have certain Control Tables reorganized regularly (let's say weekly - as a starting point). This should include:

- CD and CCD tables
- IBMSNAP_UOW
- IBMSNAP_CAPMON
- IBMSNAP_CAPTRACE
- IBMSNAP_APPLYTRAIL
- IBMSNAP_APPLYTRACE
- IBMSNAP_MONTRAIL
- IBMSNAP_MONTRACE
- IBMSNAP_ALERTS



Save your environment against failures

The good thing first: Anything we have within our databases is as safe as our database data is. As all meta information is stored in DB2 tables:

- configurations are safe
- status is safe
    - crash recovery will perform rollforward and rollback of uncommitted transactions
        - any failure of a participating Replication system will restart at the position of the last known consistency point

When recovering to an earlier version of a database that uses Replication (version or PIT recovery), these automatisms do not apply. In this case, we need to set a new baseline on dependent systems. The easy solution for this has a name: Full Refresh.

If there are issues with the amount of data (or whatever), in-depth knowledge of the mechanisms is inevitable to regain consistency across heterogeneous systems.

DB2 environment

Shops who are used to running DB2 databases will have procedures or scripts in place ensuring that the environment specifics are backed up and stored in a safe place to allow complete recovery. This should include DB and DBM config files and registry variables, but also the instance owner's environment (.profile, userprofile, db2dj.ini), which is extended if you use heterogeneous federation.

Capture/Apply/Monitor processes

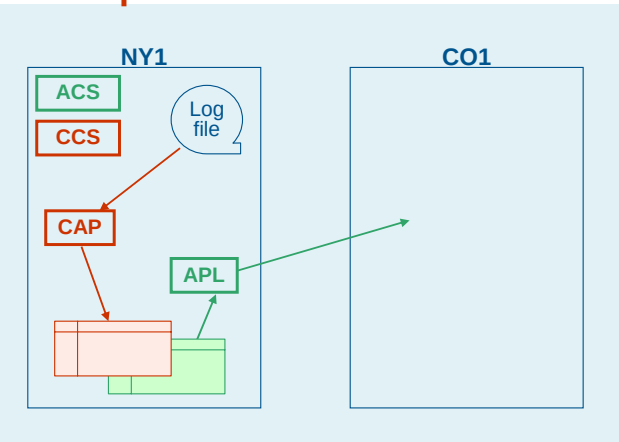
The processes which perform replication activities are independent of DB2. They are started, scheduled and maintained independently and as a result, you need to make sure you include the respective information in your environment backups.

non-DB2 Elements

In heterogeneous environments, we also have to deal with all the additional products which interact with our DB2 Replication system. Recovery of these products and their environments is purely product-specific; no additional implications arise from DB2 Replication.

Replication Scenarios – 1

1:1 Replication



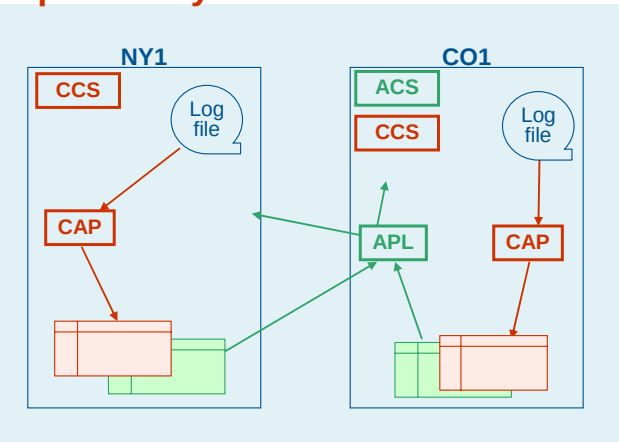
- Typical
- Simple
- Apply could also be located at CO1
  - would even perform better

The Replication Scenarios slides give you an idea of what can be achieved with DB2 Replication. "CCS" (Capture Control Server), "CAP" (Capture Process), "ACS" (Apply Control Server) and "APL" (Apply Process) show you where to best place your Control Servers and where to run your processes.

The graphics concentrate on Capture and Apply processes, their Control Servers and the Staging tables. Source and Target tables are not shown in these scenarios.

Replication Scenarios – 2

Update Anywhere



- Changes from both systems captured
- Applied to both systems
- ACS located on either system

The Replication Scenarios slides give you an idea of what can be achieved with DB2 Replication. "CCS" (Capture Control Server), "CAP" (Capture Process), "ACS" (Apply Control Server) and "APL" (Apply Process) show you where to best place your Control Servers and where to run your processes.

