


TOC**INDEX****VIEW****F08 Using MQSeries From DB2 Applications**
Connie Nelin, Senior Technical Staff Member, IBM

Many DB2 customers use messaging, queuing or publish/subscribe in their application environments. Such systems have a wide variety of uses, from linking together disparate applications, to disseminating real-time information, to integrating communications with external partners. Often, such uses combine database operations with messaging operations in the same applications. This presentation will explain how new features in DB2 can both simplify application development and leverage the combined power of DB2 and MQSeries.

F08

Using MQSeries From DB2 Applications

Connie Nelin

The logo features a central green rounded rectangle with a purple border containing the text "IBM Data Management Technical Conference". This rectangle is surrounded by a decorative arrangement of green circles of various sizes, some solid and some with a white outline, creating a bubbly, organic effect.

IBM Data Management Technical Conference

Anaheim, CA

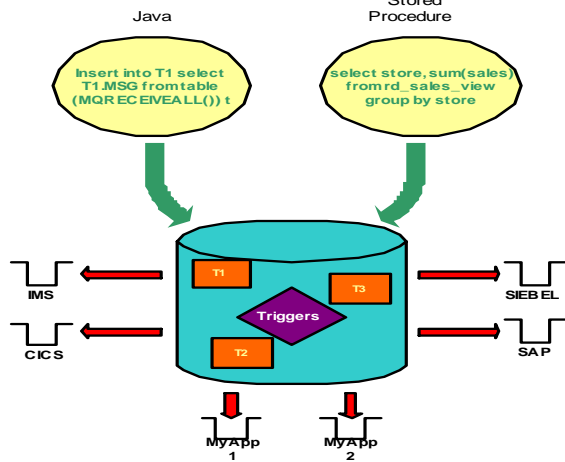
Sept 9 - 13, 2002

Outline

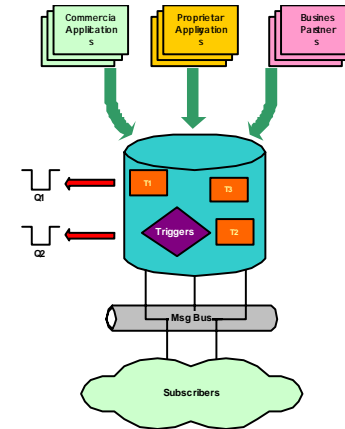
- A Few Scenarios
- About MQ Series
- The DB2 MQSeries Functions
- XML Extender
- XML Messages with MQSeries
- What's Next

