LUW Database Replication

Brian Mealman
612-397-5468
bamealm@us.ibm.com
Agenda

• Replication Product Family Overview & History
• Packaging
• SQL Replication
• Queue Replication (QREP)
• Change Data Capture (CDC)
• Replication Tools
• Replication Uses Cases
• Replication Tips and Best Practices
• Q&A
Replication Product Family Overview & History
Data Delivery Methods

Virtual Data Delivery

- Federation
  - Product Performance
  - Real-time Inventory Level

Analytical & Reporting Tools
Web Applications

Incremental Data Delivery

- Change Data Capture
  - Primary Data Center
  - Backup Data Center

Bulk Data Delivery

- Extract, Transform, Load
  - Region 1
    - Product Performance
  - Region 2
    - Product Performance

- Change Data Capture
  - ETL
  - Message Queue
  - Business Application
Key elements of incremental data delivery

IMPACT (Lowers Risk and Cost)
- Lower cost of changed data access.
- Use of native DB logs, lower overhead.
- Non intrusive to applications and databases.
- No use of database triggers.
- Management easily integrated into existing IT operations.
- Reduces risk to operational systems.

LATENCY (Increases Business Visibility)
- Near-zero latency for pervasive integration projects.
- ETL can also deliver low latency but at significantly higher impact to production systems and mission-critical applications.

CONSISTENT DATA DELIVERY (Trusted Delivery)
- Data pushed, delivered in continuous stream, continuous with business operations.
- Transaction consistency maintained to preserve units of work, referential integrity.
- Full transaction granularity, before and after image of all transactional changes.
- Data event aware, can be used to trigger specific business processes.
- Fault tolerance, recover to last committed transaction.
Incremental Data Delivery

- Is provided by change data capture technologies for
  - Publishing to consuming applications,
  - Delivery to one or more consumers
  - Real-time integration

- Enabled by log-based capture of database changes

- With minimal impact to source systems

- Supporting a wide variety of sources and targets.
Replication Server Product Offerings

• **DB2 Linux UNIX & Windows:**
  1. **SQL Replication** (aka DB2 Data Propagator - since 1994)
     • Data changes captured by reading the logs on DB2 z/OS, Linux, Unix, Windows (LUW), and iSeries
     • Data captured by triggers for non-DB2 sources
     • DB2 to DB2/Informix replication comes with the DB2 database server
     • Captured changes are stored in relational tables, the Apply process fetches them over a database connection
     • Oracle, Sybase, SQL Server, Informix Teradata targets can be updated

• **InfoSphere Replication Server LUW:**
  1. **Q Replication and Q Data Event Publishing** (since 2004)
     • Data captured by reading the logs for DB2 z/OS, DB2 LUW, and Oracle
     • Captured changes are delivered via WebSphere MQ
     • Parallel Q Apply for performance - best in the market
     • Oracle, Sybase, SQL Server, Informix Teradata targets can be updated

The Capture and Utility programs, as well as the administration interfaces are *common* to both SQL and Q Replication technologies

• Some functions only available in Q Replication today (e.g., XML data, Replication Dashboard)

Also packaged with Replication Server (restricted licenses for replication use only)

• WebSphere MQ; InfoSphere Federation Server; DB2 (on distributed only, not on z/OS)

• **InfoSphere Change Data Capture (CDC)**
  1. IBM Acquires Data Mirror in 2007, the products are renamed InfoSphere CDC
     • Heterogeneous Replication solutions across many platforms and databases
     • Log based capture
     • TCP/IP data transport
     • Windows Client for Administration and Management
Replication Server Product Offerings

- The **DB2 (LUW) Homogeneous Replication Feature** is for DB2 <--> DB2 replication only
  - Contains both Q and SQL replication
  - No Federation Server and no Event Publisher capability

- **InfoSphere Data Replication** (2010)
  1. A bundle of all the IBM Replication Technologies
  2. The following are Supporting Programs licensed with the Program:
     - IBM DB2 Enterprise Server Edition v10.1
     - IBM InfoSphere Change Data Capture v6.2
     - IBM InfoSphere Change Data Capture v6.5.2
     - IBM InfoSphere Federation Server v10.1
     - IBM InfoSphere Replication Server v9.7
     - IBM WebSphere MQ v7.1
Replication Server Product History

- A long history of IBM as the Leader of Replication Technologies:
  1. **SQL Replication** (aka DB2 Data Propagator) - since 1994
  2. **Q Replication and Q Data Event Publishing** - since 2004
- Thousands of customers world-wide
- Replication Server can run with any down-level versions of DB2 (e.g., Q Rep V9.7 with DB2 z V7)

<table>
<thead>
<tr>
<th>Release/Version</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>DpropR V1</td>
<td>1994</td>
</tr>
<tr>
<td>DpropR V5</td>
<td>1997</td>
</tr>
<tr>
<td>DpropR V6</td>
<td>1999</td>
</tr>
<tr>
<td>DpropR V7</td>
<td>2000</td>
</tr>
<tr>
<td>DpropR V8</td>
<td>2002</td>
</tr>
<tr>
<td>DpropR V8 ( SQL Replication )</td>
<td>2004</td>
</tr>
<tr>
<td>WebSphere Replication Server V9.1</td>
<td>2006</td>
</tr>
<tr>
<td>InfoSphere Replication Server V9.5</td>
<td>10/2007</td>
</tr>
<tr>
<td>InfoSphere Replication Server V9.7</td>
<td>8/2009</td>
</tr>
<tr>
<td>InfoSphere Data Replication V10.1</td>
<td>8/2011</td>
</tr>
<tr>
<td>InfoSphere Data Replication V10.3</td>
<td>4/2012</td>
</tr>
</tbody>
</table>

---

IBM DB2 DataJoiner, Version 2.1
- Adds SQL bases
- Heterogeneous Replication

WebSphere II V8.2 (Q Replication)

--> Q Replication

--> SQL Replication

IBM DB2 DataJoiner, Version 2.1
- Adds SQL bases
- Heterogeneous Replication

Replication Shipped with DB2 LUW

Packaging
<table>
<thead>
<tr>
<th>Part number</th>
<th>Part description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0408LL</td>
<td>IBM InfoSphere Change Data Capture for Non Production Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0406LL</td>
<td>IBM InfoSphere Change Data Capture Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0402LL</td>
<td>IBM InfoSphere Change Data Capture for Oracle Replication Non Production Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0403LL</td>
<td>IBM InfoSphere Change Data Capture for Oracle Replication Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0NM3LL</td>
<td>IBM InfoSphere Change Data Delivery for Netezza Managed Server License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0B73LL</td>
<td>IBM InfoSphere Change Data Delivery for Non-Production Environments Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0B6ZLL</td>
<td>IBM InfoSphere Change Data Delivery Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0NNILL</td>
<td>IBM InfoSphere Data Replication for Database Migration Install Initial Fixed Term License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0L2NLL</td>
<td>IBM InfoSphere Data Replication for Non-Production Environments Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0L34LL</td>
<td>IBM InfoSphere Data Replication Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0NMBLL</td>
<td>IBM InfoSphere Data Replication for Netezza Managed Server License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D61B0LL</td>
<td>IBM InfoSphere Replication Server Developer Edition Authorized User License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D59ILLL</td>
<td>IBM InfoSphere Replication Server Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0TD0LL</td>
<td>IBM Netezza Replication Services Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
</tbody>
</table>
Replication Bundling

- IBM InfoSphere Data Replication as an included product:
  - IBM Database Enterprise Developer Edition
  - IBM DB2 Advanced Enterprise Server Edition
SQL Replication
Relational Replication - Components

- Independent operation
- Refresh, update, and enhancement
- Database connectivity and security
Relational Replication - Capture

- Captures base table changes from log
- Runs locally to the source
- Maintains transaction consistency
- Automatically maintains staging tables
Relational Replication - Apply

- Runs from source or target platform
- Runs at user-specified intervals or events
- Refreshes, updates, and enhances copies
- Distribution optimizations

![Diagram of Relational Replication - Apply](image)
Relational Replication - Data Enhancement

Customize source data for specific target use

- Enhances data usability
- Supports unique application needs
Relational Replication - Subset Distribution

Customer Table

<table>
<thead>
<tr>
<th>CUST#</th>
<th>CUSTNAME</th>
<th>CUSTADDR</th>
<th>BRANCH</th>
</tr>
</thead>
</table>

Policy Table

<table>
<thead>
<tr>
<th>POL#</th>
<th>CUST#</th>
<th>TYPE</th>
<th>EFFDATE</th>
<th>EXPDATE</th>
</tr>
</thead>
</table>

SELECT * FROM POLICY
WHERE EXISTS (SELECT * FROM CUSTOMER
WHERE POLICY.CUST# = CUSTOMER.CUST# AND CUSTOMER.BRANCH = 'DALLAS')

Customer & Policy Data for Dallas Branch
Relational Replication - Fan-Out Distribution

Where City = LA

Where Branch = 6

Where City = SF

Where State = CA

Where Branch = 7

Where State = NY

Data Source

San Francisco

Los Angeles
Relational Replication - Administration

- GUI - part of DB2 Control Center or DSAC
- Registration/Subscription
- Relational control tables
Relational Replication - Highlight Functions

- Integrated replication administration
- Update Anywhere
- Mobile computing support
- Join View Support
- Set subscription
- Event based scheduling support
- Batch execution of Capture and Apply
- Logical partitioning key support
- Defer/run SQL support
- Subscription Cloning
- Stored procedure call
- Large answer set support
Relational Replication - Update Anywhere Implementation

• Avoid update conflicts by design
  ▪ Use application views over the replicas to enforce "distributed primary fragment" mutually exclusive update restrictions
  ▪ Serialize the schedule for when each site can issue updates
  ▪ Write insert-only applications

• Handle rejected transactions
  ▪ ASNDONE exit
  ▪ Rejection codes
  ▪ Before/After row values
  ▪ RETENTION_LIMIT pruning
Mobile Computing Support

- On demand replication
- Initiated from mobile station

- Head office
- Branch office

Mobile Sales Force

- Replication on demand
- Selective replication
- Dial exit
- Windows & Linux
Transaction & Non-Transaction Replication

- Supports both transaction and non-transaction replication

**Transaction Replication**
- Every Update

**Non-Transaction Replication**
- Net Updates Only

**Diagram Details**
- **SOURCE**
  - BASE
  - CHANGE DATA
  - CCD
  - CAPTURE
  - Log: Full Row Logging

- **TARGET**
  - APPLY
  - TRANSACTION Replication
  - Every Update
  - Non-Transaction Replication
  - Net Updates Only
Relational Replication - Join View Support

- Replication through join views
  - Denormalize copies for data warehousing
  - Route data for distributed computing

Action ➔ DEFINE VIEW
- One or more user tables
Relational Replication - Set Subscription

- Every subscription must belong to a set. A set may have one or more subscription members.
- Maintains referential constraints that exist among a set of tables at both source and target servers.
- Limits the boundary for cascade rejections due to RI violation or update collision.
- Keeps the subscriptions for all the components of a view subscription together.
Relational Replication - Event Based Scheduling Support

- Subscriptions can be triggered by
  - Relative time
  - Event timer
  - Both

Diagram:
- SOURCE
  - Log
  - Full Row Logging
  - Capture
- TARGET
  - Control
  - Apply
  - Copy
  - Replica
- Control Table
  - Event Name
  - Event Time
  - End Of Period
  - * (Upper bound for change data)
Logical Partitioning Key Support

- Allows a row to migrate from one partition to another

![Diagram showing Logical Partitioning Key Support](image)

- Change AGENT # from 6 to 99
- Processes "Delete" with Agent# = 6
- Processes "Insert" with Agent# = 99

**SOURCE**

- CUST#
- CUSTADD
- CUSTNAME
- AGENT#

**BASE**

**CHANGE**

**CAPTURE**

**TARGET**

- COPY

**APPLY**

- BEFORE IMAGE: Agent# = 6
- AFTER IMAGE: Agent# = 99
Relational Replication - Defer/Run SQL Support

- Allows customization of DDL before execution
- Saves definitions to plain ASCII file
Relational Replication - Subscription Cloning Support
Use MAX_SYNCH_MINUTES to break up the backlog into mini-subscriptions, using multiple real subscription cycles to perform one scheduled cycle. This applies to change data and not to full refresh.
IBM SQL Replication Features

- Versatile Infrastructure
  - Operational application
  - Informational application
  - Mobile computing
  - Update anywhere
- Leverages Current Assets
  - Legacy Sources
  - Standard SQL
- Efficient Operations
  - Log-based capture
  - Network optimizations
- Flexible Distribution
  - Subsetting with join views
  - Cascading distribution
- Scalable Design
  - Servers
  - Data
  - Networks
- Easy to Administer
  - GUI administration
  - Automated initialization
- Robust Data Enhancements
  - Derivation, Summarization, Translation...
- Multi-vendor Interoperability
  - SQL based architecture
  - Architected data staging area
Queue Replication (QREP)
Q Replication Components

- Replication Center – defines replication source-to-target mappings, manages Q Replication processes, provides monitoring reports, defines and manages Alert Monitor processes
- Q Capture – captures changes from the DB2 log and places them on a WebSphere MQ queue
- WebSphere MQ – transport for captured changes
- Q Apply – retrieves captured changes from a WebSphere MQ queue and processes them
- Alert Monitor – monitors Q Capture and Q Apply based on user-defined thresholds and events, sends e-mail notification when thresholds are exceeded or events occur
- Utilities – asntdiff/asntrep for reconciliation, ansqmfmt/exception formatter and Q Analyzer for diagnosis
- Data Studio Administration Console (DSAC) – the new interface to monitor health of Q Replication
- DSAC includes the Replication Dashboard – real time monitoring of Q Capture and Q Apply
Unidirectional Q Replication – DB2 to DB2

**USES**
Maintaining operational data stores, decision support systems, reporting systems, data warehouses

**FEATURES**
Filtering by
- row and column
- change type – replication of deletes can be suppressed
- authorization id, plan name
SQL transformations

**PERFORMANCE**
12,500 – 15,000+ changed rows replicated per second in development lab environment
Unidirectional Q Replication – DB2 to non-DB2

USES
Maintaining operational data stores, decision support systems, reporting systems, data warehouses

FEATURES
Filtering by
- row and column
- change type – replication of deletes can be suppressed
- authorization id, plan name
SQL transformations

PERFORMANCE
Some customers have reported 6000 rows per second. Other customers report much less.
Unidirectional Q Replication – DB2 to DB2 CCD

**USES**
Consistent Change Data (CCD) tables are an audit trail of changes. They are used for auditing and feeding changes to other applications.

**FEATURES**
Filtering by
- row and column
- change type – replication of deletes can be suppressed
- authorization id, plan name
SQL transformations

**PERFORMANCE**
12,500 – 15,000 changed rows replicated per second in development lab environment
Unidirectional Q Replication – DB2 Data Distribution

USES
Beginning with v9.5 Fixpack 1, CCDs can act as a source for SQL Replication. This adds “fan-out” capability to Q Replication.

FEATURES
Filtering by
● row and column
● change type – replication of deletes can be suppressed
● authorization id, plan name

SQL transformations

PERFORMANCE
Numbers not available
Unidirectional Q Replication – DB2 to Stored Procedure

<table>
<thead>
<tr>
<th>USES</th>
<th>Post-processing of captured transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEATURES</td>
<td>Filtering by</td>
</tr>
<tr>
<td></td>
<td>● row and column</td>
</tr>
<tr>
<td></td>
<td>● change type – replication of deletes can be suppressed</td>
</tr>
<tr>
<td></td>
<td>● authorization id, plan name</td>
</tr>
<tr>
<td></td>
<td>SQL transformations</td>
</tr>
<tr>
<td>PERFORMANCE</td>
<td>Q Apply calls the stored procedure for each changed row received and waits for the stored procedure to return. We expect that this will not perform as well as unidirectional to a DB2 target.</td>
</tr>
</tbody>
</table>
Event Publisher for DB2 – DB2 to XML

**USES**
Feed applications (WebSphere Message Broker) or ETL programs (WebSphere DataStage)

**FEATURES**
Same filtering and transformations as DB2 -> DB2
Different message content – 1 message per row changed or 1 message per transaction

**PERFORMANCE**
5000-10000 changed rows replicated per second in development lab environment, but this performance is only measured for the publishing of the XML message.
Bidirectional Q Replication – DB2 to DB2 (2 Sites)

USES
Maintaining high availability or disaster recovery databases

FEATURES
No filtering
Standby databases can be used for reporting, queries, and/or other applications (use caution)
Value based conflict detection
One site is designated as the winner for conflict resolution

PERFORMANCE
Slower than unidirectional (10,000 – 15,000 rows replicated per second), depending on the number of conflicts
Bidirectional Q Replication – DB2 to DB2 (3 Sites)

<table>
<thead>
<tr>
<th>USES</th>
<th>Maintaining high availability and disaster recovery databases. This is a master-replica configuration.</th>
</tr>
</thead>
</table>
| FEATURES | No filtering  
Standby and DR databases can be used for reporting, queries, and/or other applications (use caution)  
Value based conflict detection  
One site, usually the primary is designated as the winner for conflict resolution. |
| PERFORMANCE | Slower than unidirectional (10,000 – 15,000 rows replicated per second), depending on the number of conflicts |

This is a new configuration, supported by InfoSphere Replication Server 9.5 and above.
Bidirectional Q Replication – DB2 to DB2 Two-tier

**USES**
Maintaining high availability and disaster recovery databases. This is a two-tier configuration that minimizes the replication work done on the primary.

**FEATURES**
- No filtering
- Standby and DR databases can be used for reporting, queries, and/or other applications (use caution)
- Value based conflict detection
- One site is designated as the winner for conflict resolution in each tier.

**PERFORMANCE**
Slower than unidirectional (10,000 – 15,000 rows replicated per second at each tier), depending on the number of conflicts.

This is a new configuration, supported by InfoSphere Replication Server 9.5 and above.
Peer-to-Peer Q Replication – DB2 to DB2 (2-n Sites)

**USES**
Split query workload and provide high availability

**FEATURES**
No filtering
All peer databases are available for application updates
Key column conflict detection
Version based (timestamp) conflict resolution – timestamps maintained by triggers on source/target tables

**PERFORMANCE**
20% slower than unidirectional with higher CPU due to trigger execution
Q Replication Architecture
Q Replication High Level Architecture

- **DB2 Source** (Log based)
- **Q Capture**
- **WebSphere MQ**
- **Q Apply**
- **DB2 Target**

- Admin
- Utilities
- Monitor

OR

- Non-DB2 target via InfoSphere Federation Server (Included)
Q Replication Setup

• Install and configure WebSphere MQ 6.x or 7.x server on all source and target servers

• Install and configure InfoSphere Replication Server on all source and target servers

• Install DB2 Client 9.7 on replication administration workstation (also need DB2 Connect if sources and/or targets are DB2 Universal Database for OS/390® and z/OS®)

• Use the Replication Center to create Q Capture and Q Apply control tables

• Use the Replication Center to define Q Subscriptions and Queue Maps (send/receive queue pairs)

• Start Q Capture

• Start Q Apply

• Start Alert Monitor (optional)

• Start Q Replication Dashboard (optional)
MQ Setup for Queue Replication

1. Adming for Q Capture to receive control messages from QApply or subscribing app
2. Restartq holds the Q Capture position in the DB2 log
3. Sendq that points to the target receive queue – Capture transmits messages on this queue

1. Recvq for QApply to receive the transaction and informational messages from Q Capture
2. Spillq, dynamic queue for QApply to hold the transaction messages as the target table is being loaded
3. Adming that points to the source adminq – Apply sends messages to Capture on this queue.

Graphical checklists are available to help configure MQ for Q Replication.
You fill in the blanks
Setting up WebSphere MQ objects on the source system

1. On the source system, open a new command window and use the following command to create the queue manager:
   
   ```bash
   otware GMSYSA
   ```

2. Open a text editor. Copy and paste the following code into a new file and save it as `uni_source.txt`:
   
   ```bash
   *S2. Restart queue
   DEFINE QLOCAL('SYSA RESTARTQ') DEFPST(YES)
   *S3. Administration queue
   DEFINE QLOCAL('SYSA ADMMQ') DEFPST(YES)
   *S4. Send queue
   DEFINE GREMOTE('SYSA.SENDQ') RNAME('SYSA.RECVQ') RQMNAM('QMSYSB') XMITQ('SYSA.XMITQ') DEFPST(YES)
   *S5. Transmission queue
   DEFINE GLOCAL('SYSA.XMITQ') USAGE('XMITQ') DEFPST(YES)
   *S6. Sender channel
   DEFINE CHL('SYSA2SYSB') CHLTYPE('SDR') TRPTYPE('TCP') CONNAME('9.99.999/1414') XMITQ('SYSA.XMITQ') DISCNT (0)
   *S7. Receiver channel
   DEFINE CHL('SYSA2SYSA') CHLTYPE('RCVR') TRPTYPE('TCP')
   *Start the channel:
   START CHL('SYSA2SYSB')
   ```

3. Start the queue manager:
   
   ```bash
   startmq GMSYSA
   ```

4. Run the `uni_source.txt` script by issuing the following command:
   
   ```bash
   Linux or UNIX: runmqsc GMSYSA < tripath/uni_source.txt
   Windows: runmqsc GMSYSA < tripath/uni_source.txt
   ```

5. Start the listener:
   
   ```bash
   Linux or UNIX: runmqsr -h tcp -m GMSYSA -p 1414 & (Runs in the background)
   Windows: start runmqsr -h tcp -m GMSYSA -p 1414 & (Runs in a new window that will not show any output)
   ```

Proceed to page 3 to begin setting up WebSphere MQ objects on the target system.