The graphics concentrate on Capture and Apply processes, their Control Servers and the Staging tables. Source and Target tables are not shown in these scenarios.

For this screnario:

- all of the source and target tables must be DB2 resources
- target tables must be of type "Replica"
- Apply has to ensure that no ping-pong effect occurs (via SIGNAL table)

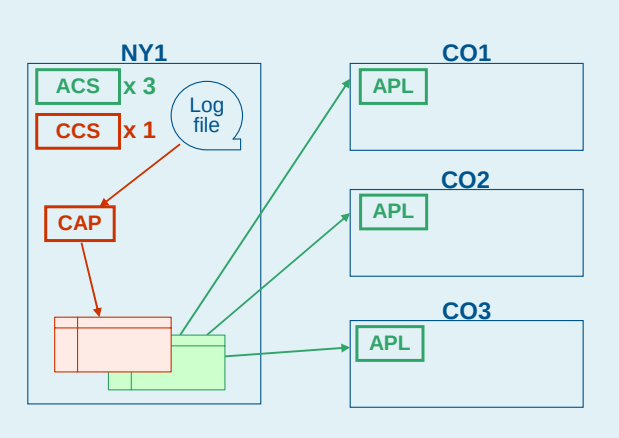
Performance considerations:

- Apply can run on either of the systems, but usually runs on the system that receives more data
- Initial full refresh might perform better if using LOAD
    - this can only be done on the system where Apply runs
    - Apply could also be run on "NY1" for initialization, then stopped and restarted on "CO1" for differential replication

See "A Practical Guide to DB2 UDB Data Replication V8" (SG24-6828-00), pg 141 for more details.

Replication Scenarios – 3

Distribution



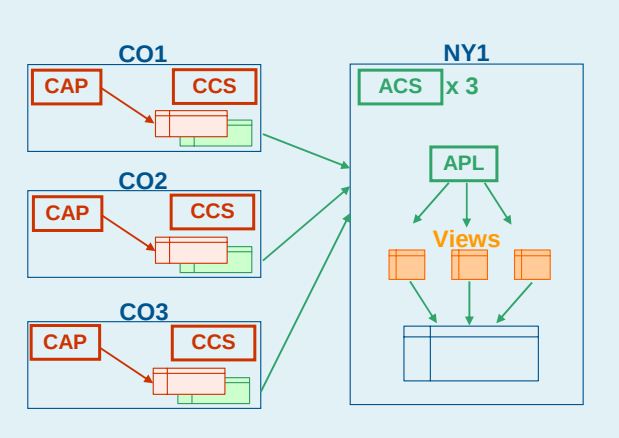
- Identical to 1:1 but with more than one recipient

The Replication Scenarios slides give you an idea of what can be achieved with DB2 Replication. "CCS" (Capture Control Server), "CAP" (Capture Process), "ACS" (Apply Control Server) and "APL" (Apply Process) show you where to best place your Control Servers and where to run your processes.

The graphics concentrate on Capture and Apply processes, their Control Servers and the Staging tables. Source and Target tables are not shown in these scenarios.

Replication Scenarios – 4

Consolidation



- Reverse of Distribution
- Initial refresh -> deletes all data!
  - Replicate **to views** with keys from client DB

The Replication Scenarios slides give you an idea of what can be achieved with DB2 Replication. "CCS" (Capture Control Server), "CAP" (Capture Process), "ACS" (Apply Control Server) and "APL" (Apply Process) show you where to best place your Control Servers and where to run your processes.

The graphics concentrate on Capture and Apply processes, their Control Servers and the Staging tables. Source and Target tables are not shown in these scenarios.

For consolidation of several source to one target table, we do have the problem of the initial refresh which deletes all data. To overcome this, we need to create views on DB "NY1" which are restricted to the keys of each of the DBs delivering the data ("CO1", "CO2", "CO3"). We would then replicate to these views instead of to the target table.