A Sampling of Scenarios

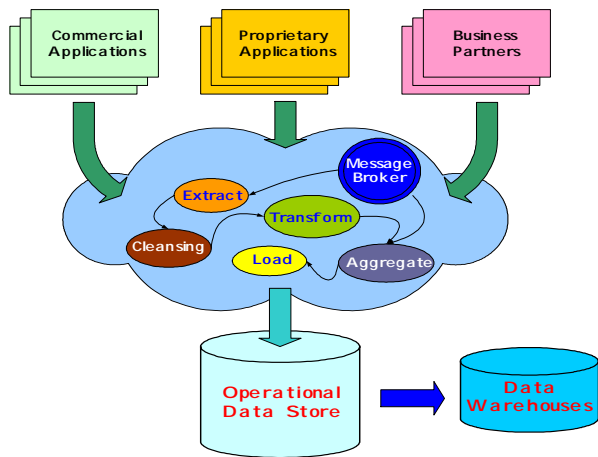
Query Integration



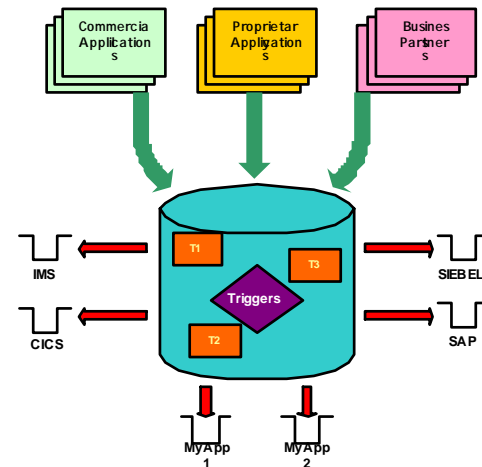
Event Notification



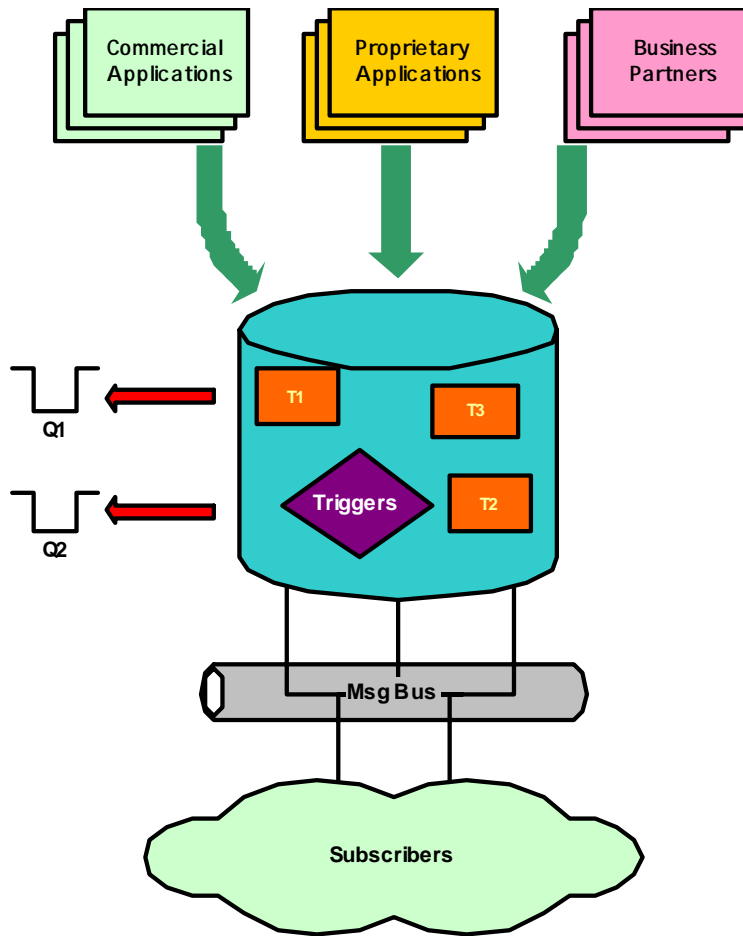
Information Aggregation



Information Forwarding

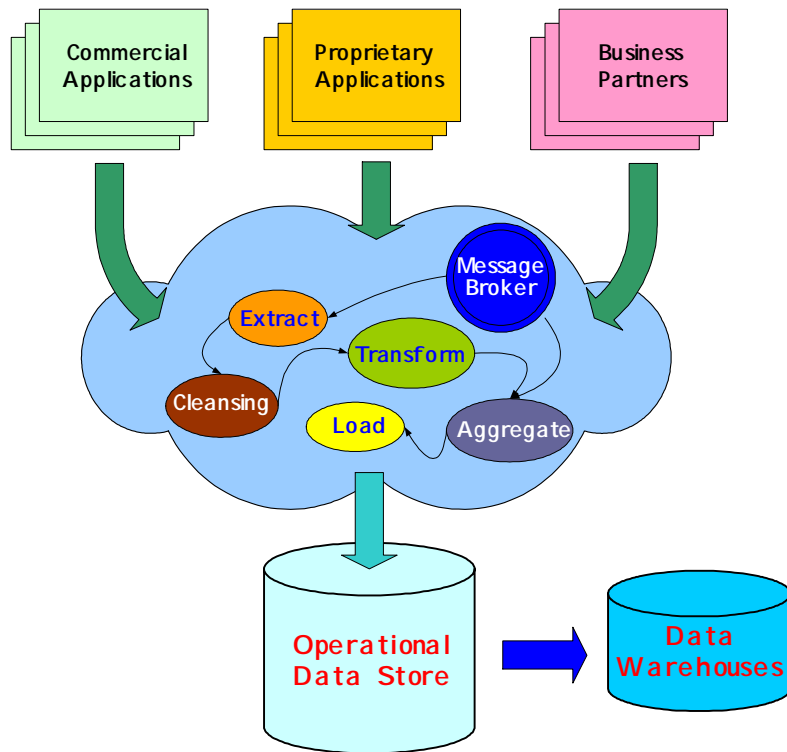


Event Notification



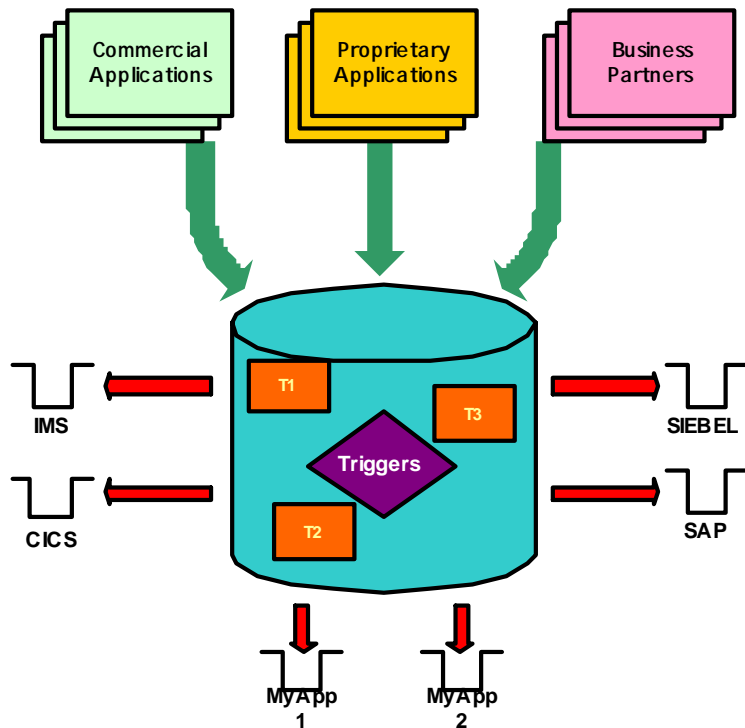
- Messages representing interesting events are automatically generated by triggers.
 - Events represent that something happened
 - Do not usually contain a lot of data.
 - Used to notify interested subscribers or applications to take action.
- Examples
 - An inventory management system notifying low inventory
 - A financial institution notifying applications of new research report
 - A lab running software may wish to notify to researchers when lab results are complete.

Information Aggregation



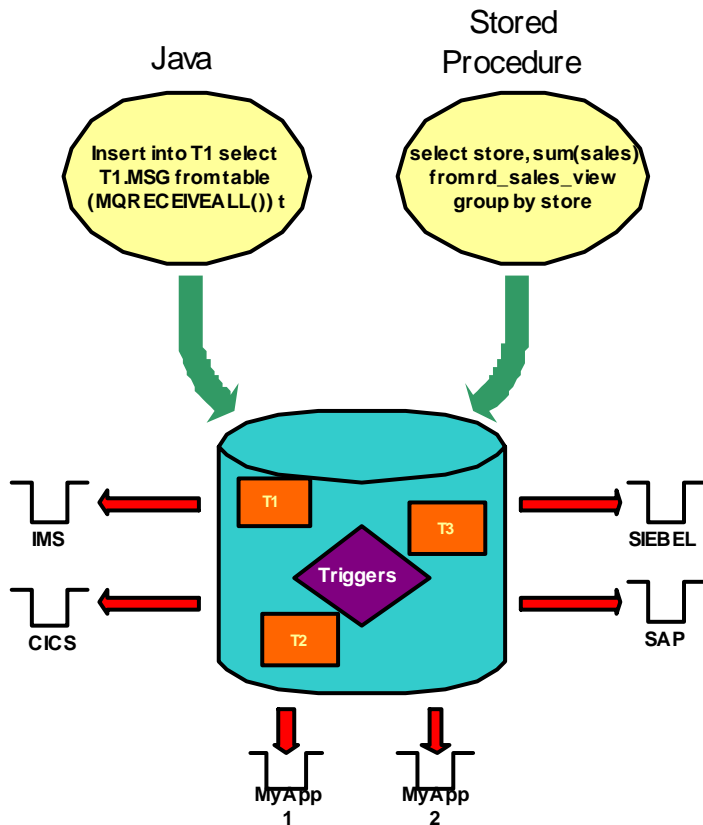
- Messages automatically processed to maintain an accurate view of operational activities.
 - Information from many sources
 - Information may represent real-time snapshots or batches.
 - An external message broker such as MQSI may be used to receive, process, and store the information.
 - DB2 Data Warehouse manager can be used to coordinate the collection and process message data.
- Examples
 - A corporate wide roll-up of financial transactions
 - An insurance company may feed a data warehouse from externally provided data.
 - A manufacturing company may want to collect process information from multiple machines for trend analysis

Information Forwarding



- **Changes to important information may automatically be forwarded to downstream systems.**
 - The message sent may be enriched beyond the data that changed.
 - Downstream systems may commercial applications or transaction systems such as IMS or CICS.
- **Examples**
 - A proprietary banking application inserts a new customer and a message containing the information is forwarded to a CRM application.
 - After a batch load of new risk information, a financial institution may forward high risk changes to risk management applications.
 - As sales people re-synchronize order data from their pervasive devices with the office database, transactions on external systems are executed.

Query Integration



- **Developers incorporate messaging operations into SQL queries from any environment.**

- Rich set of scalar and table functions.
- Support for text (big & small), delimited, and XML messages.
- May be used wherever functions may be used; can construct views for common usages.
- Application executes integrated operations that produce or utilize messages.