The checklist generates the MQ commands
MQ Client Support

Distributed platforms only
Allows separation of Database servers and MQ servers
Allows replication support on platforms which currently lack MQ Server support
Supports HADR
MQ Client Support and HADR

New – MQ Server not required on source or target
The BIG Picture
Administration - The Replication Center
Replication Center Choices

Getting Started with the Replication Center
Welcome to the launchpad for the Replication Center! Click on a button to set up a basic scenario in Q replication, event publishing, or SQL replication. After you click a button, the Select launchpad view field appears at the top of the launchpad. Use this field to switch between the launchpad views for the three different tools.

Q replication
This high-volume, low-latency replication solution captures changes that are made to source tables, transports committed changes via WebSphere MQ message queues to a target location, and applies the changes to target tables. You can replicate to update tables in decision support systems, tables in a backup database for failover situations, tables that are used by geographically-distributed applications, and more.

Event publishing
This publishing solution, which is a feature of Q replication, captures changes that are made to source tables and publishes committed changes as XML messages on WebSphere MQ message queues. You determine what happens to the published changes. You can use them to feed data to your web applications, trigger events, and more.

SQL replication
This replication solution captures changes that are made to source tables and stores committed changes in staging tables. The changes are replicated from the staging tables to target tables or views. You can replicate to multiple targets from data that is captured only once. You can also use the staging tables to audit change history.
Replication Center Launchpad

1. Create Q Capture Control Tables
2. Create Q Apply Control Tables
3. Create a Q Subscription
4. Start a Q Capture Program
5. Start a Q Apply Program

A Q Apply program receives messages containing committed changes to source tables and applies the changes to corresponding target tables. You can either start it with the default settings or modify those settings for your environment.

When you create a Q subscription, you can specify whether you want the target to be loaded automatically when replication begins. In this case, the Q Apply program calls the appropriate utility, depending on the platform of the Q Capture and Q Apply servers, to load the target when you activate the Q subscription.
Replication Center – Definitions

• The Replication Center must have DB2 (DRDA) connectivity to all source and target servers.
  • If the target is non-DB2, then the Replication Center must have DRDA to the federated database that connects to the non-DB2 target.

• The Replication Center is used to DEFINE replication objects.
  • The Replication Administrator uses wizards to define replication.
    • Q Capture and Q Apply control tables
    • SENDQ/RECVQ pairs
    • Q Subscriptions and/or XML Publications

• The Replication Center generates SQL scripts based on the Administrators input. The scripts are run using the Replication Center and can also be saved for reference or reuse.

• Replication definitions can be changed using the Replication Center.
Replication Center – Operations

• The Replication Center can be used to OPERATE Q Capture and Q Apply.
• The Replication Administrator can perform the following operations
  • Start and Stop Q Capture and Q Apply
  • Check the status of Q Capture and Q Apply
  • Display reports (throughput, latency, messages)
• The Replication Center generates replication command scripts to stop and start Q Capture and Q Apply and to check status. The scripts are run using the Replication Center and can also be saved for reference or reuse.
• Reports are generated via SQL and displayed immediately.
• Q Apply and Q Capture runtime parameters can be changed using the Replication Center.
Replication Center -- Monitoring

• The Replication Center can be used to MONITOR Q Capture and Q Apply.
• The Replication Administrator can perform the following operations
  • Create Monitor Control Tables
  • Set thresholds and choose events for alert monitoring
  • Identify contacts and groups of contacts for notification via an SMTP mail server
  • Start and stop the Alert Monitor
• The Replication Center generates SQL and replication command scripts to stop and start the Alert Monitor and to check status. The scripts are run using the Replication Center and can also be saved for reference or reuse.
Administration – asnclp command line processor

asnclp

- Command line processor to define Replication Scenarios
- Calls same Java™ APIs as the Replication Center
- Interactive and Script Mode supported

Example

- Interactive Mode

  C:\asnclp
  asnclp session set to q replication;
  set output target script "create_apply_cntl.sql";
  set log "create_apply_cntl.err";
  set server target to db LOCATION id YourTSOid password "YourTSOpassword";
  set apply schema ASNV9;

- Script Mode

  C:\asnclp -f replscript.asn
The BIG Picture

**SOURCE**
- SOURCE1
- SOURCE2
- DB2 Log
- Capture

**ADMINISTRATION**
- Replication Center
- Replication Monitor

**TARGET**
- TGT1
- TGT2
- TGT3
- Q Apply Browser
- Apply Agent
- Apply Agent
- Apply Agent

**METADATA**
Source Table Requirements

• Unidirectional, Bidirectional, Peer-to-Peer, XML Publishing
  • If the source is DB2 for Linux, UNIX, or Windows, RECOVERY logging must be enabled.
  • If the source table does not have the DATA CAPTURE CHANGES attribute, that attribute will be set during replication definition

• Bidirectional, Peer-to-Peer
  • The source table must have a set of columns that uniquely identify a row so that the Apply program or XML generator can locate the row to be inserted, updated, or deleted

• Replication is not supported for
  • Data Links columns
  • Spatial columns
  • DB2 z/OS columns with FIELDPROCs, or VALIDPROCs (EDITPROCs are supported)
  • DB2 Linux, UNIX, Windows XML data type support is planned for a future fixpack.
  • Source table LOADs done by the DB2 LOAD utility or command are NOT replicated.
  • Source data stored in compressed tablespaces on z/OS must be REORGed with KEEPDICTIONARY YES (DB2 V8 APAR PK19539 removes this restriction)
Q Capture

In-Memory-Transactions

TX1: INSERT S1
TX1: UPDATE S1
TX1: COMMIT
TX3: DELETE S1
TX3: ROLLBACK
TX2: INSERT S2

DB2 Log

SOURCE2
SOURCE1

Q-SUBS
Q-PUBS

Restart Queue

Send Queue

MQ Put when Commit record is found

Transaction is still "in-flight"
Nothing inserted yet.

„Zapped“ at Abort
Never makes it to send queue
Filtering and subsetting

- Subset data
  - Subset of rows through Q Capture predicate on subscription/publication
  - Subset of columns through subscription/publication definition
  - Option included for ignoring deletes
  - Filter transactions by userid, plan name using entries in a control table
  - Signal defined to allow user selected transactions to be ignored

- Subsetting is done by
  - Q Apply during the initialization of the target table
  - Q Capture when capturing changes
Q Capture Miscellaneous

• One Q Capture process (schema) can process multiple SENDQs.

• For Peer to Peer replication, the Q Capture and Q Apply schemas must be the same on all Peer nodes.

• Q Capture and Q Apply control tables must be on the catalog node in a DB2 for Linux, UNIX and Windows partitioned database.

• Q Capture can handle LOBs that are larger than the maximum message size allowed by WebSphere MQ. Q Capture splits the LOBs into multiple messages based on the maximum message size defined for the SENDQ. Q Apply combines the LOB messages before applying the change.

• Q Capture can handle large transactions by sending multiple messages.

• Columns added to source tables can be added to replication (for DB2 targets) via the Replication Center
  • Alter add to the source table
  • Insert ADDCOL signal to the Q Capture IBMQREP_SIGNAL table
  • New column automatically added to replication control tables and the target table
The BIG Picture

SOURCE

SOURCE1

SOURCE2

METADATA

DB2 Log

Q Capture

ADMINISTRATION

Replication Center

Replication Monitor

TARGET

Q Apply Browser

Apply Agent

Apply Agent

Apply Agent

TGT1

TGT2

TGT3

METADATA
Target Table Requirements

- Bidirectional and Peer-to-Peer
  - Each target table MUST have a set of columns which uniquely identify each row in the target table.
  - Uniqueness can be defined as a primary key or unique index on the target table
- Unidirectional only to non-DB2 targets (Oracle, Sybase, etc.), CCD targets, and stored procedures
- All configurations
  - If target tables are related through referential constraints or through application logic, then they must be be processed by a single SENDQ/RECVQ pair so that all transactions are processed in the proper order.
  
- LOBs cannot be replicated to non-DB2 targets except Oracle. This is a Federation Server restriction – LOB updates are not supported for any data source but Oracle.
Q Apply: Initial Load Processing (Full Refresh)

- Handles RI automatically
  - i.e. drops and saves RI, loads, puts RI back on
- Q Capture begins capturing changes before Q Apply starts the unload of the source table, to ensure that no changes are lost.
  - Changes that are captured during the load are placed on a temporary spill queue and processed after the load is complete.
Target Table Initialization

• Q Capture detects a new Q subscription and inserts a CAPSTART signal in the Q Capture control table capschema.IBMQREP_SIGNAL. This log sequence number of this insert is the point in the log where capturing will start for the source table.

• Q Capture sends a SCHEMA msg to Q Apply on the SENDQ and begins capturing transactions and places them on the SENDQ for Q apply.

• Q Apply creates a SPILLQ from the model definition and saves the transactions in the RECVQ to the SPILLQ.

• Q Apply does the initial refresh based on the SCHEMA msg from Q Capture and the LOAD options in the IBMQREP_TARGETS table row for the Q Subscription.

• Q Apply applies the transactions from the SPILLQ and then deletes the SPILLQ.

• Q Apply applies the transactions from the SENDQ.
Q Apply Load Options

- A subscription can be defined with:
  - automatic load, manual load, no load required

- Automatic load:
  - Load is performed by Apply, with automatic coordination of the simultaneous capture of changes, loading of the new table, and apply of changes to other tables.

- Manual load:
  - Load is performed by user, coordination is required, and will be handled by user (with some help from our administration).

- No load:
  - No loading required, no coordination required, can immediately capture and apply changes
  - Example: target system is built through backup/restore, with replication started from an inactive source
Q Apply Processes

SOURCE

SOURCE1

SOURCE2

METADATA

Q Capture

DB2 Log

TARGET

Q Apply Browser

Apply Agent

Apply Agent

Apply Agent

TGT1

TGT2

TGT3

METADATA

SOURCE
How Does Apply work for unidirectional?

• A Q Apply program can process multiple RECVQs.
• A Q Apply browser thread is started for each RECVQ defined in the IBMQREP_TARGETS table. This thread coordinates the application of changes for the associated target tables.
• A Q Apply browser thread starts 1 or more Q Apply agents (configurable by the Replication Administrator)
• Transactions are passed from the Q Apply browser to the Q Apply agents and are processed in parallel if possible.
  • Transactions which affect the same rows in the same table are always processed in order by a single Apply agent
  • Transactions which affect tables that are related by RI constraints are always processed in order by a single Apply agent.
• Each message has a unique, sequential message id. Q Apply saves the last message id processed for restart.
Q Apply Transformations  SQL expressions

Fixpack 1: Transformations for non-key columns
Fixpack 2: Transformations for key columns and non-key columns

<table>
<thead>
<tr>
<th>Target Column</th>
<th>Target Column Expression</th>
<th>Mapping Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY1</td>
<td>KEY1</td>
<td>(1-1 mapping)</td>
</tr>
<tr>
<td>C12</td>
<td>[:C1</td>
<td></td>
</tr>
<tr>
<td>C2A</td>
<td>[substr(:C2,2,3)]</td>
<td>(1-N mapping)</td>
</tr>
<tr>
<td>C2B</td>
<td>[substr(:C2,5,5)]</td>
<td>(1-N mapping)</td>
</tr>
<tr>
<td>C2C</td>
<td>[int(substr(:2,1,1))]</td>
<td>(1-N mapping)</td>
</tr>
<tr>
<td>C34</td>
<td>[:C3 + :C4]</td>
<td>(N-1 mapping)</td>
</tr>
<tr>
<td>C5</td>
<td>[CURRENT_TIMESTAMP]</td>
<td>Generated column</td>
</tr>
<tr>
<td>C6</td>
<td>‘IBM’</td>
<td>Generated column</td>
</tr>
<tr>
<td>C7</td>
<td>substr(‘1’,1,1)</td>
<td>Generated column</td>
</tr>
</tbody>
</table>

Generated columns C12, C2A, C2B, C34 are based on source table Column values.

Generated columns C5, C6, C7 are literals.

Expressions are stored in Q Apply control table IBMQREP_TRG_COLS
How Does Apply work for bidirectional?

• Bidirectional replication is supported for 2 or 3 servers.
• Each server has a Q Capture and Q Apply program.
• There are 2 sets of MQ definitions on each server.
  • Queue definitions for Q Capture
  • Queue definitions for Q Apply
• The Q Subscription type is Bidirectional and the replication definitions are automatically setup through the Replication Center.
• Q Capture behaves the same regardless of Q Subscription type
  • Changes are captured in the same manner
  • More data may be sent depending on the subscription type and options
• Q Apply behaves the same as unidirectional EXCEPT for conflict handling.
Conflict Handling for Bidirectional Replication

- **VALUE based conflict detection:**
  - Conflict level options offered:
    - ✓ Check all columns on update - requires transmission of all old/new values
    - ✓ Check only changed columns on update - allows for column merge
    - ✓ Check only key columns

- **Resolution choices offered: Force or Ignore set at each server**
  - Force Action - requires transmission of all new values
    - force convergence on conflicts – apply the change (this is the loser!)
    - log the conflict
  - Ignore Action
    - log the conflict – do not apply the change (this is the winner!)
Value Based Conflict Detection

- Do the current row values at the apply target match the old row (before values) carried over from the source update?
- Designated site wins.

Server A (winner)

<table>
<thead>
<tr>
<th>row ((a,b,c)) changed to ((a,b,x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before values ((abc)) and after values ((abx)) are sent to Server B</td>
</tr>
</tbody>
</table>

1. Does old row \((a,b,c)\) match existing row \((a,b,x)\) ?
2. No = conflict
3. Who wins? = Server A
4. Ignore the change, Log the conflict
5. Row at Server A is \((a,b,x)\)

Server B (loser)

<table>
<thead>
<tr>
<th>row ((a,b,c)) changed to ((a,z,c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before values ((abc)) and after values ((azc)) are sent to Server A</td>
</tr>
</tbody>
</table>

1. Does old row \((a,b,c)\) match existing row \((a,z,c)\) ?
2. No = conflict
3. Who wins? = Server A
4. Force the change, Log the conflict
5. Row at Server B is \((a,b,x)\)
How Does Apply work for peer-to-peer?

• Peer-to-peer replication is supported for any number of servers. The practical limit is 6, based on development tests.

• Each server has a Q Capture and Q Apply program.

• There are multiple sets of MQ definitions on each server.
  • Multiple Queue definitions for Q Capture (one SENDQ for each peer)
  • Multiple queue definitions for Q Apply (one RECVQ for each peer)

• The Q Subscription type is Peer to Peer and the replication definitions are automatically setup through the Replication Center. Additional peer servers can be added without stopping the existing peers.

• Q Capture behaves the same regardless of Q Subscription type.
  • Changes are captured in the same manner
  • More data may be sent depending on the subscription type and options
  • Q Apply behaves the same as unidirectional EXCEPT for conflict handling.
Conflict Handling for Peer-to-Peer Replication

• Conflict detection based on timestamp

• VERSION based conflict resolution:
  • Based upon time zone adjusted timestamps, most recent timestamp “wins”
  • Each source/target table must have two extra columns to support version-based conflict handling (timestamp, tie-breaker)
    • Extra columns maintained by triggers (insert/update)
    • Replication Center adds the required columns and creates the necessary triggers to maintain those columns
    • Time zones can vary, but the machine clocks should be well synchronized
    • Triggers have impact on applications that access source/target tables
Version Based Conflict Resolution

- All rows are augmented with a “Version” = timestamp Tx and smallint Nx, indicating when and by which server the row was last updated.
- Do the current values of Tx and Nx at the apply target match the old values of Tx and Nx carried over from the source update?
- Most current timestamp Tx wins.

**Server A (N1)**

row \((a,b,c,T1,N1)\) changed to \((a,b,x,T3,N1)\)

Before values \(T1,N1\) and after values \(a,b,x,T3,N1\) are sent to Server B

1. Does old version \((T1,N1)\) match existing version \((T3,N1)\) ??
2. No = conflict
3. \(T3 > T2\), \(T3\) version wins
4. Row at Server A is \((a,b,x,T3,N1)\)

**Server B (N2)**

row \((a,b,c,T1,N1)\) changed to \((a,z,c,T2,N2)\)

Before values \(T1,N1\) and after values \(a,z,c,T2,N2\) are sent to Server A

1. Does old version \((T1,N1)\) match existing version \((T2,N2)\) ??
2. No = conflict
3. \(T3 > T2\), \(T3\) version wins
4. Row at Server B is \((a,b,x,T3,N1)\)
What happens to the conflict that loses?

- Changes that are not applied because of a conflict are logged in the IBMQREP_EXCEPTIONS control table at the target. The rejected change is stored in XML format in the control table.

- This behavior is the same for bidirectional and peer-to-peer replication.

- The Exceptions Table Formatter Utility can be used to display the exceptions in readable format.
How Does Apply work for stored procedure targets?

• Q Apply agents receive transactions the same way that they do for all other replication scenarios.

• The Q Apply agent calls the user-supplied stored procedure for each SQL statement in the transaction.

• The input to the user-supplied stored procedure is the type of operation (insert, update, delete) and the values from the changed row at the source.

• The stored procedure must not issue a COMMIT or ROLLBACK.

• The stored procedure must return an SQLCODE that indicates success or failure.

• Q Apply has no control over the stored procedure processing.
Q Apply Miscellaneous

- One Q Apply process (schema) can process multiple RECVQs.
- For Peer to Peer replication, the Q Capture and Q Apply schemas must be the same on all Peer nodes.
- Q Capture and Q Apply control tables must be on the catalog node in a DB2 for Linux, UNIX and Windows partitioned database.
- There must be a set of columns that uniquely identify a row in the target table for bidirectional and peer-to-peer processing. This can be defined as a primary key or a unique index. The uniqueness must also be enforced at the source.
- Q Apply programs running on Linux, UNIX, or Windows need a password file to provide connect information. You create this password file on the system where Q Apply will run with the asnpwd command.
What happens if Q Apply encounters an error?

The Q Apply program updates target tables with changes replicated from your source tables. If other applications are also making changes to your target tables, the Q Apply program might discover rows in the target tables that are different than it expects. For example, the Q Apply program might try to update a row that another application already deleted. Read more examples here.

**Unexpected conditions in the target data**

How should the Q Apply program respond to unexpected conditions in target data?

- Force the source change into the target table
- Do not force the source change into the target table
- Ignore the condition and continue
- Stop the corresponding Q subscription
- Stop the corresponding receive queue
- Shutdown

The Q Apply program might encounter other types of unexpected conditions while applying data to target tables, such as SQL errors.

**Other unexpected conditions**

How should the Q Apply program respond to these other unexpected conditions?

- Stop the corresponding Q subscription
- Stop the corresponding receive queue
- Shutdown
What happens if Apply encounters a data error?

If Apply cannot process an insert (row is already in target table) or update/delete (row is not in target table), the action taken depends on the values set when the Q Subscription was created:

• Force the change to the target table (this is how SQL Replication handles the problem – rework)
• Skip this change and continue
  All skipped changes are logged in the IBMQREP_EXCEPTIONS table
• Stop the Q Subscription – only the failing table is affected
• Stop the RECVQ – all tables processed by the queue are affected
• Stop the Apply program – all tables processed by this Apply are affected
What happens if Apply encounters a database error?