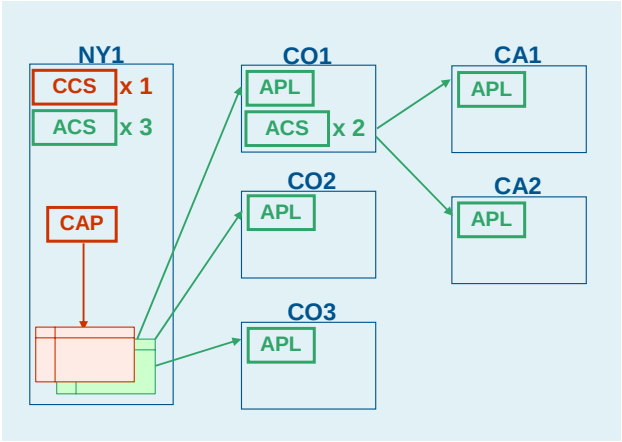
As an alternative to replicating to views, we could

Performance considerations:

- To enhance throughput, more than one Apply process may be started on DB "NY1"
  - Apply Qualifier in Subscription Set is relevant

Replication Scenarios – 5

Fan Out



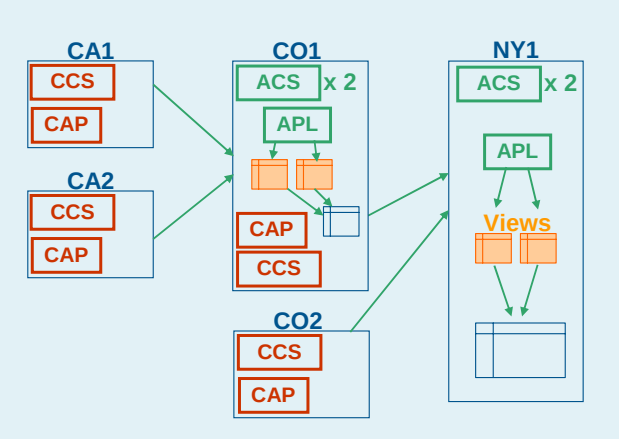
- CO1 requires no capture
  - if replicated as CCD
  - if not changed or enriched on CO 1
  - otherwise recapture!

The Replication Scenarios slides give you an idea of what can be achieved with DB2 Replication. "CCS" (Capture Control Server), "CAP" (Capture Process), "ACS" (Apply Control Server) and "APL" (Apply Process) show you where to best place your Control Servers and where to run your processes.

The graphics concentrate on Capture and Apply processes, their Control Servers and the Staging tables. Source and Target tables are not shown in these scenarios.

Replication Scenarios – 6

Multitier Consolidation



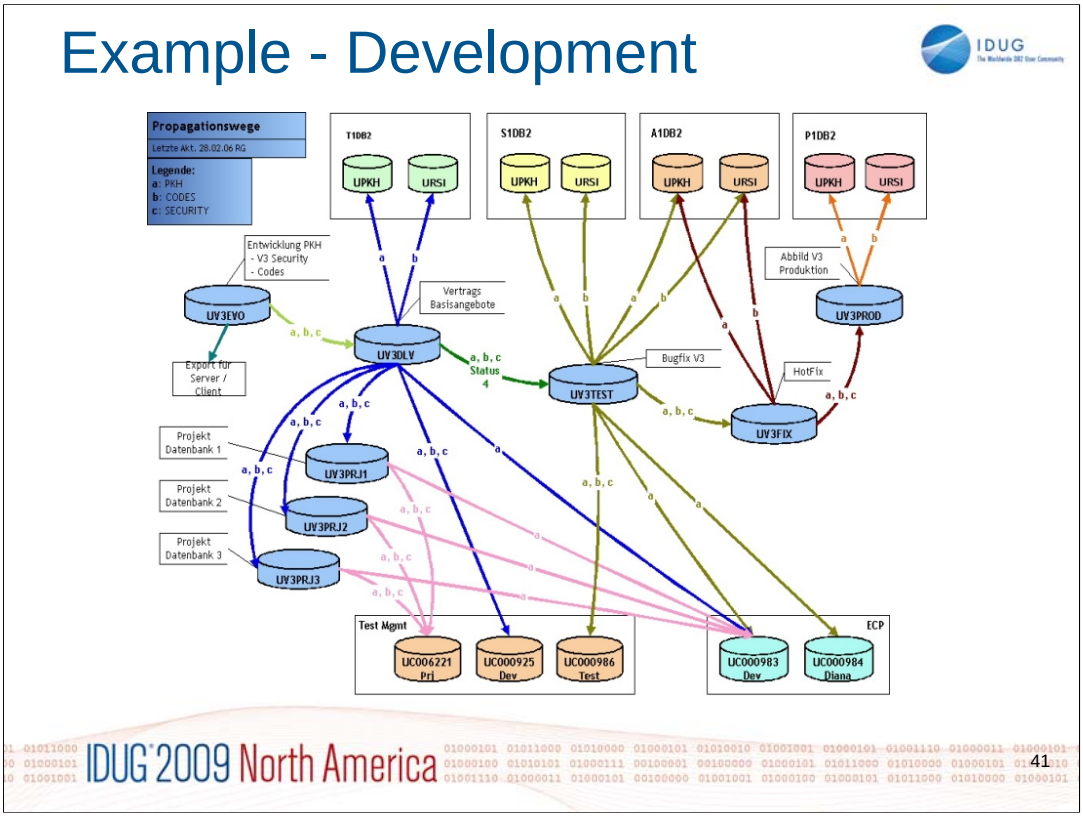
- CO1 requires recapture:
  - SYNCHPOINT is used as key
  - From disparate systems
  - incomparable