- **Examples**

- A WebSphere developer wants to display the value of unprocessed sales orders from a sales queue.
- A researcher writes a simple stored procedure to take messages from the default queue and store them into a table for analysis.
- A developer wishes to generate a report containing a union of processed and unprocessed sales data.

About the WebSphere MQ Family (formerly MQSeries Family)

- WebSphere MQSeries
 - Base messaging servers and clients provide once, and once only, message and queuing capabilities on over 35 platforms.
- WebSphere Adapters (formerly MQSeries Adapter Offering)
 - Provides a framework and tools that build and customize MQSeries adapters for existing and new, pre-packaged or custom-developed applications.
- WebSphere MQSeries Everyplace
 - Extends the capabilities of MQSeries base messaging to remote servers and to mobile workers using laptops, PDAs and telephones
- WebSphere MQ Integrator (formerly MQSeries Integrator)
 - Combines a one-to-many connectivity model, plus transformation, intelligent routing and information flow modelling. It facilitates the development of new application services that comprise the functions of multiple, disparate existing business systems.
- MQSeries Workflow
 - A business process management system which facilitates the rapid development and management of the business processes that integrate the IT and organizational infrastructure of a company.



WebSphere MQSeries API

- APIs for exchanging messages:
 - MQI
 - Provides full access to the underlying messaging implementation
 - Available for all key languages and environments.
 - JMS or Java Message Service
 - Java standard providing much of the function available through the MQI.
 - AMI, or Application Messaging Interface
 - Simplifies the handling of messages with a higher level of abstraction : service points & policies
 - Separates message actions and definitions of how actions should be carried out
- All of the APIs can interoperate.



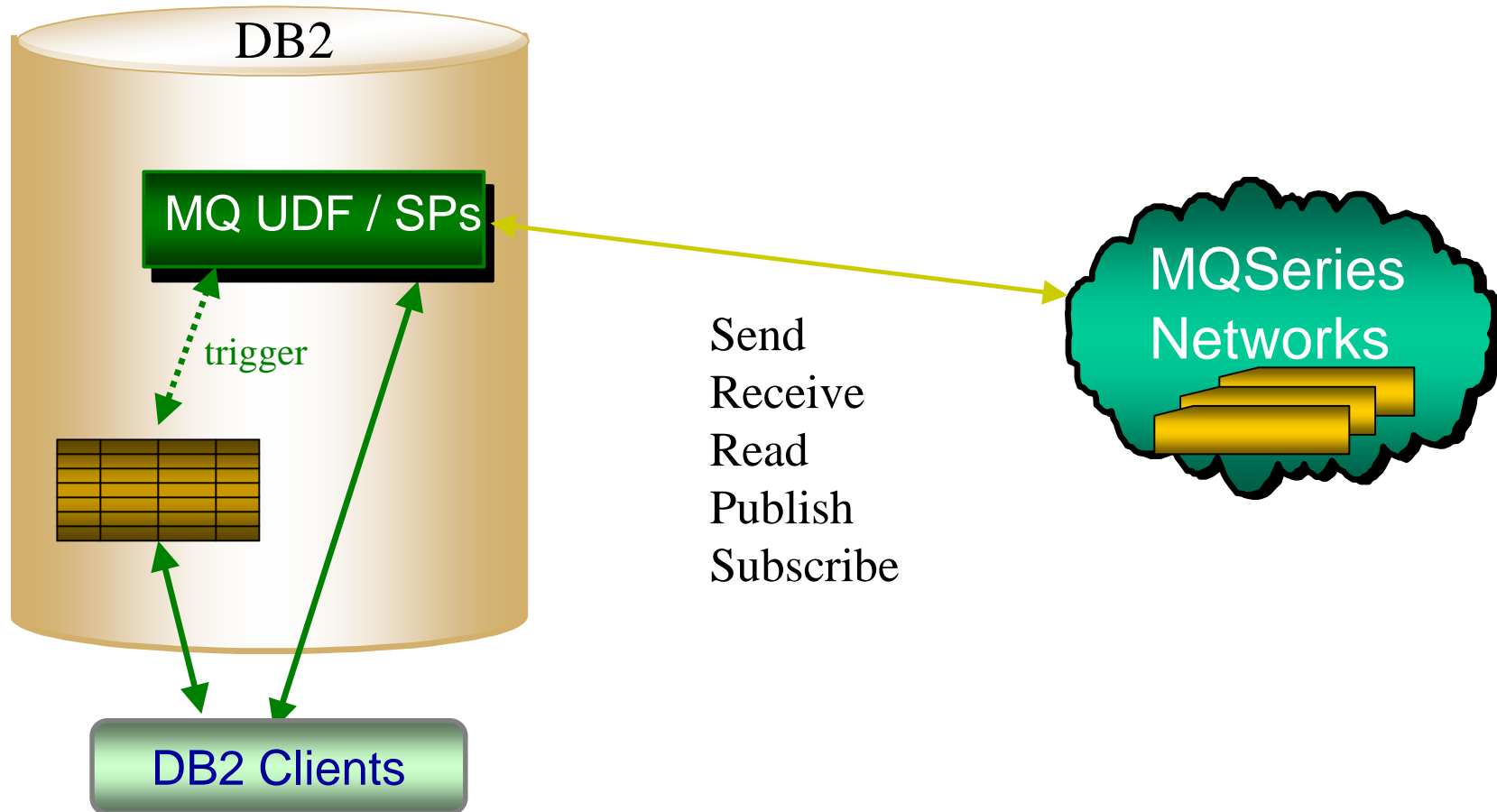
MQSeries Messaging Styles

- Three basic models
 - Datagrams / send and forget
 - Send to single location / no expected reply
 - Publish/Subscribe
 - One or more publishers send message to publication service which distributes it to interested subscribers
 - Request/Reply
 - Like datagram but expect reply

DB2 / MQ Integration

- Provide a simple way to access MQSeries from database applications
 - Leverage the set-oriented nature of SQL with messaging
 - Provide a new kind of data source, leveraging the heterogeneous, federated capabilities of DB2.
 - Simplify the creation of operational data stores and data warehouses for business intelligence
 - Publish database changes
- Provide a gentle introduction to the MQSeries Family to database programmers and administrators
 - Integration with MQSeries, MQSeries Pub/Sub, MQSI