- If Apply cannot process a change because of some other condition (tablespace full, transaction log full, database not available), the action chosen when the Q Subscription was created is taken:
  - Stop the Q Subscription – affects only the failing table
  - Stop the RECVQ – affects all tables processed by this queue
  - Stop the Apply program – affects all tables processed by this Apply
- For unidirectional Q Subscriptions only, you can specify SQL states that you are willing to accept and Q Apply will skip any change that returns one of those states.
Operating Q Capture and Q Apply

1. Q Replication processes can be started/stopped by:
   - Replication Center (requires a Database Administration Server – DAS – running at the Q Capture and Q Apply server)
   - Line commands on Linux, UNIX, Windows – asnqcap, asnqapp
     - Windows services
     - Started tasks or batch jobs on z/OS

2. Status of Q Replication processes can be displayed by:
   - Replication Center “Check Status”
   - Line commands on Linux, UNIX, Windows – asnqccmd, asnqacmd
     - Modify command on z/OS
Q Capture Status Command with details

```
asnqccmd capture_server= qtest status show details
```

<table>
<thead>
<tr>
<th>Q Capture program status</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>(SERVER) = QTEST</td>
</tr>
<tr>
<td>Schema name</td>
<td>(SCHEMA) = ASN</td>
</tr>
<tr>
<td>Program status</td>
<td>(STATUS) = Up</td>
</tr>
<tr>
<td>Time since program started</td>
<td>(UP_TIME) = 0d 0h 1m 57s</td>
</tr>
<tr>
<td>Log file location</td>
<td>(LOGFILE) =</td>
</tr>
<tr>
<td>/vbs/dpropr/mqpub/qttest.QTEST.ASN.QCAP.log</td>
<td></td>
</tr>
<tr>
<td>Number of active Q subscriptions</td>
<td>(ACTIVE_QSUBS) = 3</td>
</tr>
<tr>
<td>Log reader currency</td>
<td>(CURRENT_LOG_TIME) = 2006-06-13-11.50.17.106870</td>
</tr>
<tr>
<td>Last committed transaction published (LSN)</td>
<td>(ALL_PUBLISHED_AS_OF_LSN) = 0000:0000:0000:6978:20E0</td>
</tr>
<tr>
<td>Current application memory</td>
<td>(CURRENT_MEMORY) = 0 MB</td>
</tr>
<tr>
<td>Path to DB2 log files</td>
<td>(DB2LOG_PATH) =</td>
</tr>
<tr>
<td>/home2/qttest/qttest/NODE0000/SQL00002/SQLOGDIR/</td>
<td></td>
</tr>
<tr>
<td>Oldest DB2 log file needed for Q Capture restart</td>
<td>(OLDEST_DB2LOG) = S0000214.LOG</td>
</tr>
<tr>
<td>Current DB2 log file captured</td>
<td>(CURRENT_DB2LOG) = S0000214.LOG</td>
</tr>
</tbody>
</table>
Q Apply Status Command with details

```
asnqacmd apply_server=qtest status show details
```

Q Apply program status

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>(SERVER) = QTEST</td>
</tr>
<tr>
<td>Schema name</td>
<td>(SCHEMA) = ASN</td>
</tr>
<tr>
<td>Program status</td>
<td>(STATUS) = Up</td>
</tr>
<tr>
<td>Time since program started</td>
<td>(UP_TIME) = 0d 0h 0m 29s</td>
</tr>
<tr>
<td>Log file location</td>
<td>(LOGFILE) = /home/tolleson/mylogs</td>
</tr>
<tr>
<td>Number of active Q subscriptions</td>
<td>(ACTIVE_QSUBS) = 2</td>
</tr>
<tr>
<td>Time period used to calculate average</td>
<td>(INTERVAL_LENGTH) = 0h 0m 0.50s</td>
</tr>
</tbody>
</table>

Receive queue: Q2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of active Q subscriptions</td>
<td>(ACTIVE_QSUBS) = 1</td>
</tr>
<tr>
<td>All transactions applied as of (time)</td>
<td>(OLDEST_TRANS) = 2005-07-30-12.52.42.000001</td>
</tr>
<tr>
<td>All transactions applied as of (LSN)</td>
<td>(OLDEST_TRANS) = 0000:0000:0000:0000:0000</td>
</tr>
<tr>
<td>Oldest in-progress transaction</td>
<td>(OLDEST_INFLT_TRANS) = 2005-07-30-12.52.42.000001</td>
</tr>
<tr>
<td>Average end-to-end latency</td>
<td>(END2END_LATENCY) = 0h 0m 1.476s</td>
</tr>
<tr>
<td>Average Q Capture latency</td>
<td>(CAPTURE_LATENCY) = 0h 0m 0.661s</td>
</tr>
<tr>
<td>Average W&amp;MQ latency</td>
<td>(Q_LATENCY) = 0h 0m 0.786s</td>
</tr>
<tr>
<td>Average Q Apply latency</td>
<td>(APPLY_LATENCY) = 0h 0m 0.29s</td>
</tr>
<tr>
<td>Current memory</td>
<td>(CURRENT_MEMORY) = 0 MB</td>
</tr>
<tr>
<td>Current queue depth</td>
<td>(QDEPTH) = 92</td>
</tr>
</tbody>
</table>
Change Data Capture (CDC)
High level architecture

Sources
- Oracle
- SQL Server
- DB2
- Informix
- Sybase

Journal Log
Redo/Archive Logs

Source Engine
and Metadata

Java-based GUI
for admin and monitoring

TCP/IP

Target Engine
and Metadata

Targets
- Database
  (Oracle, DB2, SQL Server, Teradata, etc.)
- ETL (DataStage, others)
- JMS (MQ, others)
- Information Server
  (DataStage, QualityStage, etc.)
- Flat files
- Web Services
Expansive source, target, platform support

<table>
<thead>
<tr>
<th>DATABASES Source and Target</th>
<th>TARGETS</th>
<th>NETWORK PROTOCOL</th>
<th>MESSAGE QUEUE</th>
<th>OPERATING SYSTEMS</th>
<th>HARDWARE PLATFORMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 z/OS®</td>
<td>Teradata</td>
<td>TCP/IP</td>
<td>JMS</td>
<td>IBM i OS</td>
<td>IBM i OS</td>
</tr>
<tr>
<td>Oracle</td>
<td>Information Server</td>
<td></td>
<td>MQ Series</td>
<td>z/OS</td>
<td>IBM System z®</td>
</tr>
<tr>
<td>Sybase</td>
<td>Cognos Now!</td>
<td></td>
<td>TIBCO</td>
<td>AIX®</td>
<td>IBM System p®</td>
</tr>
<tr>
<td>MS SQL Server</td>
<td>Netezza*</td>
<td></td>
<td>WebMethods</td>
<td>HP-UX</td>
<td>HP PA-RISC</td>
</tr>
<tr>
<td>DB2 UDB</td>
<td>Greenplum*</td>
<td></td>
<td>BEA</td>
<td>Solaris</td>
<td>HP Itanium</td>
</tr>
<tr>
<td>DB2 i</td>
<td>MySQL*</td>
<td></td>
<td></td>
<td>MS Windows</td>
<td>RedHat, SUSE Linux</td>
</tr>
<tr>
<td>Informix®</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solid DB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Customized solution, limited requirements*
Flexible implementation

- Uni-directional
- Bi-directional
- Two-way
- Remote capture
- Cascade
- Local
- Multi-thread
- Consolidation
- Distribution

© 2013 IBM Corporation
Easy to use

- Java-based GUI for configuration, administration and monitoring
  - Manage data integration processes from one screen
  - Automatic mapping, drag-and-drop transformations
  - No programming required
  - Event logs, alerts and alarms and statistics reporting
Integration scenario 1: Heterogeneous databases
Integration scenario 2: Database to flat file

Oracle engine

Management console

Flat files for ETL/BI consumption

Log scrape

TCP/IP

Oracle

InfoSphere CDC

ETL / BI

InfoSphere DataStage engine
Integration scenario 3: Database to message queue

Oracle engine

Message queue or topic

XML

ESB / ETL / BI

Log scrape

TCP/IP

InfoSphere CDC

Oracle engine

Event Server engine

Management console
Integration scenario 4: InfoSphere Change Data Capture/InfoSphere DataStage integration

Oracle

Management console

InfoSphere DataStage
InfoSphere QualityStage

Log scrape

Direct TCP/IP connection

Oracle engine

InfoSphere DataStage engine

TCP/IP
Modes of replication

- Continuous mirroring
  - Changes read from database log.
  - Apply change at the target as soon as it is generated at the source.
  - Replication job remains active waiting for next available log entry.

- Periodic mirroring
  - Changes read from database log.
  - Apply net changes on a scheduled basis.
  - Replication job ends when available log entries are processed.

- Refresh
  - File/table level operation.
  - Apply a snapshot version of source table.
  - Typically used to achieve initial synchronization of source and target table.
Subset refresh and differential refresh

- Common uses for subset refresh functionality
  - Refreshing very large tables in stages
    - Accommodating smaller batch windows
    - Less interruption for other tables being replicated
    - Example:
      - Refreshing a table of one billion rows can be spread over multiple days
      - Every day 200 million rows can be refreshed
  - Refreshing a day's worth of changes, if there is a column that contains change date
  - Synchronization check for subset of rows
    - Using differential refresh functionality

- Differential refresh
  - Allows for refreshing/checking rows with discrepancies
  - This function may also be used to perform a synchronization check
Table mapping methods

- One-to-one
  - Source and target tables have similar table structures
- LiveAudit™
  - Generates audit trail of data transactions from source
- Adaptive Apply
  - Automatically synchronizes data for dissimilar sources and targets
- Summarization
  - Keeps a running total of numerical values at the target
- Consolidation: One-to-One
  - Merges data from several tables into a single row
- Consolidation: One-to-Many
  - Used to apply a source lookup table change to all affected target rows
## Filtering

- Integrate entire systems or only a subset of data
- Table/row/column-level filtering options available

<table>
<thead>
<tr>
<th>CUST_NO</th>
<th>L_NAME</th>
<th>F_NAME</th>
<th>PHONE</th>
<th>REP_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>58699</td>
<td>Smith</td>
<td>John</td>
<td>404-555-3874</td>
<td>45</td>
</tr>
<tr>
<td>37283</td>
<td>Duggan</td>
<td>Ira</td>
<td>613-555-8367</td>
<td>25</td>
</tr>
<tr>
<td>89863</td>
<td>Quinn</td>
<td>Fran</td>
<td>905-555-1296</td>
<td>11</td>
</tr>
<tr>
<td>89732</td>
<td>Muntz</td>
<td>Muntz</td>
<td>704-555-2738</td>
<td>25</td>
</tr>
</tbody>
</table>

**ROW SELECT**

<table>
<thead>
<tr>
<th>CUST_NO</th>
<th>L_NAME</th>
<th>F_NAME</th>
<th>REP_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>37283</td>
<td>Duggan</td>
<td>Ira</td>
<td>25</td>
</tr>
<tr>
<td>89732</td>
<td>Muntz</td>
<td>Josie</td>
<td>25</td>
</tr>
</tbody>
</table>
Transformations and data translations

<table>
<thead>
<tr>
<th>EMP</th>
<th>LAST</th>
<th>FIRST</th>
<th>HIRE_DATE</th>
<th>STAT</th>
<th>SALARY</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Moreiro</td>
<td>Nicole</td>
<td>01/05/97</td>
<td>A</td>
<td>$55,000</td>
<td>$60,000</td>
</tr>
<tr>
<td>2345</td>
<td>Ellison</td>
<td>Val</td>
<td>04/12/97</td>
<td>I</td>
<td>$40,000</td>
<td>$50,000</td>
</tr>
</tbody>
</table>

Increase field size
Concatenation
Century dates
Transform fields
Derived fields

<table>
<thead>
<tr>
<th>EMP_ID</th>
<th>FULL_NAME</th>
<th>HIRE_DATE</th>
<th>STATUS</th>
<th>%SALARYMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>001234</td>
<td>Nicole Moreiro</td>
<td>01/05/1997</td>
<td>Active</td>
<td>92%</td>
</tr>
<tr>
<td>002345</td>
<td>Val Ellison</td>
<td>04/12/1997</td>
<td>Inactive</td>
<td>80%</td>
</tr>
</tbody>
</table>
## Character encoding/translations

- Automatic data type mappings for character conversions
Auditing

• Switch all operations into INSERT to keep transactional history
• Capture additional data for full data traceability
  • Type of data change, origin of data change, etc.

JOURNAL CONTROL COLUMNS

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;CCID</td>
<td>An identifier for the transaction with the update.</td>
</tr>
<tr>
<td>&amp;CNTRRN</td>
<td>Source table relative record number</td>
</tr>
<tr>
<td>&amp;CODE</td>
<td>Always “U” for refresh. Always “R” for mirror.</td>
</tr>
<tr>
<td>&amp;ENTTYP</td>
<td>Indicates the type of update.</td>
</tr>
<tr>
<td>&amp;JOB</td>
<td>The name of the source job that made the update.</td>
</tr>
<tr>
<td>&amp;JOBNO</td>
<td>The operating system user Id of the update process.</td>
</tr>
<tr>
<td>&amp;JOBUSER</td>
<td>The operating system user at the time of the update.</td>
</tr>
<tr>
<td>&amp;JOURNAL</td>
<td>The name of the journal, as described in Properties.</td>
</tr>
<tr>
<td>&amp;JRNFLG</td>
<td>Indicates if before image is present</td>
</tr>
<tr>
<td>&amp;JRNLIB</td>
<td>The name of the journal schema.</td>
</tr>
<tr>
<td>&amp;LIBRARY</td>
<td>The source table schema or its alias.</td>
</tr>
<tr>
<td>&amp;MEMBER</td>
<td>The source table name or its alias.</td>
</tr>
<tr>
<td>&amp;PROGRAM</td>
<td>The name of source program that made the update.</td>
</tr>
<tr>
<td>&amp;OBJECT</td>
<td>The source table name or its alias.</td>
</tr>
<tr>
<td>&amp;SEQNO</td>
<td>The sequence number of this update in the journal.</td>
</tr>
<tr>
<td>&amp;SYSTEM</td>
<td>The hostname of the source system</td>
</tr>
<tr>
<td>&amp;TIMESTAMP</td>
<td>Time of the update or refresh.</td>
</tr>
<tr>
<td>&amp;USER</td>
<td>The user ID which made the update.</td>
</tr>
</tbody>
</table>
User exits

• Execute custom business logic
  • React to business events in real time
  • Automate business processes
• Multiple implementation methods available:
  • C/C++, Java, stored procedures
Conflict detection and resolution

• Provides data integrity when multiple systems change the same data simultaneously
• Conflicts can be resolved in various ways:
  • Source wins, target wins
  • By data value
  • Execute user exit
Change management

- Promote test and development integration processes into production without risk
- Eliminates potential user error
- Enables faster rollout of new business processes
- Rollback capabilities available
- Changes are tracked for compliance
Monitoring

- Graphical visualization of replication processes
- Detailed performance statistics to optimize tuning
- Event logs, alerts and alarms
- Exportable throughput and latency statistics
Exceptional data integrity

• Data transactions are applied at the target in the same order as it was generated at the source
• Target acknowledges each apply operation to ensure delivery
• No data is lost even if communications link becomes unavailable
• Automatic restart of replication processes after a network failure
Reliability and recoverability

1. Scrape

Journal/Log

Source Tables

2. Push

Push Engine

3a. Apply

Apply Engine

Target Tables

Metadata
What was the last transaction you successfully applied?

3b. Confirm
Persistency

- InfoSphere CDC may initiate a normal shutdown and end mirroring after:
  - Communications error
  - Instance termination
  - Deadlock scenarios

- To automatically restart continuous mirroring of subscriptions after a normal shutdown, you can mark the subscriptions as persistent
- InfoSphere CDC will attempt to automatically restart continuous mirroring at regular intervals
- Continuous mirroring for a persistent subscription can automatically restart in response to a normal or abnormal (recoverable) termination for the above mentioned conditions
InfoSphere CDC → DataStage/QualityStage integration

**Option 1: Database Staging**
- InfoSphere CDC captures change made to source database
- InfoSphere CDC writes changes to a staging table.
- DataStage reads the changes from the staging table, transforms and cleans the data as needed
- Update target database and internal tracking with last bookmark processed

**Option 2: MQ-based integration**
- InfoSphere CDC captures/collects changes made to remote database
- Captured changes written to MQ
- DataStage (via MQ connector) processes message and passes data off to downstream stages
- Updates written to target warehouse

**Option 3: File-based**
- InfoSphere CDC captures change made to source database
- InfoSphere CDC writes each transaction to a file
- DataStage reads the changes from the file
- Update target database with changes

**Option 4: Direct connect**
- InfoSphere CDC captures and collects changes made to source
- Captured changes passed to CDC for DataStage engine
- DataStage transaction aware stage processes transactions and passes data off to downstream stages
- Update target database with changed data
Direct connect

- Custom operator, which runs continuously, requests the changed data from CDC
- CDC captures/collects changes made to source database
- Captured changes passed via direct connection to transaction stage
- Custom transaction stage passes data off to downstream stages
- Update target database with changed data
InfoSphere Change Data Capture - Recap

• InfoSphere Change Data Capture provides real-time changed data capture across the enterprise.

• Key benefits:
  • Low impact
    • Does not impact performance and requires no changes to applications
  • Heterogeneous
    • Integrates data from all platforms and databases
  • Flexible
    • Supports any topology
  • Easy to use
    • Fast deployment with low risk
  • Integrated with Information Server
    • Single solution for all data integration requirements
Replication Tools
Monitoring Q Replication
The BIG Picture

ADMINISTRATION

SOURCE
SOURCE1
SOURCE2

DB2 Log → Q Capture

Q Capture → Replication Monitor

Replication Monitor → Replication Center

APPLICATION

SOURCE
SOURCE1
SOURCE2

METADATA

Q Apply Browser

Q Apply Agent

Apply Agent

Apply Agent

TARGET

TGT1

TGT2
Viewing Reports in the Replication Center
Q Capture Activity Reporting

1. Q Capture stores runtime statistics in the control tables at the source server
   1. IBMQREP_CAPMON and IBMQREP_CAPQMON
   2. The value for MONITOR_INTERVAL in the IBMQREP_CAPPARMS table determines how often Capture inserts to the monitor tables.
   3. The value for MONITOR_LIMIT in the IBMQREP_CAPPARMS table determines how much monitor data is kept.

2. Q Capture stores informational, warning, and error messages in
   1. IBMQREP_CAPTRACE table at the source server
   2. The value for TRACE_LIMIT in the IBMQREP_CAPPARMS table determines how much trace information is kept.
   3. Q Capture log file at the source server
# Q Capture Monitor Tables

**IBMQREP_CAPMON**

Statistics on log records processed

<table>
<thead>
<tr>
<th>MONITOR_TIME</th>
<th>ROWS_PROCESSED</th>
<th>TRANS_PROCESSED</th>
</tr>
</thead>
</table>

**IBMQREP_CAPQMON**

Statistics on groups of subscriptions (SENDQ)

<table>
<thead>
<tr>
<th>MONITOR_TIME</th>
<th>SENDQ</th>
<th>ROWS_PUBLISHED</th>
<th>TRANS_PUBLISHED</th>
</tr>
</thead>
</table>
## Q Capture Throughput Report

<table>
<thead>
<tr>
<th>Monitor Time</th>
<th>Send Queue</th>
<th>Rows Published</th>
<th>Transactions Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-05-30 20:46:44.843000</td>
<td>SYSA SENDQ</td>
<td>8,565</td>
<td>2,855</td>
</tr>
<tr>
<td>2007-05-30 20:46:34.859000</td>
<td>SYSA SENDQ</td>
<td>5,841</td>
<td>1,947</td>
</tr>
<tr>
<td>2007-05-30 20:46:04.781000</td>
<td>SYSA SENDQ</td>
<td>5,934</td>
<td>1,978</td>
</tr>
<tr>
<td>2007-05-30 20:45:54.765000</td>
<td>SYSA SENDQ</td>
<td>6,282</td>
<td>2,094</td>
</tr>
<tr>
<td>2007-05-30 20:45:44.734000</td>
<td>SYSA SENDQ</td>
<td>6,510</td>
<td>2,170</td>
</tr>
<tr>
<td>2007-05-30 20:45:34.718000</td>
<td>SYSA SENDQ</td>
<td>5,424</td>
<td>1,808</td>
</tr>
<tr>
<td>2007-05-30 20:45:24.687000</td>
<td>SYSA SENDQ</td>
<td>4,536</td>
<td>1,512</td>
</tr>
<tr>
<td>2007-05-30 20:45:14.671000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:45:04.656000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:44:54.640000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:44:44.625000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:44:34.625000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:44:24.609000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:44:14.593000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:44:04.578000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:43:54.578000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:43:44.546000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:43:34.531000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:43:24.515000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:43:14.500000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:42:54.484000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:42:44.453000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-05-30 20:42:34.437000</td>
<td>SYSA SENDQ</td>
<td>3,114</td>
<td>1,038</td>
</tr>
<tr>
<td>2007-05-30 20:42:14.390000</td>
<td>SYSA SENDQ</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Q Capture Latency Report

The image shows a window displaying a table with two columns: Monitor Time and Q Capture Latency. The table contains rows of dates and corresponding latency values. The dates range from 2007-05-30 to 2007-06-02, and the latency values range from 2 to 248. The bottom of the window indicates that 163 of 163 items are displayed.
Q Apply Activity Reporting

1. Q Apply stores runtime statistics in the control tables at the target server
   1. IBMQREP_APPLYMON
   2. The value for MONITOR_INTERVAL in the IBMQREP_APPLYPARMS table determines how often Q Apply inserts to the monitor tables.
   3. The value for MONITOR_LIMIT in the IBMQREP_APPLYPARMS table determines how much monitor data is kept.