The Replication Scenarios slides give you an idea of what can be achieved with DB2 Replication. "CCS" (Capture Control Server), "CAP" (Capture Process), "ACS" (Apply Control Server) and "APL" (Apply Process) show you where to best place your Control Servers and where to run your processes.

The graphics concentrate on Capture and Apply processes, their Control Servers and the Staging tables. Source and Target tables are not shown in these scenarios.

For information on using views for replication, refer to scenario 4 (Consolidation).

Example - Development



This is a real-life view of the propagation environment used for the development of a financial application which is able to create, support and maintain a variety of different products in the area of finance and insurances.

Development is done on Windows platform (with a set of DB2 LUW servers) and involves a large amount of meta data which actually makes up the definition of these products. This meta data is therefore one of the major results of the development process and it is propagated along a clearly specified way through various stages from development to production, using DB2 Replication.

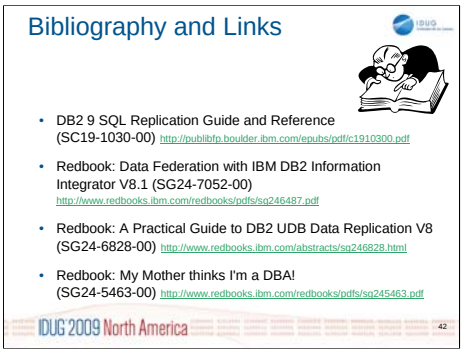
Stable versions of the meta data are propagated to the central mainframe (T1DB2 = Development, S1DB2 = Education, A1DB2 = Acceptance, P1DB2 = Production).

Not shown on this slide:

From the mainframe, this meta data is propagated to around 400 DB2 LUW servers and further on to the sales staff's laptops.

At the same time, all of the production data (mainly customer's contracts for all of the products) is replicated in both directions.

All of this is under control of the DB2 Replication feature.



There's a whole bunch more to be known about Replication/DataPropagation and this slide's contents lead you to useful resources for more details on this topic.

To what concerns the Redbook "My Mother thinks I'm a DBA!": Even though this book is 10 years old (and refers to terms like DataJoiner, etc.), much is still the same and the information therefore can be very useful. On the other hand, some of the information is outdated and therefore canbe misleading. But the book provides several case studies and gives you many interesting hints. Appendix A is an overview and reference to specific Data Replication tips, tricks and techniques

A base link to the official IBM books on Websphere Information Integration is here: http://www-01.ibm.com/support/docview.wss?rs=3572&context=SSDPBH&context=SS2K6Z&context=SS2KAH&context=SS2K84&context=SS2KBU&context=SSVH42&context=SS9UMX&uid=swg27008559&loc=en_US&cs=UTF-8&lang=en

Apart from the books mentioned on the slide, the web contains a whole bunch of further information on specific replication topics. Some interesting links are:

```
SQL Replication Roadmap: http://www.ibm.com/developerworks/db2/roadmaps/sqlrepl-roadmap.html
Q Replication Roadmap:   http://www.ibm.com/developerworks/db2/roadmaps/qrepl-roadmap.html
WebSphere Replication Server Home:   http://www.ibm.com/software/data/integration/replication_server
Replication Monitoring Center for DB2:   http://www.alphaworks.ibm.com/tech/db2rmc
```

Session: F07

DB2 Heterogeneous Replication Quickie

Peter Suhner

peter_suhner@hotmail.com

or via www.linkedin.com

What's left to say?

Maybe this: Whenever questions arise, don't hesitate to contact me.