DB2 / MQ Integration Today



DB2 / MQ Series Integration

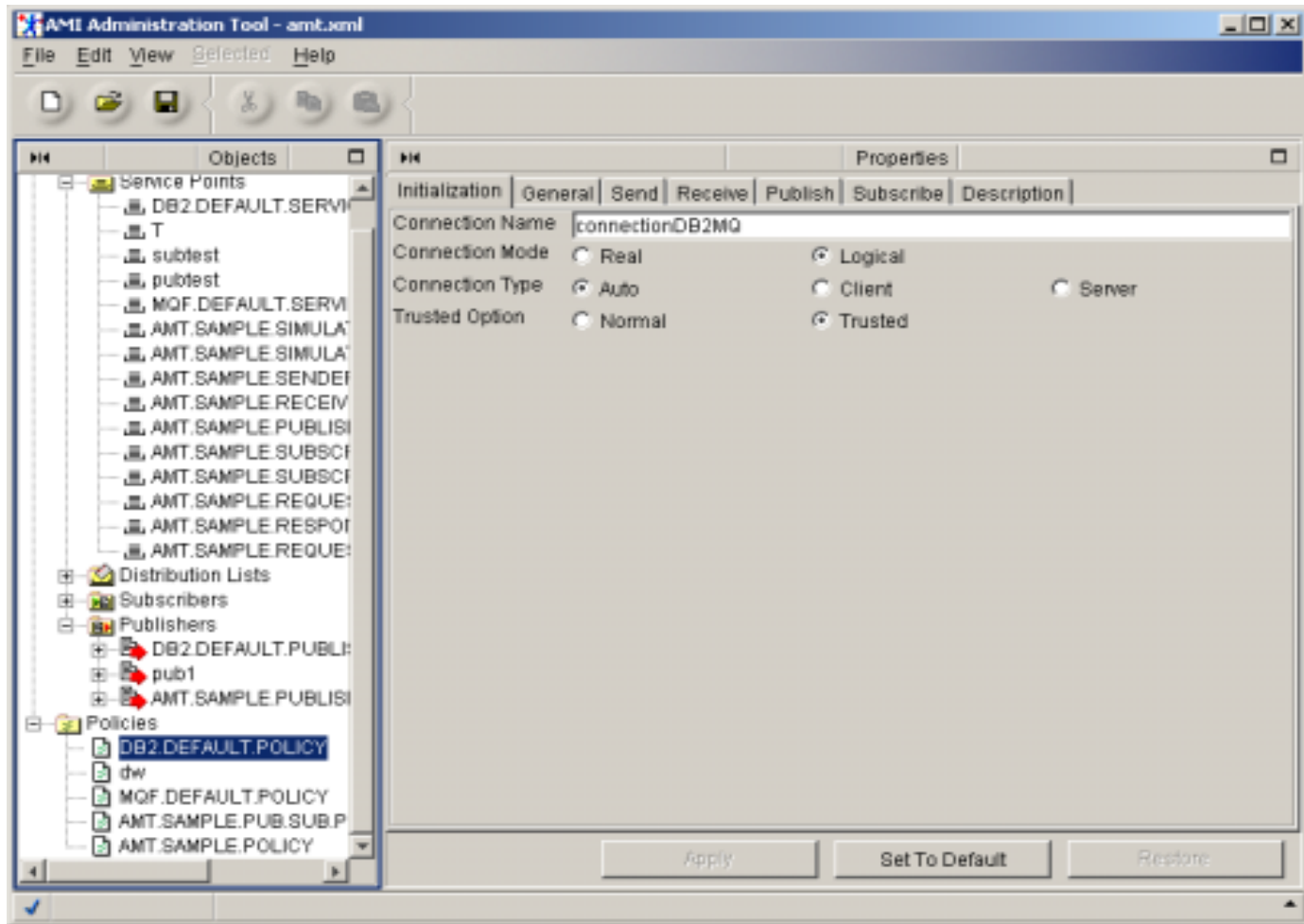
- Database programmers can invoke messaging functions in a familiar environment.
 - MQ integration functions can be used as part of SQL statement
 - Can be used in applications, triggers and stored procedures
 - Available from any language supporting SQL
 - C, Java, C++, SQL, ...
 - Support for datagrams, request/reply, pub/sub
 - Support for XML messaging through the DB2 XML Extender