2. Q Apply stores informational, warning, and error messages in
   1. IBMQREP_APPLYTRACE table at the target server
   2. The value for TRACE_LIMIT in the IBMQREP_APPLYPARMS table determines how much trace information is kept.
   3. Q Apply log file at the target server
Q Apply Monitor Table

IBMQREP_APPLYMON

Statistics on groups of subscriptions (RECVQ)

| MONITOR_TIME | RECVQ | ROWS_APPLIED | TRANS_APPLIED |
### Q Apply Throughput

<table>
<thead>
<tr>
<th>Monitor Time</th>
<th>Receive Queue</th>
<th>Transactions Applied to Target</th>
<th>Rows Applied to Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-06-30 20:47:51 895000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:47:41 843000</td>
<td>SYSB.RECVQ</td>
<td>384</td>
<td>1,152</td>
</tr>
<tr>
<td>2007-06-30 20:47:31 826000</td>
<td>SYSB.RECVQ</td>
<td>601</td>
<td>1,603</td>
</tr>
<tr>
<td>2007-06-30 20:47:21 796000</td>
<td>SYSB.RECVQ</td>
<td>449</td>
<td>1,347</td>
</tr>
<tr>
<td>2007-06-30 20:47:11 781000</td>
<td>SYSB.RECVQ</td>
<td>675</td>
<td>2,025</td>
</tr>
<tr>
<td>2007-06-30 20:47:01 765000</td>
<td>SYSB.RECVQ</td>
<td>518</td>
<td>1,554</td>
</tr>
<tr>
<td>2007-06-30 20:46:51 750000</td>
<td>SYSB.RECVQ</td>
<td>594</td>
<td>1,782</td>
</tr>
<tr>
<td>2007-06-30 20:46:41 734000</td>
<td>SYSB.RECVQ</td>
<td>979</td>
<td>2,937</td>
</tr>
<tr>
<td>2007-06-30 20:46:21 703000</td>
<td>SYSB.RECVQ</td>
<td>921</td>
<td>2,763</td>
</tr>
<tr>
<td>2007-06-30 20:46:11 687000</td>
<td>SYSB.RECVQ</td>
<td>921</td>
<td>2,763</td>
</tr>
<tr>
<td>2007-06-30 20:46:01 671000</td>
<td>SYSB.RECVQ</td>
<td>802</td>
<td>2,286</td>
</tr>
<tr>
<td>2007-06-30 20:45:51 703000</td>
<td>SYSB.RECVQ</td>
<td>901</td>
<td>2,703</td>
</tr>
<tr>
<td>2007-06-30 20:45:41 640000</td>
<td>SYSB.RECVQ</td>
<td>871</td>
<td>2,613</td>
</tr>
<tr>
<td>2007-06-30 20:45:31 625000</td>
<td>SYSB.RECVQ</td>
<td>783</td>
<td>2,349</td>
</tr>
<tr>
<td>2007-06-30 20:45:21 609000</td>
<td>SYSB.RECVQ</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2007-06-30 20:45:11 593000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:45:01 578000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:44:51 562000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:44:41 546000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:44:31 531000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:44:21 515000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:44:11 500000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:44:01 484000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:43:51 468000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:43:41 453000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2007-06-30 20:43:31 437000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

426 of 426 items displayed.
Latency

<table>
<thead>
<tr>
<th>Monitor Time</th>
<th>Receive Queue</th>
<th>End-to-End Latency</th>
<th>Queue Latency</th>
<th>Q Apply Latency</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-05-30 20:46:51.750000</td>
<td>SYSB.RECVQ</td>
<td>17.923</td>
<td>12.043</td>
<td>1,930</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:46:31.718000</td>
<td>SYSB.RECVQ</td>
<td>17.434</td>
<td>1,046</td>
<td>12,634</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:46:01.671000</td>
<td>SYSB.RECVQ</td>
<td>13.456</td>
<td>1,154</td>
<td>8,993</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:45:51.703000</td>
<td>SYSB.RECVQ</td>
<td>12.492</td>
<td>1,713</td>
<td>7,062</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:45:11.593000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:45:01.578000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:44:51.562000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:44:41.546000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:44:31.531000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:44:21.515000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:44:11.500000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:44:01.484000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:43:51.468000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:43:41.453000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:43:31.437000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2007-05-30 20:43:21.421000</td>
<td>SYSB.RECVQ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

426 of 426 items displayed
Q Replication dashboard

Real-time monitoring tool can be downloaded from the web
Dashboard can drill down to the table level:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>State</th>
<th>Type</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TJACOPI.TOM1</td>
<td>Active</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJACOPI.TOM2</td>
<td>Active</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJACOPI.TOM3</td>
<td>Active</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJACOPI.TOM3</td>
<td>Inactive</td>
<td>XML Pub</td>
<td></td>
</tr>
<tr>
<td>TJACOPI.TOM5</td>
<td>Inactive</td>
<td>Uni (Tgt)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJACOPI.TOM8</td>
<td>Active</td>
<td>P2P</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJACOPI.TOM9</td>
<td>Partially active</td>
<td>P2P</td>
<td>RED2(ASN), GREEN2(ASN)</td>
</tr>
<tr>
<td>TJACOPI.TOM9</td>
<td>Inactive</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
</tbody>
</table>
The Data Studio Administration Console

1. New Web-based monitoring and administration tool for Q Replication
The Replication Alert Monitor

The Replication Alert Monitor is a replication program (asnmon) that runs continuously, checking Capture and Apply servers for error and warning conditions chosen by the Replication Administrator.

Alerts are sent as e-mail messages via an SMTP mail server. The Replication Administrator identifies the contacts or group of contacts to be notified for each condition.

Alerts are also logged in an Alert Monitor control table.
What are the Alert Monitor requirements?

1. Alert Monitor platforms
   • DB2 Universal Database for OS/390 and z/OS V7 or V8
   • DB2 Universal Database for Linux, UNIX and Windows 9.1

2. Alert Monitor prerequisites
   • DB2 (DRDA) connectivity to monitored Q Capture and Q Apply servers
   • Database Administration Services (DAS) installed and configured on all monitored Q Capture and Q Apply servers
     • Required to determine whether the replication programs are running
       Not required for other monitoring elements
     • Requires additional installs of code for DB2 Universal Database for OS/390 and z/OS
   • Database Administration Services (DAS) installed and configured on the server where the Alert Monitor runs
     • Required to send e-mail notification
How Does the Alert Monitor work?

1. The Replication Center is used to define alert monitoring for Q Capture and Q Apply.

2. The Replication Administrator performs the following operations
   • Create Monitor Control Tables
   • Set thresholds and choose events for alert monitoring
   • Identify contacts and groups of contacts for notification via an SMTP mail server
   • Start and stop the Alert Monitor

3. The Replication Center generates SQL and replication command scripts to stop and start the Alert Monitor and to check status. The scripts are run using the Replication Center and can also be saved for reference or reuse.
What events/conditions can be monitored?

1. Q Capture and Q Apply status – alert if programs not running
2. Q Capture and Q Apply error or warning messages – alert if a message is generated by the programs
3. Latency – alert if administrator-set thresholds are reached for Q Capture, Q Apply, or End-to-End latency
4. Memory usage – alert if memory usage exceeds administrator-set threshold
5. Transaction size – alert if a transaction takes more memory than the threshold set by the administrator
6. Queue Depth – alert if number of messages on the receive queue exceeds threshold set by administrator
7. Exception – alert if Q Apply processes an exception (skipped change, conflict skipped, other error)
Where does the Alert Monitor get information?

Source Server
- Q Capture control tables
  - IBMQREP_CAPMON
  - IBMQREP_CAPQMON
  - IBMQREP_CAPTRACE
- DB2 DAS

Target Server
- Q Apply control table
  - IBMQREP_APPLYMON
  - IBMQREP_APPLYTRACE
- DB2 DAS

Monitor Server
- IBMSNAP_MONSERVERS
- IBMSNAP_CONDITIONS
- IBMSNAP_CONTACTS
- IBMSNAP_GROUPS
- IBMSNAP_ALERTS
Q Replication Utilities and System Commands

1. System commands for operating Q Capture, Q Apply, and the Alert Monitor

2. Troubleshooting commands and utilities for diagnosing problems
System Commands

1. Q Capture
   • asnqcap – start Q Capture
   • asnqccmd – work with a running Q Capture program, including query status and stop

2. Q Apply
   • asnqapp – start Q Apply
   • asnqacmd – work with a running Q Apply program, including query status and stop

3. Monitor
   • asnmon – start the Alert Monitor
   • asnmcmd – work with a running Alert Monitor program, including query status and stop
More System Commands

1. Q Capture and Q Apply on Windows
   • asncrt – create Windows services for replication programs
   • asnlist – list Windows services created by asncrt
   • asndrop – remove Windows services created by asncrt

2. Q Apply Password management on DB2 for Linux, UNIX, and Windows
   • asnpwd – create an encrypted list of passwords for Q Apply to use when doing a full refresh
Invoking System Commands

1. **System commands are invoked from the command line on**
   - DB2 Universal Database for Linux, UNIX, and Windows
   - UNIX System Services shell on z/OS
   - Modify command on z/OS

2. **Sample syntax**
   - Linux, UNIX, Windows
     - asnqcap capture_server=DB2SYSA capture_schema=ASN
     - asnqacmd apply_server=DB2SYSB apply_schema=ASN1 stop
   - Z/OS
     - /F Qcapstartedtask,STOP
Replication Utilities

1. **asnqanalyze**
   - Detailed or summary report of replication environment

2. **asntdiff**
   - Compares a replication source table to a replication target table and reports the differences

3. **asntrep**
   - Uses the information from asntdiff to reconcile differences between a replication source and target table

4. **asntrc**
   - Traces Q Capture, Q Apply, and Alert Monitor programs

5. **asnqmfmt**
   - Formats replication messages for troubleshooting
Utility -- asnqanalyze

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows

2. DB2 connectivity to Q Capture or Q Apply servers is required
   - DB2 Connect is needed if any of the servers are DB2 on z/OS
   - Password file is required for connection to remote servers
     The asnpwd system command is used to create the password file

3. Output is an HTML report

   Example:
   asnqanalyze –db DB2SYSA –la detailed
   asnqanalyze with no parameters will display help
Utility asnqanalyze Sample Report

IBM DataPropagator ™
Q-replication analyzer

ANALYZER - Build date: Jun 16, 2004

It is called at Fri Oct 22 11:12:51 with the following parameters:

asnqanalyze -db db2sysa -la detailed

Table Of Contents

DB2SYSA Control table details.
Database / Platform connection summary.
Missing table or view references.
Inconsistency - orphan details.
Incorrect or inefficient of indexes.
Utility -- asntdiff

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows or z/OS USS

2. DB2 connectivity to Q Capture and Q Apply servers is required
   • DB2 Connect is needed if any of the servers are DB2 on z/OS
   • Password file is required for connection to remote servers
     The asnpwd system command is used to create the password file

3. Output is a DB2 table named ASN.TDIFF at the Q Capture server
   • The difference table, ASN.TDIFF, must be manually dropped

EXAMPLE:

```sql
db2 connect to <Qcapdatabase>
db2 drop table asn.tdiff
asntdiff db=DB2SYSA where='SUBNAME='CUSTMER0001””
```
How does asntdiff work?

1. Uses the SUBNAME in the WHERE parameter to find the column mappings and search conditions (predicate) in the Q Capture control tables.

2. Creates ASN.TDIFF based on the column mappings

3. Selects rows from the source table based on the column mappings and search conditions

4. Compares the source table rows to the target table (check sum scheme is used for efficiency)

5. Puts differences in the ASN.TDIFF table

6. Writes messages to the console
Utility -- asntrep

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows or z/OS USS

2. DB2 connectivity to Q Capture and Q Apply servers is required
   • DB2 Connect is needed if any of the servers are DB2 on z/OS
   • Password file is required for connection to remote servers
     The asnpwd system command is used to create the password file

3. Input is the DB2 table named ASN.TDIFF at the Q Capture server
   • The difference table is created by the asntdiff utility.

   EXAMPLE:
   
   db2 connect to <Qcapdatabase>
   asntrep db=DB2SYSA where='SUBNAME='CUSTMER0001'"
How does asntrep work?

1. Uses the SUBNAME in the WHERE parameter to find the column mappings in the Q Capture and Q Apply control tables.
2. Selects from ASN.TDIFF to get the differences
3. Inserts missing rows to the target
4. Deletes extra rows from the target
5. Updates mismatched rows in the target
6. Writes messages
Utility - asntrc

1. Traces Q Capture, Q Apply, or Monitor programs
2. Dynamically turned on and off while traced programs are running
3. Used at the direction of IBM Support to diagnose problems

EXAMPLE
asntrc on –db DB2SYSA –schema ASN –qcap
asntrc fmt –db DB2SYSA –schema ASN -qcap
asntrc off –db DB2SYSA –schema ASN –qcap
Utility -- asnqmfmt

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows or z/OS USS

2. WebSphere MQ client or server access to the Q Apply receive queue (RECVQ)

3. asnqmfmt gets a message from the Q Apply receive queue and presents it in XML format. This is not an application interface – it is intended for diagnostics.

EXAMPLE:
asnqmfmt SYSB.RECVQ QMSYSB
Replication Uses Cases
Customers Require Different Types Of Data Integration

Each type is like a different tool: hammer, wrench, screwdriver, and saw

Federation

Analytical & Reporting Tools

Federation

Real-time Inventory Level

Product Performance

Consolidation

Data Warehouse

Consolidation

Region 1 Product Performance

Region 2 Product Performance

Data Event Publishing

Capture Publish

Database

EAI Repl ETL RYO

Production Live Copy

Replication

Replication

Live Copy
Many Usage Scenarios For Replication

High Availability

Rollup (many to 1)

Load Balancing

Distribution (1 to many)
Replication Tips and Best Practices
QUESTIONS
LUW Database Replication

Brian Mealman
612-397-5468
bamealm@us.ibm.com
Agenda

• Replication Product Family Overview & History
• Packaging
• SQL Replication
• Queue Replication (QREP)
• Change Data Capture (CDC)
• Replication Tools
• Replication Uses Cases
• Replication Tips and Best Practices
• Q&A
Replication Product Family Overview & History
Virtual Data Delivery is typically used when the amount of information to be shared from the serving data source is limited and there are no limitations to accessing this data at the time that it is needed. Virtual data delivery requires access to the data sources holding the information at the time the data is requested.

Bulk Data Delivery

Bulk data delivery lends itself to periodic updates of information stores from the various data sources and is capable of handling large volumes of data to be delivered at once. When talking about Bulk Data Delivery you would typically use ETL (Extract Transform Load) solutions such as InfoSphere DataStage to load the data from the data sources and deliver them in bulk to the information stores.

Incremental Data Delivery

Change Data Capture (cdc)

An industry recognized integration technique that uses native database transaction logs or journals to capture insert, update and delete operations. Change Data Capture (cdc) is an asynchronous push technology which allows users to deliver changing OLTP data to consumer applications on a near real-time or period basis. IBM has a market leading portfolio of data integration products that leverage Incremental Data Delivery from a wide variety of DB's and platforms.
There are 3 key elements of InfoSphere Change Data Capture:

**Low impact**
Changed data access by traditional ETL process often comes with high source database and CPU utilization. After all, it may not be straightforward to identify the changed rows from the tables. InfoSphere CDC lowers the costs of retrieving these “change records” by obtaining the information from the native database logs. The database logs are already there as part of the database’s recovery mechanism; CDC utilizes them to detect the changes which have been made to the tables in scope.

For the implementation of incremental data delivery with CDC there is no need to change the business applications and table structures. For most database engines, CDC does not place triggers on the source tables (the one exception is CDC for Oracle Trigger-based) to detect the changes: it utilizes the native database logs.

**Latency**
Data warehouse environments are typically not updated very frequently due to the costs associated with re-retrieving the information from the source systems. Although IT may own the production (business application) systems, it is usually the business users who determine when ETL processes can be run so that they do not impact the business users. CDC allows businesses to provide incremental data in near real-time with minimal or no impact to the business users (only changed data is retrieved from the database logs). By doing this, business intelligence users can retrieve their reports with a much lower latency than they are used to.

**Consistent data delivery**
InfoSphere CDC uses a “push” technology, starting from the database logs and pushing the incremental changes to the target systems and databases. As transactions are performed by the business application, these transactions get written into the database logs, from where they are picked up by CDC. CDC will only send committed transactions (units of work) to the target systems and will keep the commit order on the target side. This allows for synchronization of tables that have referential integrity constraints attached to them.

If a row is updated multiple times within the same transaction or in different transactions, CDC is aware of all the individual changes and will send them to the target as separate items. This kind of granularity is often important for customers for auditing purposes.

Should replication be ended abnormally, for example because of a network failure, it will not result in lost or duplicated transactions. CDC keeps a bookmark on the target, that is updated in the same unit of work as the changes are applied to the source. If replication is restarted, CDC will restart from the last successfully applied transaction.
Incremental Data Delivery by InfoSphere Change Data Capture always starts with a source database that supports transaction logging. Changes are read (scraped) from the native database logs and then published to one or more consuming applications (targets). Reading the changes natively from the logs ensures that CDC has a minimal impact on the source application.

There are 3 main categories of consuming targets:

- **RDBMS**: Target is another database management system (for example, when replicating to a DB2 LUW database).

- **Message queue**: Changes are replicated as XML messages to a JMS compliant message oriented middleware application or ESB (Enterprise Service Bus). Placing messages on a queue or bus opens up integration with a high variety of target applications such as MDM systems or other 3rd party applications.