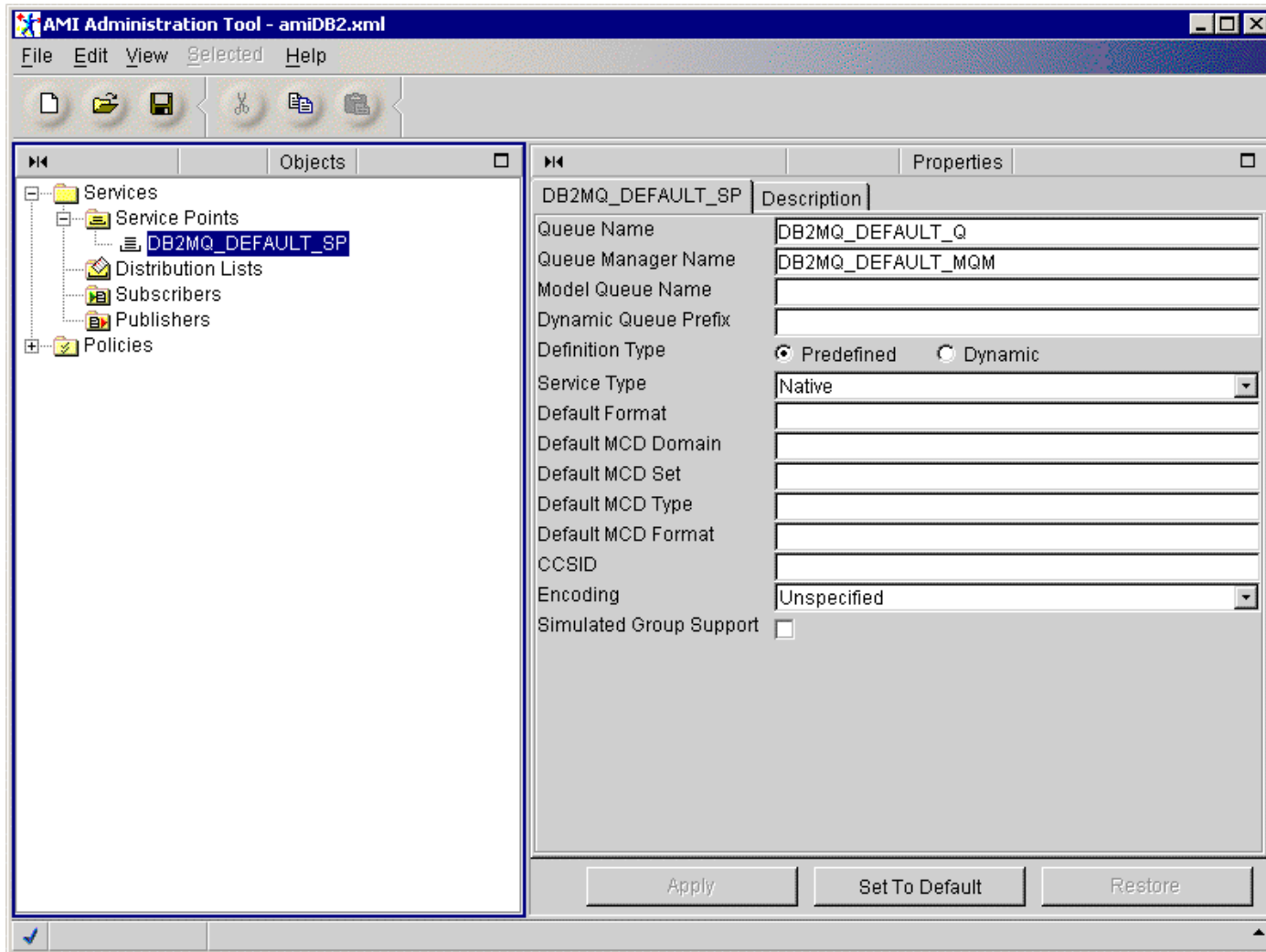
DB2 / MQ Series Integration

- Support for multiple message contents models
 - Simple structures (positional or delimited)
 - Unstructured strings
 - Self-describing XML
- Based on MQ AMI APIs
 - Optionally specific of location/service point and policy
 - Location is the logical name describing the destination or source of a message
 - Policy is the logical name describing message quality of service
 - Use AMI Administration tool to configure

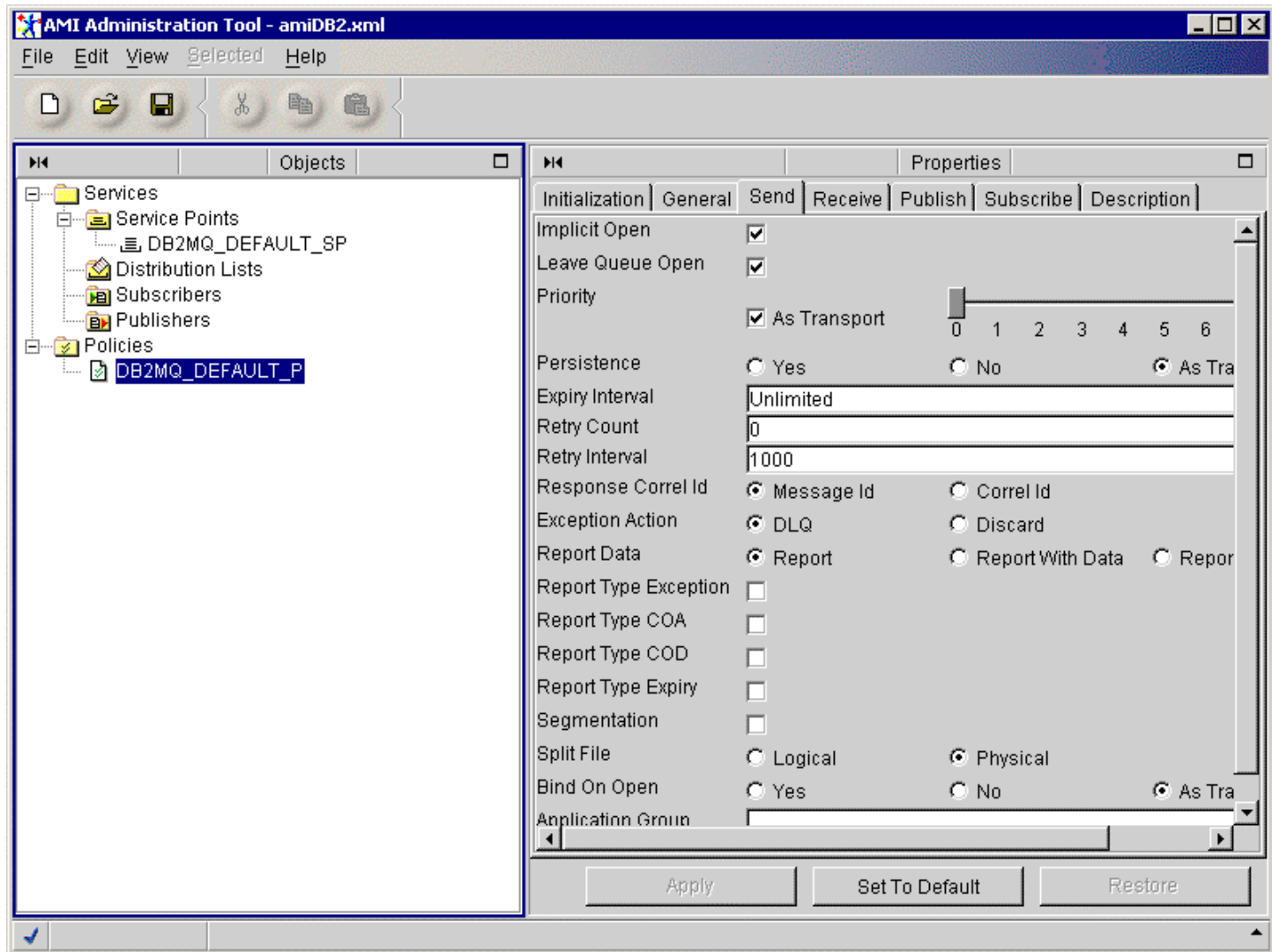
Administer with AMI Administration Tool



AMI - Service Points



AMI Policies



DB2 / MQ Series Integration

- The basic functions are implemented as User Defined Functions (UDFs)
 - Written in C / Optimized for performance
 - Win/Unix:
 - Schema: DB2MQ (full names of functions DB2MQ.*)
 - Must be installed using the ENABLE_MQFUNCTIONS
 - 390
 - Schema DB2MQ1C and DB2 MQ2C
 - Must install APAR: PQ59549
- The XML related UDFs depend on the XML Extender
 - Must be installed with the ENABLE_MQXML
 - XML Extender is a separate, optional installation that must precede this command
- Default queues, service points and policies are provided



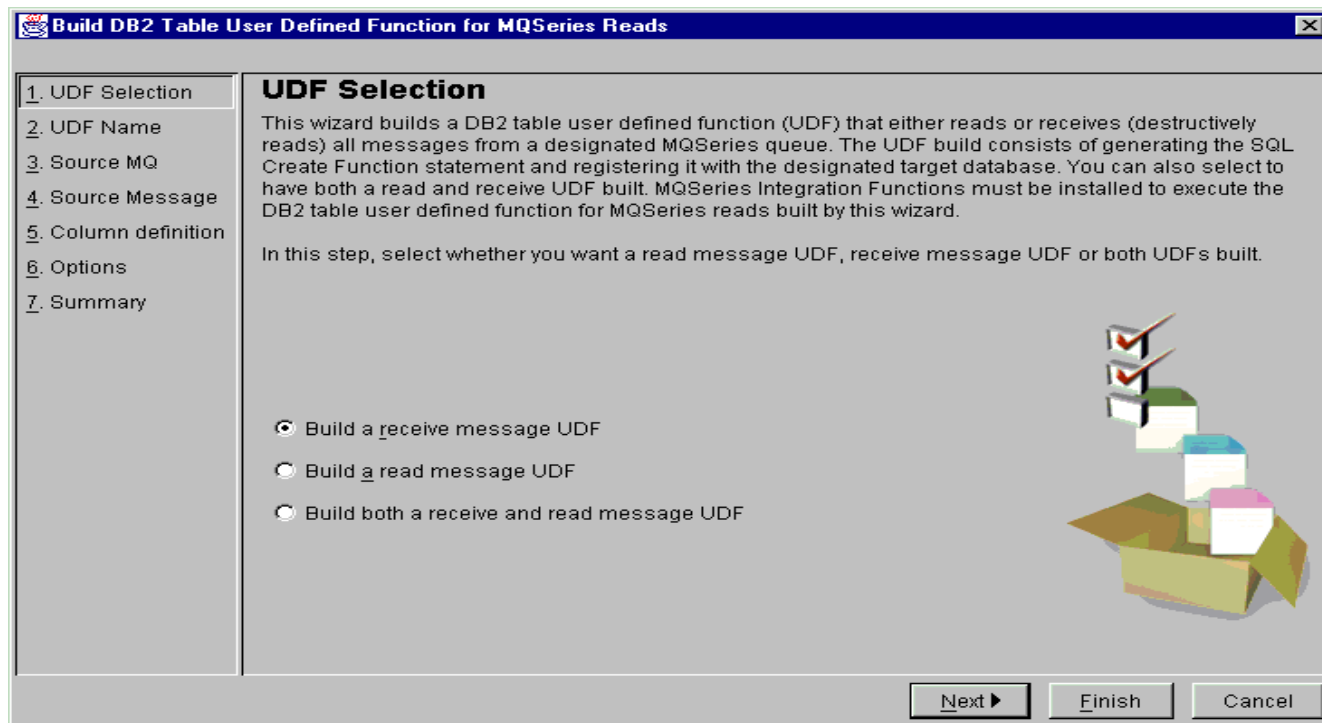
DB2 / MQ Series Integration

- MQAssist Wizard to facilitate construction of table functions and views
- Allows MQSeries as a source of information for a DB2 Data Warehouse
- Available on Windows NT, Windows 2000, AIX, Sun Solaris, HP-UX, 390
 - Coming on Linux
- DB2 Win/Unix ships limited use MQSeries