- **ETL**: Changes can be landed in flat files and can then be processed by any ETL solution that supports flat files (most do), including InfoSphere DataStage or QualityStage. When targeting DataStage or QualityStage, InfoSphere CDC can directly connect to these integration solutions and replicate changes on a continuous basis, delivering change in near real-time to the targeted database or application.
Replication Server Product Offerings

• DB2 Linux UNIX & Windows:
  1. **SQL Replication** (aka DB2 Data Propagator - since 1994)
     * Data changes captured by reading the logs on DB2 z/OS, Linux, Unix, Windows (LUW), and iSeries
     * DB2 to DB2/Informix replication comes with the DB2 database server
     * Captured changes are stored in relational tables, the Apply process fetches them over a database connection
     * Oracle, Sybase, SQL Server, Informix Teradata targets can be updated

• **InfoSphere Replication Server LUW:**
  1. Q Replication and Q Data Event Publishing (since 2004)
     * Data captured by reading the logs for DB2 z/OS, DB2 LUW, and Oracle
     * Captured changes are delivered via WebSphere MQ
     * Parallel Q Apply for performance - best in the market
     * Oracle, Sybase, SQL Server, Informix Teradata targets can be updated

The Capture and Utility programs, as well as the administration interfaces are common to both SQL and Q Replication technologies

• Some functions only available in Q Replication today (e.g., XML data, Replication Dashboard)

Also packaged with Replication Server (restricted licenses for replication use only)

• WebSphere MQ; InfoSphere Federation Server; DB2 (on distributed only, not on z/OS)

• **InfoSphere Change Data Capture (CDC)**
  1. IBM Acquires Data Mirror in 2007, the products are renamed InfoSphere CDC
     * Heterogeneous Replication solutions across many platforms and databases
     * Log based capture
     * TCP/IP data transport
     * Windows Client for Administration and Management

Discovering the value of IBM InfoSphere CI
Replication Server Product Offerings

- **The DB2 (LUW) Homogeneous Replication Feature** is for DB2 --> DB2 replication only
  - Contains both Q and SQL replication
  - No Federation Server and no Event Publisher capability
- **InfoSphere Data Replication (2010)**
  1. A bundle of all the IBM Replication Technologies
  2. The following are Supporting Programs licensed with the Program:
     - IBM DB2 Enterprise Server Edition v10.1
     - IBM InfoSphere Change Data Capture v6.2
     - IBM InfoSphere Change Data Capture v6.5.2
     - IBM InfoSphere Federation Server v10.1
     - IBM InfoSphere Replication Server v9.7
     - IBM WebSphere MQ v7.1
Replication Server Product History

- A long history of IBM as the Leader of Replication Technologies:
  1. SQL Replication (aka DB2 Data Propagator) - since 1994
  2. Q Replication and Q Data Event Publishing - since 2004
- Thousands of customers world-wide
- Replication Server can run with any down-level versions of DB2 (e.g., Q Rep V9.7 with DB2 z V7)

<table>
<thead>
<tr>
<th>Release/Version</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>DpropR V1</td>
<td>1994</td>
</tr>
<tr>
<td>DpropR V5</td>
<td>1997</td>
</tr>
<tr>
<td>DpropR V6</td>
<td>1999</td>
</tr>
<tr>
<td>DpropR V7</td>
<td>2000</td>
</tr>
<tr>
<td>WebSphere II V8.2</td>
<td>2000</td>
</tr>
<tr>
<td>InfoSphere Replication Server V9.1</td>
<td>2004</td>
</tr>
<tr>
<td>InfoSphere Replication Server V9.5</td>
<td>2006</td>
</tr>
<tr>
<td>InfoSphere Replication Server V9.7</td>
<td>2010</td>
</tr>
<tr>
<td>IBM DB2 DataJoiner, Version 2.1</td>
<td>2010</td>
</tr>
<tr>
<td>IBM DB2 DataJoiner, Version 2.1 Adds SQL bases Heterogeneous Replication</td>
<td>2010</td>
</tr>
<tr>
<td>InfoSphere Data Replication V10.1</td>
<td>2012</td>
</tr>
<tr>
<td>InfoSphere Data Replication V10.3</td>
<td>2012</td>
</tr>
<tr>
<td>IBM DB2 DataJoiner, Version 2.1</td>
<td>2012</td>
</tr>
</tbody>
</table>

Distributing the IBM DB2 DataJoiner, Version 2.1
Adds SQL bases Heterogeneous Replication

© 2013 IBM Corporation
Packaging
## Replication Part Numbers

<table>
<thead>
<tr>
<th>Part number</th>
<th>Part description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0408LL</td>
<td>IBM InfoSphere Change Data Capture for Non Production Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0406LL</td>
<td>IBM InfoSphere Change Data Capture Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0402LL</td>
<td>IBM InfoSphere Change Data Capture for Oracle Replication Non Production Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0403LL</td>
<td>IBM InfoSphere Change Data Capture for Oracle Replication Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0NM3LL</td>
<td>IBM InfoSphere Change Data Delivery for Netezza Managed Server License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0B73LL</td>
<td>IBM InfoSphere Change Data Delivery for Non-Production Environments Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0B6ZLL</td>
<td>IBM InfoSphere Change Data Delivery Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0NNILL</td>
<td>IBM InfoSphere Data Replication for Database Migration Install Initial Fixed Term License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0L2NLL</td>
<td>IBM InfoSphere Data Replication for Non-Production Environments Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0L34LL</td>
<td>IBM InfoSphere Data Replication Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D0NMBLL</td>
<td>IBM InfoSphere Data Replication for Netezza Managed Server License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D61B0LL</td>
<td>IBM InfoSphere Replication Server Developer Edition Authorized User License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D59ILL</td>
<td>IBM InfoSphere Replication Server Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
<tr>
<td>D07DSLL</td>
<td>IBM Netezza Replication Services Processor Value Unit (PVU) License + SW Subscription &amp; Support 12 Months</td>
</tr>
</tbody>
</table>
Replication Bundling

- IBM InfoSphere Data Replication as an included product:
  - IBM Database Enterprise Developer Edition
  - IBM DB2 Advanced Enterprise Server Edition
SQL Replication
Relational Replication - Components

- Independent operation
- Refresh, update, and enhancement
- Database connectivity and security
Relational Replication - Capture

- Captures base table changes from log
- Runs locally to the source
- Maintains transaction consistency
- Automatically maintains staging tables
Relational Replication - Apply

- Runs from source or target platform
- Runs at user-specified intervals or events
- Refreshes, updates, and enhances copies
- Distribution optimizations

- Base and Copy Tables
- Interval and Repetition
- Column and Row Selection
- Computed Columns
- Aggregations
- Append or Replace
Relational Replication - Data Enhancement

Customize source data for specific target use

- Enhances data usability
- Supports unique application needs

Join  Aggregate  Derive
Relational Replication - Subset Distribution

```
SELECT * FROM POLICY
WHERE EXISTS (SELECT * FROM CUSTOMER
WHERE POLICY.CUST# = CUSTOMER.CUST# AND
CUSTOMER.BRANCH = 'DALLAS')
```
Relational Replication - Fan-Out Distribution

San Francisco

Where City = SF

Where State = CA

Los Angeles

State = CA

Branch = 7

Branch = 6

New York

Where State = NY

Data Source
Relational Replication - Administration

- GUI - part of DB2 Control Center or DSAC
- Registration/Subscription
- Relational control tables

- Source Table
- Capture
- Base Unit of Work
- Change Data Capture
- Administration

- Target Table
- Base and Copy Tables
- Interval and Repetition
- Column and Row Selection
- Computed Columns
- Aggregations
- Dynamic SQL
- Append or Replace
- PIT/USER History Staging Replica
Relational Replication - Highlight Functions

- Integrated replication administration
- Update Anywhere
- Mobile computing support
- Join View Support
- Set subscription
- Event based scheduling support
- Batch execution of Capture and Apply
- Logical partitioning key support
- Defer/run SQL support
- Subscription Cloning
- Stored procedure call
- Large answer set support
Relational Replication - Update Anywhere
Implementation

- Avoid update conflicts by design
  - Use application views over the replicas to enforce
    "distributed primary fragment" mutually exclusive update
    restrictions
  - Serialize the schedule for when each site can issue updates
  - Write insert-only applications
  - Handle rejected transactions

- ASNDONE exit
- Rejection codes
- Before/After row values
- RETENTION_LIMIT pruning
Mobile Computing Support

- On demand replication
- Initiated from mobile station

- Head office
- Branch office

- Mobile Sales Force

- Replication on demand
- Selective replication
- Dial exit
- Windows & Linux
Transaction & Non-Transaction Replication

- Supports both transaction and non-transaction replication
Relational Replication - Join View Support

- Replication through join views
  - Denormalize copies for data warehousing
  - Route data for distributed computing

Action - DEFINE VIEW
- One or more user tables
Relational Replication - Set Subscription

- Every subscription must belong to a set. A set may have one or more subscription members.
- Maintains referential constraints that exist among a set of tables at both source and target servers.
- Limits the boundary for cascade rejections due to RI violation or update collision.
- Keeps the subscriptions for all the components of a view subscription together.
Relational Replication - Event Based Scheduling Support

Subscriptions can be triggered by
- Relative time
- Event timer
- Both

Full Row Logging

SOURCE

TARGET

COPY

COPY

COPY

CONTROL

APPLY

UNIT OF WORK

CHANGE DATA

CONTROL TABLE

EVENT_NAME

EVENT_TIME

END_OF_PERIOD

Upper bound for change data

REPLICA
Logical Partitioning Key Support

- Allows a row to migrate from one partition to another

Diagram:

- Change AGENT # from 6 to 99
- Processes "Delete"

- AGENT # = 99
- Processes "Insert"
Relational Replication - Defer/Run SQL Support

- Allows customization of DDL before execution
- Saves definitions to plain ASCII file
Relational Replication - Subscription Cloning Support
Use MAX_SYNCH_MINUTES to break up the backlog into mini-subscriptions, using multiple real subscription cycles to perform one scheduled cycle. This applies to change data and not to full refresh.
IBM SQL Replication Features

- Versatile Infrastructure
  - Operational application
  - Informational application
  - Mobile computing
  - Update anywhere
- Leverages Current Assets
  - Legacy Sources
  - Standard SQL
- Efficient Operations
  - Log-based capture
  - Network optimizations
- Flexible Distribution
  - Subsetting with join views
  - Cascading distribution
- Scalable Design
  - Servers
  - Data
  - Networks
- Easy to Administer
  - GUI administration
  - Automated initialization
- Robust Data Enhancements
  - Derivation, Summarization, Translation...
- Multi-vendor Interoperability
  - SQL based architecture
  - Architected data staging area
Queue Replication (QREP)
Q Replication Components

- Replication Center – defines replication source-to-target mappings, manages Q Replication processes, provides monitoring reports, defines and manages Alert Monitor processes
- Q Capture – captures changes from the DB2 log and places them on a WebSphere MQ queue
- WebSphere MQ – transport for captured changes
- Q Apply – retrieves captured changes from a WebSphere MQ queue and processes them
- Alert Monitor – monitors Q Capture and Q Apply based on user-defined thresholds and events, sends e-mail notification when thresholds are exceeded or events occur
- Utilities – asntdiff/asntrep for reconciliation, ansqmfmt/exception formatter and Q Analyzer for diagnosis
- Data Studio Administration Console (DSAC) – the new interface to monitor health of Q Replication
- DSAC includes the Replication Dashboard – real time monitoring of Q Capture and Q Apply
Unidirectional Q Replication – DB2 to DB2

USES
Maintaining operational data stores, decision support systems, reporting systems, data warehouses

FEATURES
- Filtering by
  - row and column
  - change type – replication of deletes can be suppressed
  - authorization id, plan name
- SQL transformations

PERFORMANCE
12,500 – 15,000+ changed rows replicated per second in development lab environment

This assumes that the target is read only – updates will not be made to the target tables except by Q Apply. The performance numbers are conservative. Some customers have achieved much higher throughput with very low latency.
Unidirectional Q Replication – DB2 to non-DB2

<table>
<thead>
<tr>
<th>USES</th>
<th>FEATURES</th>
<th>PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintaining operational data stores, decision support systems,</td>
<td>Filtering by</td>
<td>Some customers have reported 6000 rows per second.</td>
</tr>
<tr>
<td>reporting systems, data warehouses</td>
<td>● row and column</td>
<td>Other customers report much less.</td>
</tr>
<tr>
<td></td>
<td>● change type – replication of deletes can be suppressed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● authorization id, plan name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL transformations</td>
<td></td>
</tr>
</tbody>
</table>

This is also a read only target. There is no support for bidirectional replication when a non-DB2 database is a target. Teradata is not yet supported.
### Uses

Consistent Change Data (CCD) tables are an audit trail of changes. They are used for auditing and feeding changes to other applications.

### Features

- Filtering by
  - row and column
  - change type – replication of deletes can be suppressed
  - authorization id, plan name
- SQL transformations

### Performance

12,500 – 15,000 changed rows replicated per second in development lab environment

CCD tables include the information about a change that occurred on the source table, including the type of change (insert, update, delete) and the approximate time the change was committed. A CCD can be complete (initially populated with all the rows currently in the source table) or non-complete (no initial synchronization, changes only as they occur). A CCD can be condensed (only the last change to a row) or non-condensed (one CCD row for every change to the source table). A CCD can optionally have additional columns like the authorization id that made the change and the transaction id of the source transaction.
This is an example of using Q Replication to move changes to a staging area CCD. The changes are then distributed to many targets using SQL Apply. The benefits are that the impact to the product system is minimized and the heavy lifting part of replication (applying to many targets) is moved away from production.
The stored procedure will be called once for every single change, so there may be significant overhead, depending on the logic inside the stored procedure. A better choice for this type of processing might be either a CCD or Event Publishing.
Event Publisher for DB2 – DB2 to XML

<table>
<thead>
<tr>
<th>USES</th>
<th>Feed applications (WebSphere Message Broker) or ETL programs (WebSphere DataStage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEATURES</td>
<td>Same filtering and transformations as DB2 -&gt; DB2</td>
</tr>
<tr>
<td></td>
<td>Different message content – 1 message per row changed or 1 message per transaction</td>
</tr>
<tr>
<td>PERFORMANCE</td>
<td>5000-10000 changed rows replicated per second in development lab environment, but this performance is only measured for the publishing of the XML message.</td>
</tr>
</tbody>
</table>

Replication sees a change as a DATA event – a row has changed and that change must be made to a corresponding row on the target. Event Publisher treats a change as a BUSINESS event. For example, a row is inserted into the DB2 CUSTOMER table. To the business, this is a NEW CUSTOMER and there are business activities that need to take place. A message broker can process the XML message and check the new customer’s credit report, send a welcome letter with a coupon, notify the outside sales rep to make a call and so on.
This is two-way replication. In this scenario, you identify a server that will always be the winner if the same row is updated on both servers. In the event of a conflict, the “losing” change is stored in an exception table. But, this happens asynchronously, after the losing change was already committed and the application had already moved on. There is no way to notify the application that the change has been backed out! The replication administrator can be notified if an exception occurs, but the review, resolution is manual. It is always better to have no conflicts if at all possible. Many customers direct all update applications to the primary site and use the standby site for reporting only to minimize conflicts and still get value from the standby site.
This new configuration works best if the Standby and DR sites are both read only, except when the primary is not available. Notice that there is no replication between the DR and the Standby site. For 3-way or greater replication, you must use peer-to-peer.
This is another variation of bidirectional, designed for customers that want to move all processing off the primary as quickly as possible. In this configuration, the Standby server is a single point of failure. If the standby is not available, then there is no replication to the Disaster Recovery server.
Peer-to-Peer Q Replication – DB2 to DB2 (2-n Sites)

<table>
<thead>
<tr>
<th>USES</th>
<th>Split query workload and provide high availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEATURES</td>
<td>No filtering</td>
</tr>
<tr>
<td></td>
<td>All peer databases are available for application updates</td>
</tr>
<tr>
<td></td>
<td>Key column conflict detection</td>
</tr>
<tr>
<td></td>
<td>Version based (timestamp) conflict resolution – timestamps maintained by triggers on source/target tables</td>
</tr>
<tr>
<td>PERFORMANCE</td>
<td>20% slower than unidirectional, with higher CPU due to trigger execution</td>
</tr>
</tbody>
</table>

The is multi-directional replication for 2-n peer nodes. The practical limit is 6, due to the difficulty of managing the queues involved. Two extra columns are placed on every replicated table, along with triggers to maintain those columns (all generated by the Replication Center). In the case of a conflict, the change with the latest timestamp is the winner. Conflicts are posted to an exception table to be reviewed by the Replication Administrator.
Q Replication Architecture
Q Replication High Level Architecture

- Capture program stages data either in queues
  - Each message represents a transaction
  - One queue per source/target database pair
- Apply is significantly re-architected
  - Parallel apply to single target table
- Conflict detection very robust, including ability to handle deleted and key changes
Q Replication Setup

• Install and configure WebSphere MQ 6.x or 7.x server on all source and target servers
• Install and configure InfoSphere Replication Server on all source and target servers
• Install DB2 Client 9.7 on replication administration workstation (also need DB2 Connect if sources and/or targets are DB2 Universal Database for OS/390® and z/OS®)
• Use the Replication Center to create Q Capture and Q Apply control tables
• Use the Replication Center to define Q Subscriptions and Queue Maps (send/receive queue pairs)
• Start Q Capture
• Start Q Apply
• Start Alert Monitor (optional)
• Start Q Replication Dashboard (optional)

Note that Q Capture must run on the source DB2 server and Q Apply must run on the target DB2 server. In most cases, this means at least 2 DB2 II installs (and licenses).

Also, you must configure DB2 connectivity between the source and target servers (DB2 Connect may be required) for the initial synchronization of the target table (full refresh) and DB2 connectivity from the Replication Administration workstation to the source and target servers.
• Q Capture side – local queue- adminq. One adminq per QCapture instance, remote definition- sendq, for QCapture to do puts – capture can put data on multiple sendq’s – each sendq must map to 1 receive queue – (QApply transactional ordering)

• also a local queue – restartq – one per Qcapture instance, to hold the position of the log (up to where the commit has happened, and the minimum inflight sequence)

• QApply – recvq is the local queue, 1instance of QApply can have multiple recvq’s, remote queue- adminq to send the control messages, spillq – dynamic, as the target table is getting loaded (either via internal/external) holds the transactions
These are MQ commands to create a queue manager (the –ll is for linear logging, and SYSA.XMITQ is the default staging area). A queue manager must be started, just like DB2. Each queue manager has a listener port which is used to receive remote requests. Other MQ objects are defined using the runmqsc command with SOURCE.mqdef as the input file. This input file is shown on the next slide.
These definitions support a Queue Replication source system where Q Capture will run. In fact, these are the MQ commands used to set up exercise 1 for this class. There is a similar set of definitions for the target side as well. All the MQ definitions used in this class are documented in the Appendices of the exercises.
Some customers prefer to install only the MQ client on their source and target servers to minimize the processing done on the production systems. The MQ client acts like a DB2 client, except that it connects to an MQ Queue Manager and issues writes (MQPUT) and reads (MQGET). The MQ server becomes a single point of failure, so you must take steps to ensure high availability for the MQ log files (HACMP is an example).
The DB2 High Availability/Disaster Recovery feature (HADR) supports a passive (no connections allow) standby server. The advantages of HADR are that it is very easy to set up (one window GUI) and offers very fast fail-over. The most significant drawback is that the standby database cannot be used except in a failover situation. In this picture the source server in one city has an HADR standby database for local failover. Q Replication is used to maintain a target in another city for remote failover and disaster recover. The remote target also has an HADR standby database for failover at that site. The weak link in this picture is the MQ Server. Which city should it be in? What if it is not available, but the source and target are both available? This configuration needs to be planned carefully to ensure that the MQ server is always available.
This is the BIG picture of the Queue Replication components:

Administration – define, operate, monitor Queue Replication

Q Capture – capture changes and send transactions to targets

Q Apply – receive transactions and apply to target tables or call stored procedures

Alert Monitor – monitor replication processes
The Replication Center is part of the DB2 Client. It is used to administer both SQL and Queue Replication. The client is installed when you install DB2 on Linux, UNIX, or Windows, so you can logon to the source or target DB2 server to run the Replication Center if you wish (requires X-Windows or Windows Terminal Services). You can also install just the client on a desktop server and administer replication remotely.
The Replication Center Launchpad includes the 3 replication options – Queue Replication, Event Publishing, and SQL Replication. Each option has its own launchpad as well.
The launchpad view is different for each type of replication. Each numbered step has a wizard that walks you through the action needed for that step.
Replication Center – Definitions

• The Replication Center must have DB2 (DRDA) connectivity to all source and target servers.
  • If the targets is non-DB2, then the Replication Center must have DRDA to the federated database that connects to the non-DB2 target.
• The Replication Center is used to DEFINE replication objects.
  • The Replication Administrator uses wizards to define replication.
    • Q Capture and Q Apply control tables
    • SENDQ/RECVQ pairs
    • Q Subscriptions and/or XML Publications
• The Replication Center generates SQL scripts based on the Administrators input. The scripts are run using the Replication Center and can also be saved for reference or reuse.
• Replication definitions can be changed using the Replication Center.

The Replication Center tasks for both SQL Replication Queue Replication are DEFINITIONS and OPERATIONS. A separate task is the Replication Alert Monitor which monitors both types of replication.
Replication Center – Operations

• The Replication Center can be used to OPERATE Q Capture and Q Apply.
• The Replication Administrator can perform the following operations
  • Start and Stop Q Capture and Q Apply
  • Check the status of Q Capture and Q Apply
  • Display reports (throughput, latency, messages)
• The Replication Center generates replication command scripts to stop and start Q Capture and Q Apply and to check status. The scripts are run using the Replication Center and can also be saved for reference or reuse.
• Reports are generated via SQL and displayed immediately.
• Q Apply and Q Capture runtime parameters can be changed using the Replication Center.

Actions which start/stop/check Q Capture and Q Apply programs require a Database Administration Server (DAS) running on the system where the programs execute. If the system is z/OS, then you will need to install/configure the DAS package to use those Replication Center actions. 99.9999% of z/OS customers prefer to manage their Q Capture and Q Apply started tasks natively on z/OS. 99.9999% of Linux/UNIX customers add the start-up of the programs to the system initialization tables and use the command line for all other tasks. 99.9999% of Windows customers create Windows services to automate the start-up of the replication programs.
Replication Center -- Monitoring

- The Replication Center can be used to MONITOR Q Capture and Q Apply.
- The Replication Administrator can perform the following operations
  - Create Monitor Control Tables
  - Set thresholds and choose events for alert monitoring
  - Identify contacts and groups of contacts for notification via an SMTP mail server
  - Start and stop the Alert Monitor
- The Replication Center generates SQL and replication command scripts to stop and start the Alert Monitor and to check status. The scripts are run using the Replication Center and can also be saved for reference or reuse.