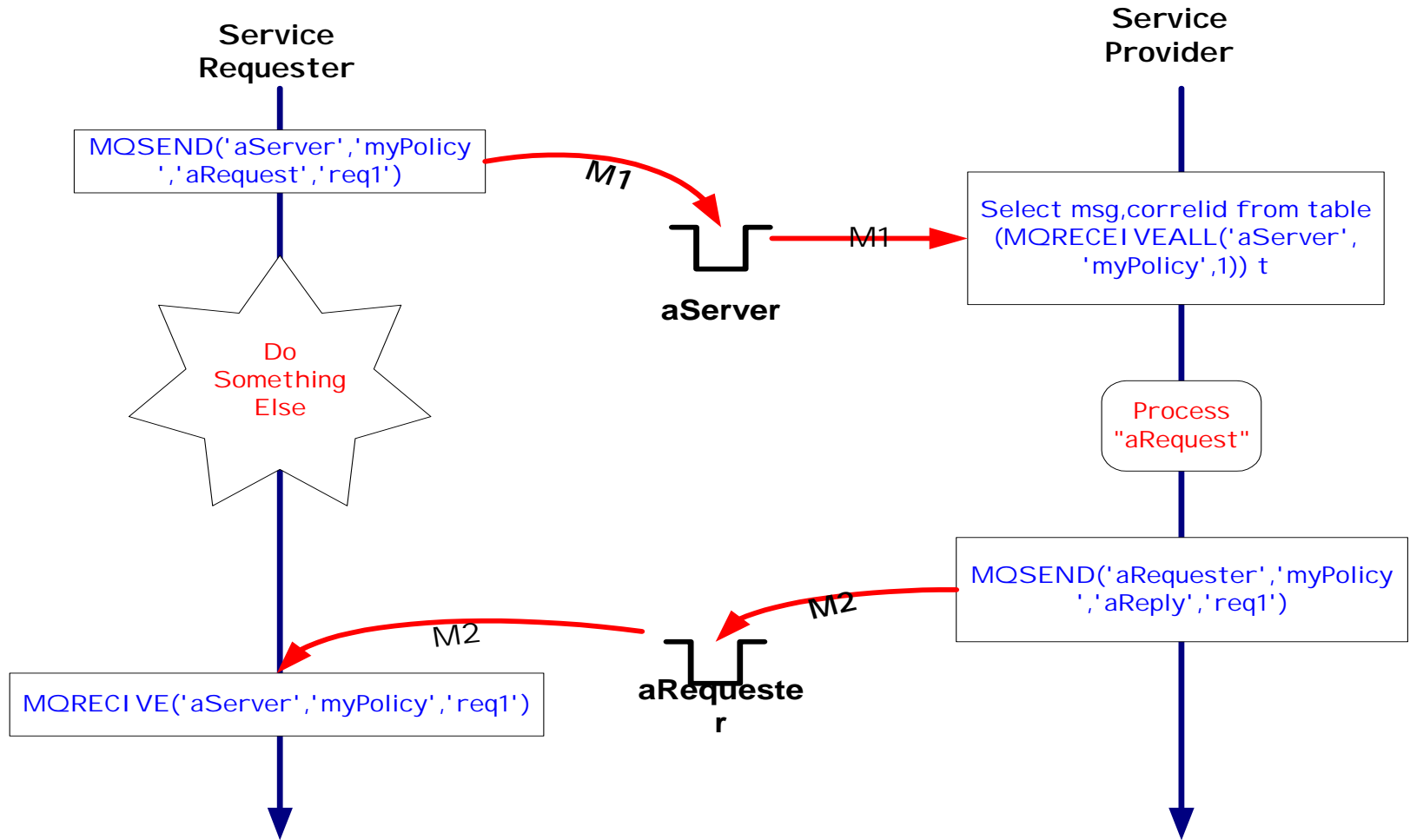


MQSeries Assistant

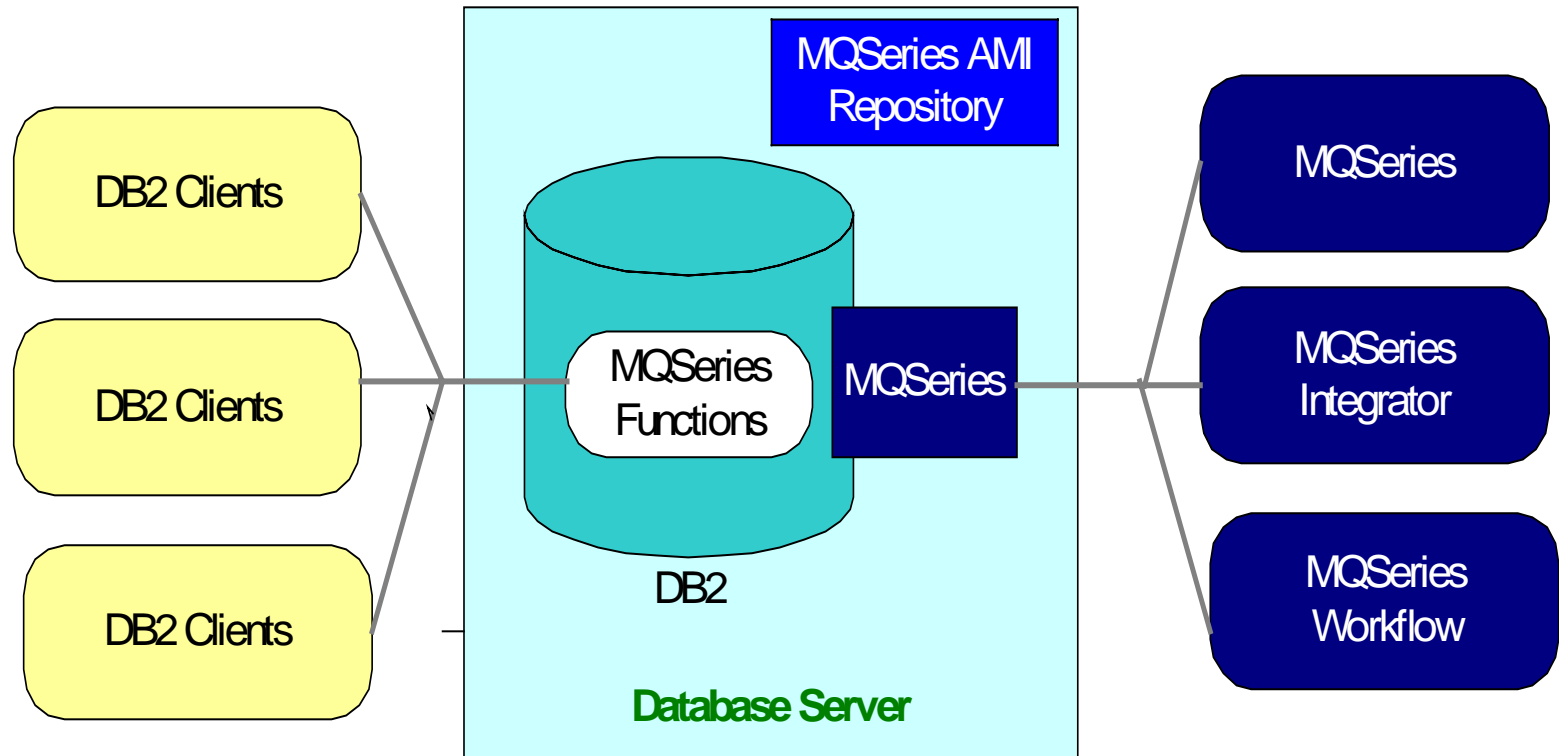
- MQ to DB2 Table Function & View wizard
- Allows mapping from DB2 tables to/from MQ message queues
 - supports mapping of positional or delimited message fields
- Launched from Stored Procedure Builder & Warehouse Center