The Alert Monitor is a separate program with it's own set of control tables. It can run on any server that can connect to the source and target servers.
Asnclp can be very useful for defining large numbers of subscriptions. We recommend that you start out with the Replication Center to get an understanding of the tasks and options available. You will use both the Replication Center and asnclp in the labs for this class.
Next, we'll look at Q Capture.
Source Table Requirements

- Unidirectional, Bidirectional, Peer-to-Peer, XML Publishing
  - If the source is DB2 for Linux, UNIX, or Windows, RECOVERY logging must be enabled.
  - If the source table does not have the DATA CAPTURE CHANGES attribute, that attribute will be set during replication definition
- Bidirectional, Peer-to-Peer
  - The source table must have a set of columns that uniquely identify a row so that the Apply program or XML generator can locate the row to be inserted, updated, or deleted
- Replication is not supported for
  - Data Links columns
  - Spatial columns
  - DB2 z/OS columns with FIELDPROCs, or VALIDPROCs (EDITPROCs are supported)
  - DB2 Linux, UNIX, Windows XML data type support is planned for a future fixpack.
  - Source table LOADs done by the DB2 LOAD utility or command are NOT replicated.
  - Source data stored in compressed tablespaces on z/OS must be REORGed with KEEPDICTIIONARY YES (DB2 V8 APAR PK19539 removes this restriction)

Circular logging cannot be used with replication, since DB2 may write over a log file before Capture has retrieved all the changes. DATA CAPTURE CHANGES forces full row logging, instead of the default where UPDATES are only logged from the first changed column to the end of the row. This may require an increase in the size of log files, depending on system activity.

There is almost always a question about why LOAD is not replicated if you specify LOG YES. The answer is that LOAD does not issue inserts or updates (on z/OS or distributed) and Q Capture only sees inserts, updates, deletes, commits, and rollbacks. When you specify LOG YES, DB2 logs the fact that a load occurred (and may log the data pages it writes), but it does not log inserts/updates that can be captured.
Source tables are updated and the updates are logged by DB2, in this case transactions TX1, TX2, and TX3. Capture requests log records from DB2 and saves those log records in memory until the transaction (unit of work) is committed or rolled back.

When the COMMIT is received for TX1, Capture places the TX1 transaction on the SENDQ and removes it from memory. When the ROLLBACK is received for TX3, Capture removes the TX3 transaction from memory. Transaction TX2 stays in memory until it is committed or rolled back. If Capture is stopped, it will request the log records for TX2 and any other in-flight transactions from DB2 and place it in memory.
Filtering and subsetting

- Subset data
  - Subset of rows through Q Capture predicate on subscription/publication
  - Subset of columns through subscription/publication definition
  - Option included for ignoring deletes
  - Filter transactions by userid, plan name using entries in a control table
  - Signal defined to allow user selected transactions to be ignored
- Subsetting is done by
  - Q Apply during the initialization of the target table
  - Q Capture when capturing changes

The signal to skip certain transactions is not defined in the Replication Center. At the start of a transaction that you don’t want to replicate, you insert a signal in the Q Capture signal table. Q Capture will skip all the SQL statements in that transaction.
Q Capture Miscellaneous

- One Q Capture process (schema) can process multiple SENDQs.
- For Peer to Peer replication, the Q Capture and Q Apply schemas must be the same on all Peer nodes.
- Q Capture and Q Apply control tables must be on the catalog node in a DB2 for Linux, UNIX and Windows partitioned database.
- Q Capture can handle LOBs that are larger than the maximum message size allowed by WebSphere MQ. Q Capture splits the LOBs into multiple messages based on the maximum message size defined for the SENDQ. Q Apply combines the LOB messages before applying the change.
- Q Capture can handle large transactions by sending multiple messages.
- Columns added to source tables can be added to replication (for DB2 targets) via the Replication Center
  - Alter add to the source table
  - Insert ADDCOL signal to the Q Capture IBMQREP_SIGNAL table
  - New column automatically added to replication control tables and the target table

Lobs are supported for all types of replication, including bidirectional and peer-to-peer
Now, let’s look at Q Apply
Target Table Requirements

• Bidirectional and Peer-to-Peer
  • Each target table MUST have a set of columns which uniquely identify each row in the target table.
  • Uniqueness can be defined as a primary key or unique index on the target table
• Unidirectional only to non-DB2 targets (Oracle, Sybase, etc.), CCD targets, and stored procedures
• All configurations
  • If target tables are related through referential constraints or through application logic, then they must be processed by a single SENDQ/RECVQ pair so that all transactions are processed in the proper order.
  • LOBs cannot be replicated to non-DB2 targets except Oracle. This is a Federation Server restriction – LOB updates are not supported for any data source but Oracle.

Uniqueness matters!
Q Apply: Initial Load Processing (Full Refresh)

- Handles RI automatically
  - i.e. drops and saves RI, loads, puts RI back on
- Q Capture begins capturing changes before Q Apply starts the unload of the source table, to ensure that no changes are lost.
  - Changes that are captured during the load are placed on a temporary spill queue and processed after the load is complete.

The slide that follows this describes the Initial Load processing.

Q Capture receives CAPSTART signal
Q Capture sends schema msg
Q Capture immediately starts capturing and sending transactions
Q Apply starts a loading agent
Q Apply creates a spill queue
Q Apply spills transactions to spill queue (during load)
  * all other subscriptions continue normally
Q Apply initiates load of target table (assuming internal load)
Q Apply sends LOADDONE msg
Q Apply applies transactions from spill queue
Q Apply integrates into “normal” agent pool process
Target Table Initialization

• Q Capture detects a new Q subscription and inserts a CAPSTART signal in the Q Capture control table capschema.IBMQREP_SIGNAL. This log sequence number of this insert is the point in the log where capturing will start for the source table.

• Q Capture sends a SCHEMA msg to Q Apply on the SENDQ and begins capturing transactions and places them on the SENDQ for Q apply

• Q Apply creates a SPILLQ from the model definition and saves the transactions in the RECVQ to the SPILLQ.

• Q Apply does the initial refresh based on the SCHEMA msg from Q Capture and the LOAD options in the IBMQREP_TARGETS table row for the Q Subscription

• Q Apply applies the transactions from the SPILLQ and then deletes the SPILLQ

• Q Apply applies the transactions from the SENDQ

This describes the actions picture on the slide right before this one.
Q Apply Load Options

- A subscription can be defined with:
  - automatic load, manual load, no load required
- Automatic load:
  - Load is performed by Apply, with automatic coordination of the simultaneous capture of changes, loading of the new table, and apply of changes to other tables.
- Manual load:
  - Load is performed by user, coordination is required, and will be handled by user (with some help from our administration).
- No load:
  - No loading required, no coordination required, can immediately capture and apply changes
  - Example: target system is built through backup/restore, with replication started from an inactive source

When source tables are being updated in parallel with the extraction of the source data to populate the target table (initially, before replication begins), then coordination is required between the Q Capture and Q Apply processes and the load itself. This coordination can be performed automatically by the product, or by the user if that is preferred.

When the source tables can be made temporarily inactive, still other methods can be employed that require no coordination. In this case the subscriptions can be defined as “no load required.”
Q Apply is a multi-threaded process. The Apply program starts a Q Apply browser thread for each RECVQ listed in the control tables for that Apply program. The Q Apply Browser thread launches 1 or more Apply agents to process transactions received on the RECVQ. Whenever possible, Apply agents work in parallel to maximize the throughput.
How Does Apply work for unidirectional?

- A Q Apply program can process multiple RECVQs.
- A Q Apply browser thread is started for each RECVQ defined in the IBMQREP_TARGETS table. This thread coordinates the application of changes for the associated target tables.
- A Q Apply browser thread starts 1 or more Q Apply agents (configurable by the Replication Administrator).
- Transactions are passed from the Q Apply browser to the Q Apply agents and are processed in parallel if possible.
  - Transactions which affect the same rows in the same table are always processed in order by a single Apply agent.
  - Transactions which affect tables that are related by RI constraints are always processed in order by a single Apply agent.
- Each message has a unique, sequential message id. Q Apply saves the last message id processed for restart.

A single transaction is never split between Apply agents. Apply gets target table RI information from the DB2 system catalog. If the target tables have a relationship that is not defined in DB2, then it is possible for the data to be inconsistent from an application point of view. The data will eventually converge to consistency as the transactions are applied. For example, each employee row in the EMP table has a DEPT value. The application requires that each DEPT value in EMP must have a matching value in the DEPARTMENT table, but no constraints are defined in DB2.

If TX1 inserts DEPT A01 in the DEPARTMENT table and TX2 inserts an employee with department A01 in the EMP table, it is possible that the insert to the EMP table will occur first. So, the EMP table will be inconsistent to the application (not to DB2) until TX1 is processed.
### Q Apply Transformations SQL expressions

Fixpack 1: Transformations for non-key columns
Fixpack 2: Transformations for key columns and non-key columns

#### Target Table Expressions

<table>
<thead>
<tr>
<th>Target Column</th>
<th>Target Column Expression</th>
<th>Mapping Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY1</td>
<td>KEY1</td>
<td>(1-1 mapping)</td>
</tr>
<tr>
<td>C12</td>
<td>[:C1][:C2]</td>
<td>(N-1 mapping)</td>
</tr>
<tr>
<td>C2A</td>
<td>substring(C2,2,3)</td>
<td>(1-N mapping)</td>
</tr>
<tr>
<td>C2B</td>
<td>substring(C2,5,5)</td>
<td>(1-N mapping)</td>
</tr>
<tr>
<td>C2C</td>
<td>int(substring(2,1,1))</td>
<td>(1-N mapping)</td>
</tr>
<tr>
<td>C34</td>
<td>[:C3 + :C4]</td>
<td>(N-1 mapping)</td>
</tr>
<tr>
<td>C5</td>
<td>CURRENT_TIMESTAMP</td>
<td>Generated column</td>
</tr>
<tr>
<td>C6</td>
<td>IBM</td>
<td>Generated column</td>
</tr>
<tr>
<td>C7</td>
<td>substring('1',1,1)</td>
<td>Generated column</td>
</tr>
</tbody>
</table>

- **Generated columns C12,C2A,C2B,C34 are based on source table Column values.**
- **Generated columns C5, C6, C7 are literals.**

Expressions are stored in Q Apply control table IBMQREP_TRG_COLS

This new function was added in Fixpacks 1 and 2 for Replication Server 9.1. They can be particularly useful when replicating to an existing target table with a structure that does not match the source.
How Does Apply work for bidirectional?

- Bidirectional replication is supported for 2 or 3 servers.
- Each server has a Q Capture and Q Apply program.
- There are 2 sets of MQ definitions on each server.
  - Queue definitions for Q Capture
  - Queue definitions for Q Apply
- The Q Subscription type is Bidirectional and the replication definitions are automatically set up through the Replication Center.
- Q Capture behaves the same regardless of Q Subscription type
  - Changes are captured in the same manner
  - More data may be sent depending on the subscription type and options
- Q Apply behaves the same as unidirectional EXCEPT for conflict handling.

The next slide discusses conflict handling for bidirectional replication.
Conflict Handling for Bidirectional Replication

• **VALUE based conflict detection:**
  • Conflict level options offered:
    ✓ Check all columns on update - requires transmission of all old/new values
    ✓ Check only changed columns on update - allows for column merge
    ✓ Check only key columns

• **Resolution choices offered: Force or Ignore set at each server**
  • Force Action - requires transmission of all new values
    ▪ force convergence on conflicts – apply the change (this is the loser!)
    ▪ log the conflict
  • Ignore Action
    ▪ log the conflict – do not apply the change (this is the winner!)

Options provided to meet the wide spectrum of needs:

Version based

**VALUE Based options: provides “very good convergence”**

- check all columns on update
  -> requires transmission of all old/new values

- update target table where key values = < > and all current target values = all old source values
  - check only changed columns on update - allows for merge

- update target table where key values = < > and current target values = old source values for columns changed at source
  - check only changed key columns

- Recursion avoidance is handled through signal log records from Apply to Capture
Value Based Conflict Detection

- Do the current row values at the apply target match the old row (before values) carried over from the source update?
- Designated site wins.

<table>
<thead>
<tr>
<th>Server A (winner)</th>
<th>Server B (loser)</th>
</tr>
</thead>
<tbody>
<tr>
<td>row ((a,b,c)) changed to ((a,b,x))</td>
<td>row ((a,b,c)) changed to ((a,z,c))</td>
</tr>
<tr>
<td>Before values ((abc)) and after values ((abx)) are sent to Server B</td>
<td>Before values ((abc)) and after values ((azc)) are sent to Server A</td>
</tr>
</tbody>
</table>

1. Does old row \((a,b,c)\) match existing row \((a,b,x)\) ??
2. No = conflict
3. Who wins? = Server A
4. Ignore the change, Log the conflict
5. Row at Server A is \((a,b,x)\)

1. Does old row \((a,b,c)\) match existing row \((a,z,c)\) ??
2. No = conflict
3. Who wins? = Server A
4. Force the change, Log the conflict
5. Row at Server B is \((a,b,x)\)

Value based conflict detection and resolution (CDR) is performed by shipping the old column values along with new column values, and comparing the old values from the source against the current values at the target. If the current values at the target do not match, this means that both copies of the data have been changed in this timeframe. By comparing to current values, this allows real time comparisons to be made (unlike older SQL replication methods that compared captured data to captured data).

This method is impractical beyond 2 servers. There is no arbitrary comparison point by which to provide a “winning” value, that would allow the databases to converge, so the simple approach of selecting a designated winning site is implemented.

By designating a secondary (failover/standby) system to be the “winner”, and by implementing a careful switchback procedure, this method can be used in a primary /secondary failover system to good effect.
How Does Apply work for peer-to-peer?

- Peer-to-peer replication is supported for any number of servers. The practical limit is 6, based on development tests.
- Each server has a Q Capture and Q Apply program.
- There are multiple sets of MQ definitions on each server.
  - Multiple Queue definitions for Q Capture (one SENDQ for each peer)
  - Multiple queue definitions for Q Apply (one RECVQ for each peer)
- The Q Subscription type is Peer to Peer and the replication definitions are automatically setup through the Replication Center. Additional peer servers can be added without stopping the existing peers.
- Q Capture behaves the same regardless of Q Subscription type.
  - Changes are captured in the same manner
  - More data may be sent depending on the subscription type and options
- Q Apply behaves the same as unidirectional EXCEPT for conflict handling.

Conflict handling for peer-to-peer is totally different from conflict handling for bidirectional. Bidirectional supports value-based conflict detection, with 1 designated server as the winner in the event of a conflict. Peer-to-Peer uses version-based conflict detection (timestamps) with the winner based on time.
Conflict Handling for Peer-to-Peer Replication

• Conflict detection based on timestamp

• VERSION based conflict resolution:
  • Based upon time zone adjusted timestamps, most recent timestamp “wins”
  • Each source/target table must have two extra columns to support version-based conflict handling (timestamp, tie-breaker)
    • Extra columns maintained by triggers (insert/update)
    • Replication Center adds the required columns and creates the necessary triggers to maintain those columns
    • Time zones can vary, but the machine clocks should be well synchronized
  • Triggers have impact on applications that access source/target tables

Options provided to meet the wide spectrum of needs:

Version based

VALUE Based options: provides “very good convergence”

- check all columns on update
  - requires transmission of all old/new values

  • update target table where key values = < > and all current target values = all old source values
    - check only changed columns on update - allows for merge

  • update target table where key values = < > and current target values = old source values for columns changed at source

    - check only changed key columns

- Recursion avoidance is handled through signal log records from Apply to Capture
Version Based Conflict Resolution

- All rows are augmented with a “Version” = timestamp Tx and smallint Nx, indicating when and by which server the row was last updated.
- Do the current values of Tx and Nx at the apply target match the old values of Tx and Nx carried over from the source update?
- Most current timestamp Tx wins.

Server A (N1)

row \((a,b,c,T1,N1)\) changed to \((a,b,x,T3,N1)\)
Before values \(T1,N1\) and after values \(a,b,x,T3,N1\) are sent to Server B

1. Does old version \((T1,N1)\) match existing version \((T3,N1)\) ??
2. No = conflict
3. \(T3 > T2\), T3 version wins
4. Row at Server A is \((a,b,x,T3,N1)\)

Server B (N2)

row \((a,b,c,T1,N1)\) changed to \((a,z,c,T2,N2)\)
Before values \(T1,N1\) and after values \(a,z,c,T2,N2\) are sent to Server A

1. Does old version \((T1,N1)\) match existing version \((T2,N2)\) ??
2. No = conflict
3. \(T3 > T2\), T3 version wins
4. Row at Server B is \((a,b,x,T3,N1)\)

The basic function of version based CDR is that the rows are augmented with values that give them a more global nature. This allows a better picture of what has happened to the data in the most recent time interval prior to new data being applied from another database. It is not complete and sufficient, and other methods are implemented in addition to the versioning columns. A “tombstone” table helps CDR remember certain deletes that have occurred. Special methods are used in handling some conflicts scenarios, in order to handle insert vs. insert+delete conflicts without having to remember all deletes.

Because of the complexities of relative arrival rate problems that occur in multidirectional replication beyond 2 servers, this peer to peer method is the only option we allow for greater than 2 servers.
What happens to the conflict that loses?

- Changes that are not applied because of a conflict are logged in the IBMQREP_EXCEPTIONS control table at the target. The rejected change is stored in XML format in the control table.

- This behavior is the same for bidirectional and peer-to-peer replication.

- The Exceptions Table Formatter Utility can be used to display the exceptions in readable format.
How Does Apply work for stored procedure targets?

• Q Apply agents receive transactions the same way that they do for all other replication scenarios.
• The Q Apply agent calls the user-supplied stored procedure for each SQL statement in the transaction.
• The input to the user-supplied stored procedure is the type of operation (insert, update, delete) and the values from the changed row at the source.
• The stored procedure must not issue a COMMIT or ROLLBACK.
• The stored procedure must return an SQLCODE that indicates success or failure.
• Q Apply has no control over the stored procedure processing.

This technique can be used to do data transformations or other processing before storing the data in DB2. But, the stored procedure itself must store the data. Q Apply passes the SQL data to the stored procedure and waits for a return code – it does not take any other action with changes from the queue.
Q Apply Miscellaneous

- One Q Apply process (schema) can process multiple RECVQs.
- For Peer to Peer replication, the Q Capture and Q Apply schemas must be the same on all Peer nodes.
- Q Capture and Q Apply control tables must be on the catalog node in a DB2 for Linux, UNIX and Windows partitioned database.
- There must be a set of columns that uniquely identify a row in the target table for bidirectional and peer-to-peer processing. This can be defined as a primary key or a unique index. The uniqueness must also be enforced at the source.
- Q Apply programs running on Linux, UNIX, or Windows need a password file to provide connect information. You create this password file on the system where Q Apply will run with the asnpwd command.
What happens if Q Apply encounters an error?

This screen is part of the Q Subscription definition. Each Q Subscription (1 source to 1 target table mapping) can have a different option, if desired. The next 2 slides discusses the options.
What happens if Apply encounters a data error?

If Apply cannot process an insert (row is already in target table) or update/delete (row is not in target table), the action taken depends on the values set when the Q Subscription was created:

- Force the change to the target table (this is how SQL Replication handles the problem – rework)
- Skip this change and continue
  All skipped changes are logged in the IBMQREP_EXCEPTIONS table
- Stop the Q Subscription – only the failing table is affected
- Stop the RECVQ – all tables processed by the queue are affected
- Stop the Apply program – all tables processed by this Apply are affected

If Stop the Q Subscription is chosen, then the subscription for this target table will be deactivated, but replication for all other subscriptions with the same RECVQ will continue to be processed.

If Stop the RECVQ is chosen, then if this subscription has a problem, replication for all subscriptions with the same RECVQ is stopped.

If Stop the Apply program is chosen, then all subscriptions processed by this Apply program are stopped.
What happens if Apply encounters a database error?

- If Apply cannot process a change because of some other condition (tablespace full, transaction log full, database not available), the action chosen when the Q Subscription was created is taken:
  - Stop the Q Subscription – affects only the failing table
  - Stop the RECVQ – affects all tables processed by this queue
  - Stop the Apply program – affects all tables processed by this Apply
- For unidirectional Q Subscriptions only, you can specify SQL states that you are willing to accept and Q Apply will skip any change that returns one of those states.

The impact here is the same as described on the previous slide. Stop the Q Subscription stops replication for one target table. Stop the RECVQ stops replication for all target tables processed by that RECVQ. Stop Apply stops all replication.
Operating Q Capture and Q Apply

1. Q Replication processes can be started/stopped by:
   - Replication Center (requires a Database Administration Server – DAS – running at the Q Capture and Q Apply server)
   - Line commands on Linux, UNIX, Windows – asnqcap, asnqapp
     - Windows services
   - Started tasks or batch jobs on z/OS

2. Status of Q Replication processes can be displayed by:
   - Replication Center “Check Status”
   - Line commands on Linux, UNIX, Windows – asnqccmd, asnqacmd
     - Modify command on z/OS

Generally, customers use the Replication Center for start/stop/status during testing, but use command line or automation for these tasks in production.
**Q Capture Status Command with details**

```bash
asncqcmd capture_server=qtest status show details
```

<table>
<thead>
<tr>
<th>Q Capture program status</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>(SERVER) = QTEST</td>
</tr>
<tr>
<td>Scheme name</td>
<td>(SCHEMA) = ASN</td>
</tr>
<tr>
<td>Program status</td>
<td>(STATUS) = Up</td>
</tr>
<tr>
<td>Time since program started</td>
<td>(UP_TIME) = 0d 0h 1m 57s</td>
</tr>
<tr>
<td>Log file location</td>
<td>(LOGFILE) = /vbs/dpropr/mpub/qtest.ASN.QCAP.log</td>
</tr>
<tr>
<td>Number of active Q subscriptions</td>
<td>(ACTIVE_Q_SUBS) = 3</td>
</tr>
<tr>
<td>Last committed transaction published (LSN)</td>
<td>(ALL_PUBLISHED_AS_OF_LSN) = 0000:0000:0000:6978:20E0</td>
</tr>
<tr>
<td>Current application memory</td>
<td>(CURRENT_MEMORY) = 0 MB</td>
</tr>
<tr>
<td>Path to DB2 log files</td>
<td>(DB2LOG_PATH) = /home2/qtest/qtest/NODE0000/SQL00002/SQLQIDIR/</td>
</tr>
<tr>
<td>Oldest DB2 log file needed for Q Capture restart</td>
<td>(OLDEST_DB2LOG) = S0000214.LOG</td>
</tr>
<tr>
<td>Current DB2 log file captured</td>
<td>(CURRENT_DB2LOG) = S0000214.LOG</td>
</tr>
</tbody>
</table>

This shows the output from the new show details parameter. Highlighted here are the items that show how current the Capture program is with the DB2 recovery log, which indicates the latency of the captured data, and the oldest DB2 log file that needs to be retained for replication purposes.
This shows the output from the new show details parameter. A few things you can see highlighted here are the current queue depth, average end-to-end latency, and the number active subscriptions for this Q Apply.
Change Data Capture (CDC)
High level architecture

Sources
- Oracle
- SQL Server
- DB2
- Informix
- Sybase

Journal Log
Redo/Archive Logs

Java-based GUI
for admin and monitoring

Target Engine
and Metadata

TCP/IP

Targets
- Database
  (Oracle, DB2, SQL Server, Teradata, etc.)
- ETL (DataStage, others)
- JMS (MQ, others)
- Information Server
  (DataStage, QualityStage, etc.)
- Flat files
- Web Services

ETL (DataStage, others)

Database

Sources

Targets
### Expansive source, target, platform support

<table>
<thead>
<tr>
<th>DATABASES Source and Target</th>
<th>TARGETS</th>
<th>NETWORK PROTOCOL</th>
<th>MESSAGE QUEUE</th>
<th>OPERATING SYSTEMS</th>
<th>HARDWARE PLATFORMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 z/OS®</td>
<td>Teradata</td>
<td>TCP/IP</td>
<td>JMS</td>
<td>IBM i OS</td>
<td>IBM i OS</td>
</tr>
<tr>
<td>Oracle</td>
<td>Information Server</td>
<td></td>
<td>MQ Series</td>
<td>z/OS</td>
<td>IBM System z®</td>
</tr>
<tr>
<td>Sybase</td>
<td>Cognos Now!</td>
<td></td>
<td>TIBCO</td>
<td>AIX®</td>
<td>IBM System p®</td>
</tr>
<tr>
<td>MS SQL Server</td>
<td>Netezza®</td>
<td></td>
<td>WebMethods</td>
<td>HP-UX</td>
<td>HP PA-RISC</td>
</tr>
<tr>
<td>DB2 UDB</td>
<td>Greenplum®</td>
<td></td>
<td>BEA</td>
<td>Solaris</td>
<td>Intel</td>
</tr>
<tr>
<td>DB2 i</td>
<td>mySQL®</td>
<td></td>
<td></td>
<td>MS Windows</td>
<td>Sun</td>
</tr>
<tr>
<td>Informix®</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solid DB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Customized solution, limited requirements

Discovering the value of IBM InfoSphere CI

© 2013 IBM Corporation
Flexible implementation

- Uni-directional
- Cascade
- Bi-directional
- Local
- Two-way
- Multi-thread
- Remote capture
- Consolidation
- Distribution

Discovering the value of IBM InfoSphere CI

© 2013 IBM Corporation
Easy to use

- Java-based GUI for configuration, administration and monitoring
  - Manage data integration processes from one screen
  - Automatic mapping, drag-and-drop transformations
  - No programming required
  - Event logs, alerts and alarms and statistics reporting

Monitoring capabilities through an intuitive GUI along with alarms and alerts makes DataMirror technology extremely easy to use. Manual configuration is reduced to a minimum with features such as automated table and column mapping. Programming, scripting, or database knowledge are not required in setting up or managing data replication processes.
What you’re looking at right now is the architecture of a technology called log-based change data capture, or log-based CDC. This is a very good method for extracting data from production systems because your applications are kept at peak performance to focus on running the business.

Here’s how it works. The database on the left side of your screen is your production system. Whenever data is inserted, updated, or deleted in your production databases, those transactions are automatically written to a database log. This is what your production system does by default. Log-based change data capture gets data out of your production system not by querying the database, but by going through these database logs. This way, extracting data doesn’t require the database to do any additional work, so the application can dedicate all of its resources in making sure that operations are running smoothly and that customers are being served quickly.

But low impact is only the log-based part. Another reason for why log-based CDC is a very efficient solution is because it uses change data capture, which means when data is updated in the production database, only the data that has changed are sent to its destination. As a result, rather than sending entire rows of data, only a fraction of the data is sent to the data warehouse, which allows for high throughput because the amount of work it takes to move information from one place to another is reduced to a minimum. This makes log-based CDC a very scalable method for integrating data across the enterprise.

So now that you are able to get data out of your production systems and send it to other locations efficiently, the rest is pretty simple. If you look on the right side of the slide you can see that you can pretty much distribute data to whatever destination you need. You may want to send that data to another database that other systems are running on, or you can integrate the fresh data with your web store, or post it onto a message queue that delivers it to your service-oriented architecture to automate downstream processes.

For example, we just implemented log-based CDC for a bank down in the Latin Americas. They wanted to distribute their Internet banking data from their web server to a backup server, because as you would expect from a bank, if their web server crashes or goes down, they can’t afford to lose their transaction records. So they need to maintain a second copy of their data at all times. But if you are constantly hitting your database to copy data over to the backup machine, then your customers wouldn’t be able to do their banking. But with log-based CDC, distributing data to the backup server doesn’t require the database to do any work, so customers are not fighting for server resources to do their banking.

Again, we go back to providing customers with better and faster service in order to meet their very high expectations. When you are able to satisfy their needs, they are more likely to stay as customers rather than take their business somewhere else.
What you’re looking at right now is the architecture of a technology called log-based change data capture, or log-based CDC. This is a very good method for extracting data from production systems because your applications are kept at peak performance to focus on running the business.

Here’s how it works. The database on the left side of your screen is your production system. Whenever data is inserted, updated, or deleted in your production databases, those transactions are automatically written to a database log. This is what your production system does by default. Log-based change data capture gets data out of your production system not by querying the database, but by going through these database logs. This way, extracting data doesn’t require the database to do any additional work, so the application can dedicate all of its resources in making sure that operations are running smoothly and that customers are being served quickly.

But low impact is only the log-based part. Another reason for why log-based CDC is a very efficient solution is because it uses change data capture, which means when data is updated in the production database, only the data that has changed are sent to its destination. As a result, rather than sending entire rows of data, only a fraction of the data is sent to the data warehouse, which allows for high throughput because the amount of work it takes to move information from one place to another is reduced to a minimum. This makes log-based CDC a very scalable method for integrating data across the enterprise.

So now that you are able to get data out of your production systems and send it to other locations efficiently, the rest is pretty simple. If you look on the right side of the slide you can see that you can pretty much distribute data to whatever destination you need. You may want to send that data to another database that other systems are running on, or you can integrate the fresh data with your web store, or post it onto a message queue that delivers it to your service-oriented architecture to automate downstream processes.

For example, we just implemented log-based CDC for a bank down in the Latin Americas. They wanted to distribute their Internet banking data from their web server to a backup server, because as you would expect from a bank, if their web server crashes or goes down, they can’t afford to lose their transaction records. So they need to maintain a second copy of their data at all times. But if you are constantly hitting your database to copy data over to the backup machine, then your customers wouldn’t be able to do their banking. But with log-based CDC, distributing data to the backup server doesn’t require the database to do any work, so customers are not fighting for server resources to do their banking.

Again, we go back to providing customers with better and faster service in order to meet their very high expectations. When you are able to satisfy their needs, they are more likely to stay as customers rather than take their business somewhere else.
What you're looking at right now is the architecture of a technology called log-based change data capture, or log-based CDC. This is a very good method for extracting data from production systems because your applications are kept at peak performance to focus on running the business. Here's how it works. The database on the left side of your screen is your production system. Whenever data is inserted, updated, or deleted in your production databases, those transactions are automatically written to a database log. This is what your production system does by default. Log-based change data capture gets data out of your production system not by querying the database, but by going through these database logs. This way, extracting data doesn't require the database to do any additional work, so the application can dedicate all of its resources in making sure that operations are running smoothly and that customers are being served quickly.

But low impact is only the log-based part. Another reason for why log-based CDC is a very efficient solution is because it uses change data capture, which means when data is updated in the production database, only the data that has changed are sent to its destination. As a result, rather than sending entire rows of data, only a fraction of the data is sent to the data warehouse, which allows for high throughput because the amount of work it takes to move information from one place to another is reduced to a minimum. This makes log-based CDC a very scalable method for integrating data across the enterprise.

So now that you are able to get data out of your production systems and send it to other locations efficiently, the rest is pretty simple. If you look on the right side of the slide you can see that you can pretty much distribute data to whatever destination you need. You may want to send that data to another database that other systems are running on, or you can integrate the fresh data with your web store, or post it onto a message queue that delivers it to your service-oriented architecture to automate downstream processes.

For example, we just implemented log-based CDC for a bank down in the Latin Americas. They wanted to distribute their Internet banking data from their web server to a backup server, because as you would expect from a bank, if their web server crashes or goes down, they can’t afford to lose their transaction records. So they need to maintain a second copy of their data at all times. But if you are constantly hitting your database to copy data over to the backup machine, then your customers wouldn’t be able to do their banking. But with log-based CDC, distributing data to the backup server doesn’t require the database to do any work, so customers are not fighting for server resources to do their banking.

Again, we go back to providing customers with better and faster service in order to meet their very high expectations. When you are able to satisfy their needs, they are more likely to stay as customers rather than take their business somewhere else.
What you’re looking at right now is the architecture of a technology called log-based change data capture, or log-based CDC. This is a very good method for extracting data from production systems because your applications are kept at peak performance to focus on running the business.

Here’s how it works. The database on the left side of your screen is your production system. Whenever data is inserted, updated, or deleted in your production databases, those transactions are automatically written to a database log. This is what your production system does by default. Log-based change data capture gets data out of your production system not by querying the database, but by going through these database logs. This way, extracting data doesn’t require the database to do any additional work, so the application can dedicate all of its resources in making sure that operations are running smoothly and that customers are being served quickly.

But low impact is only the log-based part. Another reason for why log-based CDC is a very efficient solution is because it uses change data capture, which means when data is updated in the production database, only the data that has changed are sent to its destination. As a result, rather than sending entire rows of data, only a fraction of the data is sent to the data warehouse, which allows for high throughput because the amount of work it takes to move information from one place to another is reduced to a minimum. This makes log-based CDC a very scalable method for integrating data across the enterprise.

So now that you are able to get data out of your production systems and send it to other locations efficiently, the rest is pretty simple. If you look on the right side of the slide you can see that you can pretty much distribute data to whatever destination you need. You may want to send that data to another database that other systems are running on, or you can integrate the fresh data with your web store, or post it onto a message queue that delivers it to your service-oriented architecture to automate downstream processes.

For example, we just implemented log-based CDC for a bank down in the Latin Americas. They wanted to distribute their Internet banking data from their web server to a backup server, because as you would expect from a bank, if their web server crashes or goes down, they can’t afford to lose their transaction records. So they need to maintain a second copy of their data at all times. But if you are constantly hitting your database to copy data over to the backup machine, then your customers wouldn’t be able to do their banking. But with log-based CDC, distributing data to the backup server doesn’t require the database to do any work, so customers are not fighting for server resources to do their banking.

Again, we go back to providing customers with better and faster service in order to meet their very high expectations. When you are able to satisfy their needs, they are more likely to stay as customers rather than take their business somewhere else.
Modes of replication

- Continuous mirroring
  - Changes read from database log.
  - Apply change at the target as soon as it is generated at the source.
  - Replication job remains active waiting for next available log entry.

- Periodic mirroring
  - Changes read from database log.
  - Apply net changes on a scheduled basis.
  - Replication job ends when available log entries are processed.

- Refresh
  - File/table level operation.
  - Apply a snapshot version of source table.
  - Typically used to achieve initial synchronization of source and target table.

Three types of replication modes are available depending on the needs of the business.

Continuous mirroring – uses log-based CDC to replicate data transactions in real time as they occur
Periodic mirroring – replicates data only during scheduled intervals, does not provide transactional history
Refresh – one-time snapshot replication from source to target
Subset refresh and differential refresh

- Common uses for subset refresh functionality
  - Refreshing very large tables in stages
    - Accommodating smaller batch windows
    - Less interruption for other tables being replicated
    - Example:
      - Refreshing a table of one billion rows can be spread over multiple days
      - Every day 200 million rows can be refreshed
  - Refreshing a day's worth of changes, if there is a column that contains change date
  - Synchronization check for subset of rows
    - Using differential refresh functionality

- Differential refresh
  - Allows for refreshing/checking rows with discrepancies
  - This function may also be used to perform a synchronization check
Table mapping methods

- **One-to-one**
  - Source and target tables have similar table structures
- **LiveAudit™**
  - Generates audit trail of data transactions from source
- **Adaptive Apply**
  - Automatically synchronizes data for dissimilar sources and targets
- **Summarization**
  - Keeps a running total of numerical values at the target
- **Consolidation: One-to-One**
  - Merges data from several tables into a single row
- **Consolidation: One-to-Many**
  - Used to apply a source lookup table change to all affected target rows

Various mapping methods are available depending on the needs of the business.

**One-to-one**: Replicate insert/update/delete operations from source to target

**LiveAudit**: Convert update/delete operations into inserts to create transactional history

**Adaptive apply**: Most commonly used in consolidation scenarios, when updates are made to a row that does not yet exist in the target system, the row will be automatically inserted first. (also known as upsert)

**Summarization**: Mathematically derives a running total of certain data fields.

**Consolidation (One-to-one)**: Gets data from multiple source tables and applies it into one row at target.

**Consolidation (One-to-many)**: Used when one transaction at source affects more than one row at target.
Filtering

- Integrate entire systems or only a subset of data
- Table/row/column-level filtering options available

Users do not have to replicate all data in a system, the scope of replication is fully configurable to the table/row/column level.
Transformations are available so that disparate systems are able to communicate and integrate data with each other. Transformations are configured from a graphical user interface and custom programming is not required.
Character encoding/translations

- Automatic data type mappings for character conversions
Auditing

- Switch all operations into INSERT to keep transactional history
- Capture additional data for full data traceability
  - Type of data change, origin of data change, etc.

JOURNAL CONTROL COLUMNS

- **&CCID**: An identifier for the transaction with the update.
- **&CNTRN**: Source table relative record number.
- **&CODE**: Always “U” for refresh. Always “R” for mirror.
- **&ENTTYP**: Indicates the type of update.
- **&JOB**: The name of the source job that made the update.
- **&JOBNO**: The operating system user Id of the update process.
- **&JOBUSER**: The operating system user at the time of the update.
- **&JOURNAL**: The name of the journal, as described in Properties.
- **&JRNFLG**: Indicates if before image is present.
- **&JRNLIB**: The name of the journal schema.
- **&LIBRARY**: The source table schema or its alias.
- **&MEMBER**: The source table name or its alias.
- **&PROGRAM**: The name of source program that made the update.
- **&OBJECT**: The source table name or its alias.
- **&SEQNO**: The sequence number of this update in the journal.
- **&SYSTEM**: The hostname of the source system.
- **&TIMESTAMP**: Time of the update or refresh.
- **&USER**: The user ID which made the update.

Auditing capabilities of InfoSphere Change Data Capture:

- The LiveAudit mapping method of switching all operations into inserts provides transactional history so data can be tracked to show how data has changed rather than just tracking the data values at specific times.
- Information Server Change Data Capture captures additional control data such as which system made a data change, what data was changed, when the data was changed, etc, to create audit trails of data changes.
User exits

- **Execute custom business logic**
  - React to business events in real time
  - Automate business processes
- **Multiple implementation methods available:**
  - C/C++, Java, stored procedures

On top of the large pre-defined set of functions, user exits are customized logic procedures to fulfill the needs of complex and/or unique business scenarios. User exits can be written in a large variety of programming languages such as C, C++, Java, etc. Each replication process can be configured individually, such as right after inserting new values into a particular field, run a specific user exit.
User exits (continued)

- Exit 2: Derived Columns
- Exit 1: Row Selection
- Exit 3: Derived Columns
- Exit 4: Row Level

Publisher Tables <-> Subscriber Tables
Conflict detection and resolution

• Provides data integrity when multiple systems change the same data simultaneously
• Conflicts can be resolved in various ways:
  • Source wins, target wins
  • By data value
  • Execute user exit

For bi-directional replication or consolidation scenarios where multiple systems can make a change to the same data simultaneously, data conflicts can occur and may cause source and target systems to be out of synchronization. Conflict detection and resolution rules can be configured to resolve data conflicts. Each field can be individually selected to monitor for data conflicts, and several predefined resolution methods are available (such as the data value in the source is always taken as correct, or the larger value is always taken as correct, etc). For additional conflict resolution rules, user exits can also be run once a conflict has been detected.
Change management

- Promote test and development integration processes into production without risk
  - Eliminates potential user error
  - Enables faster rollout of new business processes
  - Rollback capabilities available
  - Changes are tracked for compliance

Once replication processes have been thoroughly tested in development stages, the process doesn’t have to be set up again as a production task, which often leads to errors. Information Server Change Data Capture can promote replication processes from development into production to streamline the deployment process.
Monitoring

- Graphical visualization of replication processes
- Detailed performance statistics to optimize tuning
- Event logs, alerts and alarms
- Exportable throughput and latency statistics

All configuration, administration, and monitoring of data replication processes can be done from a single graphical user interface, which does not have to be constantly managed by the administrator. Alerts can be pre-configured to send emails to the administrator if data latency reaches a certain threshold to surface any potential problems immediately. Full, detailed event logs are also kept for all sources and targets for easy debugging whenever necessary. The monitoring panel also records statistics of how long it takes data to be replicated across systems as well as how many bytes or transactions per second is being pushed through.
Exceptional data integrity

• Data transactions are applied at the target in the same order as it was generated at the source
• Target acknowledges each apply operation to ensure delivery
• No data is lost even if communications link becomes unavailable
• Automatic restart of replication processes after a network failure

InfoSphere Change Data Capture is able to ensure that data transactions are applied at the target in the same order as they were generated at the source. This is a competitive differentiator as other solutions determine the order of data transactions through timestamp, which may cause data integrity issues especially when multiple data transactions occur at the same time. Information Server Change Data Capture determines the order of transactions through the transaction ID that exists in databases by default.

If communications become unavailable during operation, no data transactions are lost. Once communication becomes available again, the source knows where to start scraping the logs from so no data was lost.
Reliability and recoverability

1. Scrape

Journal/Log

Source Tables

2. Push

Push Engine

3a. Apply

Apply Engine

Target Tables

What was the last transaction you successfully applied?

3b. Confirm

Metadata

Discovering the value of IBM InfoSphere C
Persistency

• InfoSphere CDC may initiate a normal shutdown and end mirroring after:
  • Communications error
  • Instance termination
  • Deadlock scenarios

• To automatically restart continuous mirroring of subscriptions after a normal shutdown, you can mark the subscriptions as persistent
  • InfoSphere CDC will attempt to automatically restart continuous mirroring at regular intervals
  • Continuous mirroring for a persistent subscription can automatically restart in response to a normal or abnormal (recoverable) termination for the above mentioned conditions

CDC may initiate a normal shutdown and end mirroring after:

Communications error

For persistent subscriptions that were active when the termination occurred, continuous mirroring automatically restarts when communications are re-established.

Instance termination

For persistent subscriptions that were active when the termination occurred, continuous mirroring automatically restarts when the subscription server is restarted.

DB2 Deadlock scenarios (CDC apply)

For persistent subscriptions, if replication was terminated due to a deadlock timeout on the target, continuous mirroring automatically restarts

To automatically restart continuous mirroring of subscriptions after a normal shutdown you can mark the subscriptions as persistent

CDC will attempt to automatically restart continuous mirroring at regular intervals

Continuous mirroring for a persistent subscription can automatically restart in response to a normal or abnormal (recoverable) termination for the above mentioned conditions
InfoSphere CDC → DataStage/QualityStage integration

**Option 1: Database Staging**
- InfoSphere CDC captures changes made to source database
- InfoSphere CDC writes changes to a staging table.
- DataStage reads the changes from the staging table, transforms and cleans the data as needed.
- Update target database and internal tracking with last bookmark processed.

**Option 2: MQ-based integration**
- InfoSphere CDC captures/collects changes made to remote database
- Captured changes written to MQ
- DataStage (via MQ connector) processes message and passes data off to downstream stages
- Updates written to target warehouse

**Option 3: File-based**
- InfoSphere CDC captures change made to source database
- InfoSphere CDC writes each transaction to a file
- DataStage reads the changes from the file
- Update target database with changes

**Option 4: Direct connect**
- InfoSphere CDC captures and collects changes made to source database
- Captured changes passed to CDC for DataStage engine
- DataStage transaction aware stage processes transactions and passes data off to downstream stages
- Update target database with changed data

Discovering the value of IBM InfoSphere CI
© 2013 IBM Corporation
This scenario is based on DS controlling when the request for delta changes occur.
The refresh happens when the custom operator invokes the transformation server to collect the changes for a subscription.
InfoSphere Change Data Capture - Recap

• InfoSphere Change Data Capture provides real-time changed data capture across the enterprise.

• Key benefits:
  • Low impact
    • Does not impact performance and requires no changes to applications
  • Heterogeneous
    • Integrates data from all platforms and databases
  • Flexible
    • Supports any topology
  • Easy to use
    • Fast deployment with low risk
  • Integrated with Information Server
    • Single solution for all data integration requirements

As a summary, Information Server Change Data Capture captures changed data from production systems and delivers it across the enterprise in real time.

The reason for why customers should choose Information Server Change Data Capture is that:

- it does not interact directly with the database itself and hence does not impact the performance of mission-critical applications, data capture can occur continuously throughout the day and eliminate batch windows

- it supports the largest variety of server platforms and database systems

- data replication processes can be applied throughout the enterprise regardless of how its architecture complexity

- it does not require database skills or scripting skills while most configuration tasks are automated
Replication Tools
Monitoring Q Replication
Here’s the big picture again. This presentation focuses on the replication monitoring, including reporting and alert monitoring/notification.
The Replication Center Operations tasks include status and progress reports for Q Capture and Q Apply.
Q Capture Activity Reporting

1. Q Capture stores runtime statistics in the control tables at the source server
   1. IBMQREP_CAPMON and IBMQREP_CAPQMON
   2. The value for MONITOR_INTERVAL in the IBMQREP_CAPPARMS table determines how often Capture inserts to the monitor tables.
   3. The value for MONITOR_LIMIT in the IBMQREP_CAPPARMS table determines how much monitor data is kept.

2. Q Capture stores informational, warning, and error messages in
   1. IBMQREP_CAPTRACE table at the source server
   2. The value for TRACE_LIMIT in the IBMQREP_CAPPARMS table determines how much trace information is kept.
   3. Q Capture log file at the source server

Q Capture stores statistics and messages in DB2 tables at the source server. These tables can be queried directly or displayed in the Replication Center. Replication Center reports have a SHOW SQL button that you can select to capture the SQL used to generate the reports. This can be useful if you wish to generate standard reports or reports on demand without going through the Replication Center.
Q Capture Monitor Tables

**IBMQREP_CAPMON**
Statistics on log records processed

<table>
<thead>
<tr>
<th>MONITOR_TIME</th>
<th>ROWS_PROCESSED</th>
<th>TRANS_PROCESSED</th>
</tr>
</thead>
</table>

**IBMQREP_CAPQMON**
Statistics on groups of subscriptions (SENDQ)

<table>
<thead>
<tr>
<th>MONITOR_TIME</th>
<th>SENDQ</th>
<th>ROWS_PUBLISHED</th>
<th>TRANS_PUBLISHED</th>
</tr>
</thead>
</table>

Each row in the monitor tables has a timestamp named MONITOR_TIME which is the time the row was inserted. The statistics are not cumulative. Q Capture resets the counters in memory to zero after inserting rows in the monitor tables.
The MONITOR_INTERVAL controls how often a monitoring record is inserted into the monitor table. For this Q Capture, the MONITOR_INTERVAL is 10 seconds.
The Q Capture Latency measure how far behind Q Capture is from the end of the DB2 log. To produce this report, we flooded Q Capture with changes to produce some numbers that were more interesting than 3 and 4. This test was done on the student machine – 1G memory, 1CPU running DB2, Oracle, MS SQL Server, and WebSphere MQ with all databases on 1 disk drive, so there is very little chance that Q Capture can keep up with 10s of thousands of changes per minute.
Q Apply Activity Reporting

1. Q Apply stores runtime statistics in the control tables at the target server
   1. IBMQREP_APPLYMON
   2. The value for MONITOR_INTERVAL in the IBMQREP_APPLYPARMS table determines how often Q Apply inserts to the monitor tables.
   3. The value for MONITOR_LIMIT in the IBMQREP_APPLYPARMS table determines how much monitor data is kept.

2. Q Apply stores informational, warning, and error messages in
   1. IBMQREP_APPLYTRACE table at the target server
   2. The value for TRACE_LIMIT in the IBMQREP_APPLYPARMS table determines how much trace information is kept.
   3. Q Apply log file at the target server

Q Apply reporting is similar to Q Capture reporting. You can capture the SQL and run it natively if desired.
Q Apply Monitor Table

<table>
<thead>
<tr>
<th>IBMQREP_APPLYMON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics on groups of subscriptions (RECVQ)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MONITOR_TIME</th>
<th>RECVQ</th>
<th>ROWS_APPLIED</th>
<th>TRANS_APPLIED</th>
</tr>
</thead>
</table>

Each row in the monitor tables has a timestamp named MONITOR_TIME which is the time the row was inserted. The statistics are not cumulative. Q Apply resets the counters in memory to zero after inserting rows in the monitor tables.
Q Apply Throughput

Throughput is a count of the transactions applied and the number of rows affected by those transactions in a monitor interval.
End-to-end latency is the sum of Q Capture latency, queue latency, and Q Apply latency. This is a measure of the time from when the transaction is committed at the source to the time the transaction is committed at the target.

In this report, you can see that there is a serious problem with the SENDQ/RECVQ that needs to be researched. And, Q Capture latency does not look good either! On our test system, we deliberately stopped and restarted Q Capture and stopped and started our queues to produce some bad latency numbers. The reports weren’t very interesting when all they showed was 1’s and 0’s. Note that these values are all in milliseconds.
Q Replication dashboard

Real-time monitoring tool can be downloaded from the web

The dashboard is a small window that shows the status of the Q Capture and Q Apply programs at a set of DB2 databases and DB2 z/OS subsystems. It uses a series of green and red circles to display this information. You can also drill down on each system and monitor Q Capture activity, Q Apply activity, which is very useful during problem determination.
Q Replication Dashboard detail

Dashboard can drill down to the table level:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>State</th>
<th>Type</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TJCACPLOB1</td>
<td>Active</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJCACPLOB2</td>
<td>Active</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJCACPLOB3</td>
<td>Active</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJCACPLOB4</td>
<td>Inactive</td>
<td>XML Pub</td>
<td></td>
</tr>
<tr>
<td>TJCACPLOB5</td>
<td>Inactive</td>
<td>Uni (Tgt)</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJCACPLOB6</td>
<td>Active</td>
<td>P2P</td>
<td>RED2(ASN)</td>
</tr>
<tr>
<td>TJCACPLOB8</td>
<td>Partially active</td>
<td>P2P</td>
<td>RED2(ASN), GREEN2(ASN)</td>
</tr>
<tr>
<td>TJCACPLOB9</td>
<td>Inactive</td>
<td>Uni (Src)</td>
<td>RED2(ASN)</td>
</tr>
</tbody>
</table>

The dashboard allows you to drill down from the system view to a table view as shown in the slide. You can see that some tables are active, some are partially active and some are inactive.
1. New Web-based monitoring and administration tool for Q Replication
The Replication Alert Monitor

The Replication Alert Monitor is a replication program (asnmon) that runs continuously, checking Capture and Apply servers for error and warning conditions chosen by the Replication Administrator.

Alerts are sent as e-mail messages via an SMTP mail server. The Replication Administrator identifies the contacts or group of contacts to be notified for each condition.

Alerts are also logged in an Alert Monitor control table.

The Alert Monitor program does not need to be located on the source or the target server. It can be running on another server (where DB2 is located) and monitor multiple SQL Capture/Apply programs and/or multiple Q Capture/Apply programs.
What are the Alert Monitor requirements?

1. Alert Monitor platforms
   - DB2 Universal Database for OS/390 and z/OS V7 or V8
   - DB2 Universal Database for Linux, UNIX and Windows 9.1

2. Alert Monitor prerequisites
   - DB2 (DRDA) connectivity to monitored Q Capture and Q Apply servers
   - Database Administration Services (DAS) installed and configured on all monitored Q Capture and Q Apply servers
     - Required to determine whether the replication programs are running
     - Not required for other monitoring elements
     - Requires additional installs of code for DB2 Universal Database for OS/390 and z/OS
   - Database Administration Services (DAS) installed and configured on the server where the Alert Monitor runs
     - Required to send e-mail notification

For DB2 UDB OS/390 and z/OS V8, need the Management Clients package

For DB2 UDB OS/390 and z/OS V7, need the Database Administration Services FMID

DAS on z/Os is started and runs in the UNIX System Services (USS), so configuration is needed after installation.
How Does the Alert Monitor work?

1. The Replication Center is used to define alert monitoring for Q Capture and Q Apply.

2. The Replication Administrator performs the following operations
   - Create Monitor Control Tables
   - Set thresholds and choose events for alert monitoring
   - Identify contacts and groups of contacts for notification via an SMTP mail server
   - Start and stop the Alert Monitor

3. The Replication Center generates SQL and replication command scripts to stop and start the Alert Monitor and to check status. The scripts are run using the Replication Center and can also be saved for reference or reuse.

You can also use asnclp to define alert monitoring.
What events/conditions can be monitored?

1. Q Capture and Q Apply status – alert if programs not running
2. Q Capture and Q Apply error or warning messages – alert if a message is generated by the programs
3. Latency – alert if administrator-set thresholds are reached for Q Capture, Q Apply, or End-to-End latency
4. Memory usage – alert if memory usage exceeds administrator-set threshold
5. Transaction size – alert if a transaction takes more memory than the threshold set by the administrator
6. Queue Depth – alert if number of messages on the receive queue exceeds threshold set by administrator
7. Exception – alert if Q Apply processes an exception (skipped change, conflict skipped, other error)
The Alert Monitor queries DAS for program information (Capture or Apply programs up or down) and issues SQL to the monitor and trace tables for all other alerts.
Q Replication Utilities and System Commands

1. System commands for operating Q Capture, Q Apply, and the Alert Monitor

2. Troubleshooting commands and utilities for diagnosing problems
System Commands

1. Q Capture
   - asncap – start Q Capture
   - asnccmd – work with a running Q Capture program, including query status and stop

2. Q Apply
   - asnapp – start Q Apply
   - asnacmd – work with a running Q Apply program, including query status and stop

3. Monitor
   - asnnmon – start the Alert Monitor
   - asnmcmd – work with a running Alert Monitor program, including query status and stop

These commands can be issued from a command prompt or from the Replication Center. They are usually only used during testing. Most customers will want to automatically start the replication programs when the system is started.
More System Commands

1. Q Capture and Q Apply on Windows
   - asncrt – create Windows services for replication programs
   - asnlist – list Windows services created by asncrt
   - asndrop – remove Windows services created by asncrt

2. Q Apply Password management on DB2 for Linux, UNIX, and Windows
   - asnpwd – create an encrypted list of passwords for Q Apply to use when doing a full refresh

These steps are done during configuration of a replication scenario. The Q Apply password command is run on the Q Apply server and provides the userid/pwd used for a connection to the source server so that Q Apply can load the target table.
Invoking System Commands

1. System commands are invoked from the command line on
   • DB2 Universal Database for Linux, UNIX, and Windows
   • UNIX System Services shell on z/OS
   • Modify command on z/OS

2. Sample syntax
   • Linux, UNIX, Windows
     \( \text{asqcap capture_server=DB2SYSA capture_schema=ASN} \)
     \( \text{asqacmd apply_server=DB2SYSB apply_schema=ASN1 stop} \)
   • Z/OS
     \( /F \text{Qcapstartedtask,STOP} \)

The system commands can be invoked from the command line on Linux, UNIX, Windows or from the SDSF command line on z/OS.
Replication Utilities

1. asnqanalyze
   • Detailed or summary report of replication environment

2. asntdiff
   • Compares a replication source table to a replication target table and reports the differences

3. asntrep
   • Uses the information from asntdiff to reconcile differences between a replication source and target table

4. asntrc
   • Traces Q Capture, Q Apply, and Alert Monitor programs

5. asnqmfmt
   • Formats replication messages for troubleshooting

These utilities are included with the Q Replication product.
Utility -- asnpanalyze

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows

2. DB2 connectivity to Q Capture or Q Apply servers is required
   • DB2 Connect is needed if any of the servers are DB2 on z/OS
   • Password file is required for connection to remote servers
     The asnpwd system command is used to create the password file

3. Output is an html report

   Example:
   asnpanalyze –db DB2SYSA –la detailed
   asnpanalyze with no parameters will display help

IBM Support will often ask for an analyzer report when you call in a problem. The report gives them a picture of your environment and reduces the need to ask you lots of questions. The analyze report is also good documentation, since it provides a snapshot of your environment.
Utility asnqanalyze Sample Report

Notice that the analyzer is not just a report of the replication environment. The asnqanalyze program also checks for known problems (like incorrect indexes) and lists any that it finds. This is a good way to periodically verify your replication environment.
Utility -- asntdiff

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows or z/OS USS
2. DB2 connectivity to Q Capture and Q Apply servers is required
   • DB2 Connect is needed if any of the servers are DB2 on z/OS
   • Password file is required for connection to remote servers
     The asnpwd system command is used to create the password file
3. Output is a DB2 table named ASN.TDIFF at the Q Capture server
   • The difference table, ASN.TDIFF, must be manually dropped

   **EXAMPLE:**
   
   db2 connect to <Qcapdatabase>
   db2 drop table asn.tdiff
   asntdiff db=DB2SYSA where='SUBNAME='CUSTMER0001''

This utility does not compare 2 DB2 tables – it verifies the source and target tables of a subscription (SUBNAME) and takes into account the row/column subsetting defined for that subscription.
How does asntdiff work?

1. Uses the SUBNAME in the WHERE parameter to find the column mappings and search conditions (predicate) in the Q Capture control tables.
2. Creates ASN.TDIFF based on the column mappings
3. Selects rows from the source table based on the column mappings and search conditions
4. Compares the source table rows to the target table (check sum scheme is used for efficiency)
5. Puts differences in the ASN.TDIFF table
6. Writes messages to the console

This utility may not complete if your source/target tables are very large.
Utility -- asntrep

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows or z/OS USS

2. DB2 connectivity to Q Capture and Q Apply servers is required
   • DB2 Connect is needed if any of the servers are DB2 on z/OS
   • Password file is required for connection to remote servers
     The asnpwd system command is used to create the password file

3. Input is the DB2 table named ASN.TDIFF at the Q Capture server
   • The difference table is created by the asntdiff utility.

   EXAMPLE:
   db2 connect to <Qcapdatabase> asntrep db=DB2SYSA where='SUBNAME='CUSTMER0001’"

Asntrep is a repair utility. If you run asntdiff and discover differences between your source and target table, then you can run asntrep to correct those differences. You should use this utility with caution, especially if users are updating the source and/or target tables. If updates are occurring to the 2 tables when you run asntdiff, then the differences may not really be differences. The differences may reflect changes that occurred in the target table between the time asntdiff selected from the source and the time that asntdiff selected from the target. To be completely accurate in the comparison, you will have to stop all activity at the source, stop all user activity at the target, wait for Q Apply to process all changes, then run asntdiff.
How does asntrep work?

1. Uses the SUBNAME in the WHERE parameter to find the column mappings in the Q Capture and Q Apply control tables.
2. Selects from ASN.TDIFF to get the differences
3. Inserts missing rows to the target
4. Deletes extra rows from the target
5. Updates mismatched rows in the target
6. Writes messages

Notice that all repair work is done on the target to make the target match the source. For bidirectional or peer-to-peer, the source is the database name used when you issued asntdiff.
Utility - asntrc

1. Traces Q Capture, Q Apply, or Monitor programs
2. Dynamically turned on and off while traced programs are running
3. Used at the direction of IBM Support to diagnose problems

EXAMPLE
asntrc on –db DB2SYSA –schema ASN –qcap
asntrc fmt –db DB2SYSA –schema ASN -qcap
asntrc off –db DB2SYSA –schema ASN –qcap

This utility should only be used at the direction of IBM Support. There is also a DEBUG parameter for Q Capture and Q Apply that will display diagnostics. The advantage of asntrc is that tracing can be turned on with stopping the replication programs.
Utility -- asnqmfmt

1. Platforms – DB2 Universal Database for Linux, UNIX, and Windows or z/OS USS
2. WebSphere MQ client or server access to the Q Apply receive queue (RECVQ)
3. Asnqmfmt gets a message from the Q Apply receive queue and presents it in XML format. This is not an application interface – it is intended for diagnostics.

   EXAMPLE:
   asnqmfmt SYSB.RECVQ QMSYSB

This utility displays the replications messages on the restart and data queues.
Replication Uses Cases
Customers Require Different Types Of Data Integration

Each type is like a different tool: hammer, wrench, screwdriver, and saw.

As the highlighted graphic shows, Replication transmits data between data sources. Once the data sources are initially populated then replication sends only the changed data. The intent of Replication is to keep the data sources synchronized. The graphic shows synchronization among 2 data sources, but it can be more than 2 sources. Also, the synchronization can occur in one direction, called uni-directional replication. Or the synchronization can flow in both directions and this is called bi-directional or peer-to-peer replication. Bi-directional is used when it's between 2 sources and peer-to-peer is used to refer to bi-directional synchronization among 2 or more sources.

Replication has unique capabilities which positions it among the integration styles shown on this slide. Replication is built for high-volume and low-latency data movement. This makes it ideal for creating a back-up copy of data for rapid cutover in case of failure of the server with the primary copy. It also enables multiple application instances, each running on a different server, to run on a shared set of data which is synchronized between the servers. This is important for high availability applications.

Replication has limited transformation capabilities. This is excellent for applications where a second copy of data is needed, a copy that looks just like the original, with no transformations. Examples include the applications I just mentioned, as well as reporting or monitoring applications. However, where data must be heavily transformed such as in the creation of a data warehouse then a consolidation style of integration is a better fit. Extract, Transform and Load, or ETL, products such as DataStage are built to provide data movement with extensive transformation function.

Replication also positions well with Federation. Federation is used when the application needs to access the data source directly. This may be due to many reasons including legal reasons that preclude copying the source data, cost reasons for not wishing to maintain a copy of data that is either prohibitively large or infrequently used. On the other hand, if the data will be accessed heavily, there
Replication used in many ways:

High Availability – say you have a production system on mfg floor. But need up to the minute reporting, e.g., monitoring of processes. But don’t want to impact the production system.

Roll-up – say each state of union has a copy of a database that pertains to their own environment. But also need a central copy of this data for federal reporting.

Peer to peer – Take a large application that is accessed by 10’s thousands of users. One way to handle is to split the application across servers each with a copy of data. Some users access #1, some access #2. Changes in #1 are replicated to #2 and vice versa. One critical aspect of peer to peer is conflict detection and resolution.

Distribution
The reverse of roll-up. There is a central copy. An insurance company has claims and policy information that must be replicated to agents in the field.
Replication Tips and Best Practices
QUESTIONS