Basic Messaging – Request / Reply

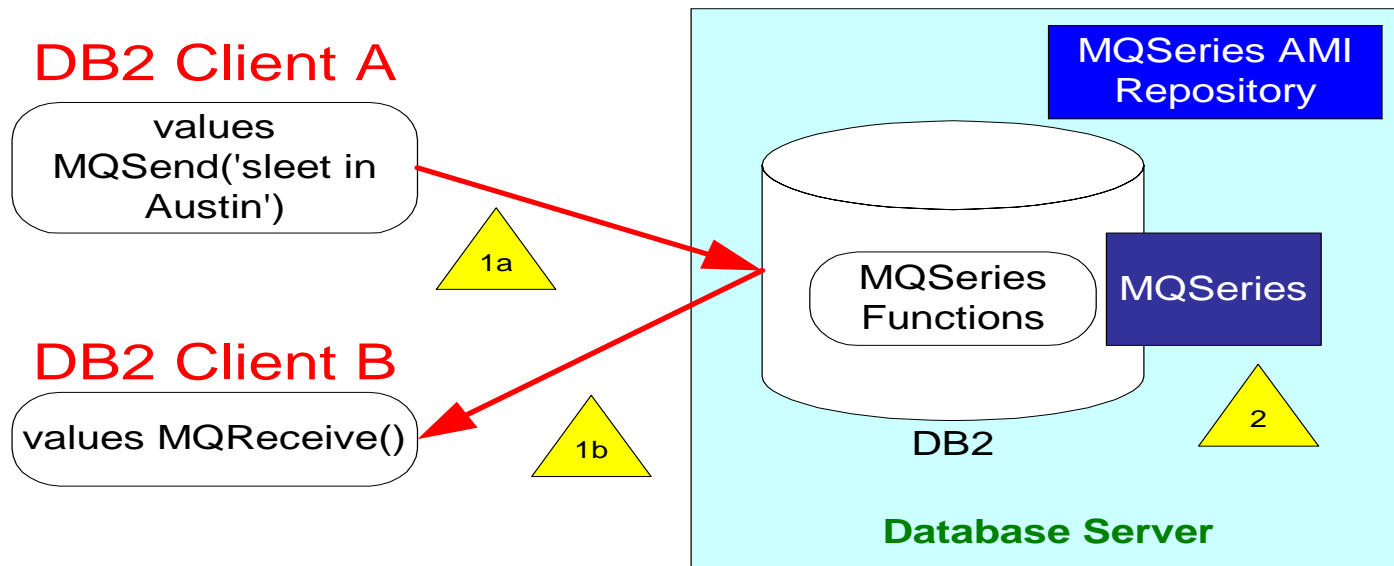


DB2 / MQ Series Integration – Basic Configuration



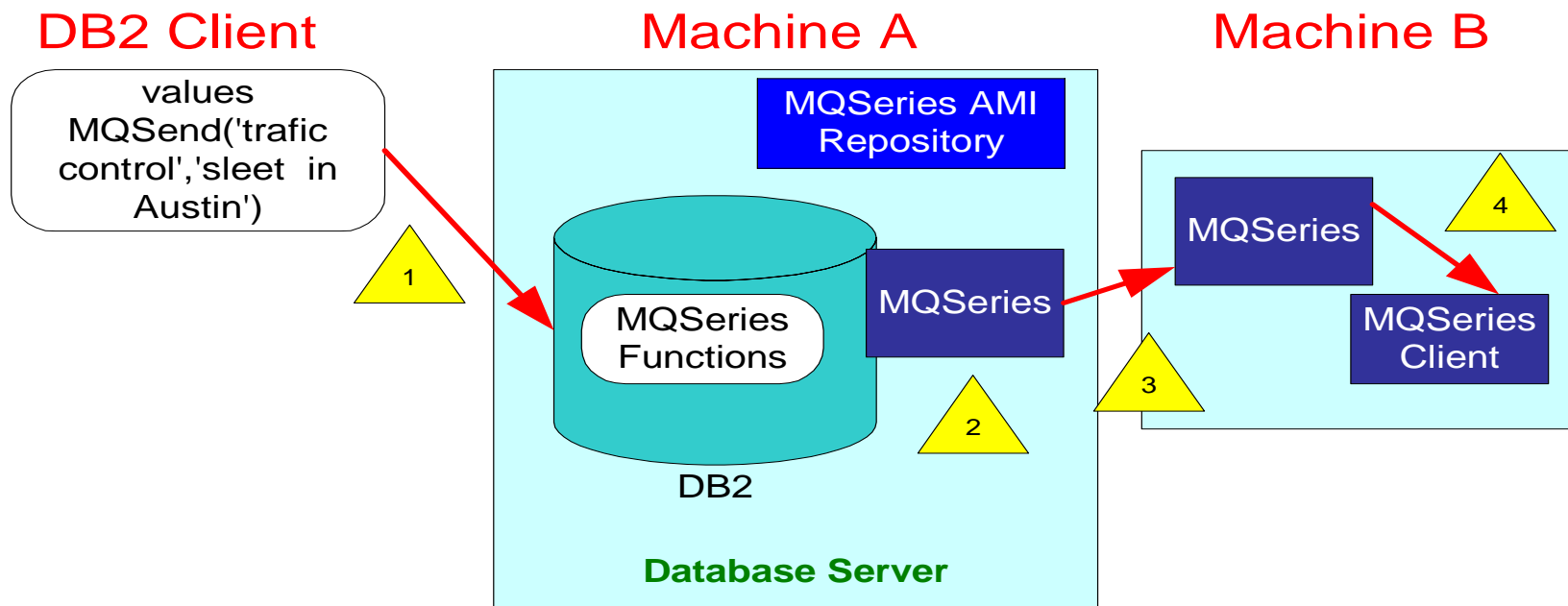
Basic Local Messaging – Scenario 1

- MQSeries server executes on the same machine as the DB2 Server
- DB2 clients may be local or remote.
 - standalone applications, WebSphere servlets or EJBs



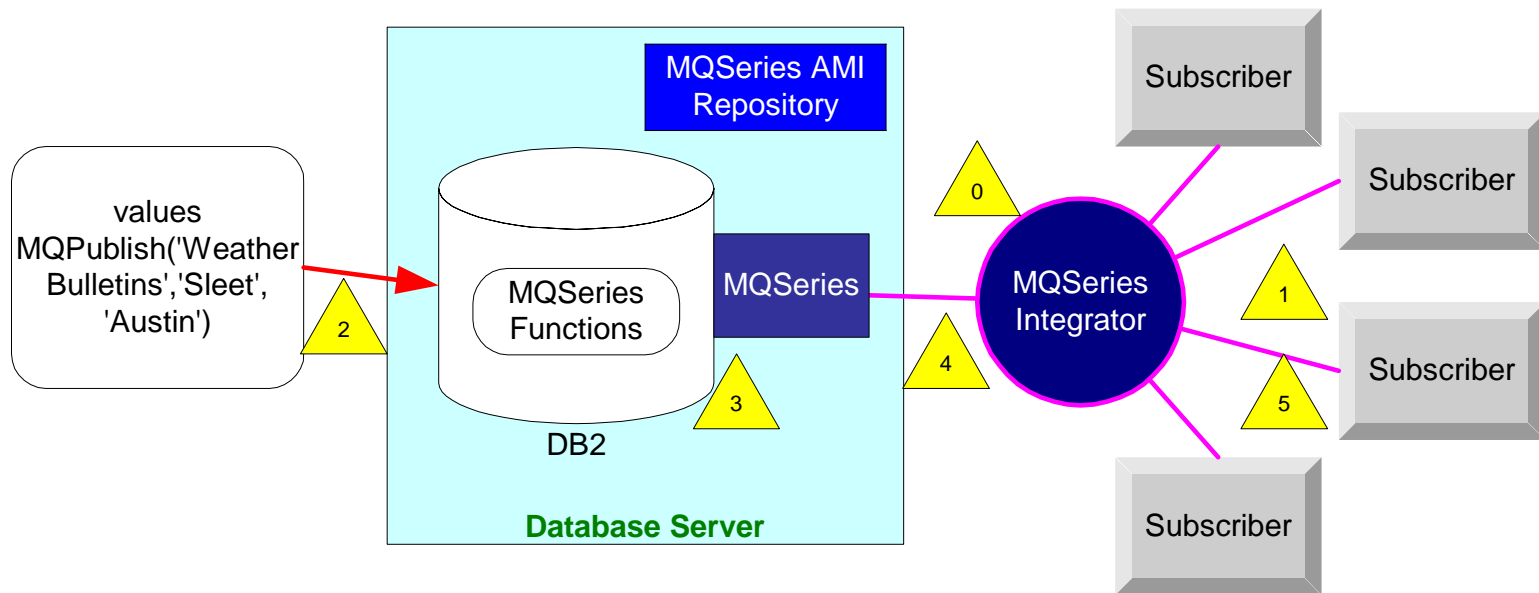
Basic Remote Messaging

- MQSeries Server A executes on the same machine as the DB2 Server
- MQSeries Server B on a remote server
- DB2 clients may be local or remote.



Publish/Subscribe

- Supports both MQSeries publish/subscribe and MQSI V2
- Publication may be on-demand or automated through triggers
- Subscribers can register one or more topics



Basic Messaging

- Sending Messages:
 - MQSend – send a message
 - MQSend(msg)
 - MQSend(service, msg)
 - MQSend(service, policy, msg)
 - MQSend(service,policy, msg, correlid)
- Examples
 - values MQSend('a test message')
 - values MQSend('DB2.DEFAULT.SERVICE','DB2.DEFAULT.POLICY', 'a message')
 - values MQSend('DB2.DEFAULT.SERVICE','DB2.DEFAULT.POLICY', 'a message','correlid1')
 - select MQSend(lastname) from employee
 - select MQSend(lastname || ' ' || firstnme) from employee
 - select MQSend(lastname) from employee where deptno=20
 - select MQSend(e.lastname || ' ' || d.manager) from employee e, dept d where e.deptno = d.deptno

Basic Messaging

- Getting messages - scalar functions
 - **MQRead, MQReadCLOB** - non-destructive read
 - **MQReceive, MQReceiveCLOB** - destructive read
 - msg = MQRead()
 - msg = MQReceive()
 - msg = MQRead(receive-service)
 - msg = MQRead(receive-service, service-policy)
 - msg = MQReceive(receive-service, service-policy, correl-id)
- Examples
 - values MQRead()
 - values MQReceive('myService','myPolicy', 'correlid1')
 - **insert into MESSAGE_ARCHIVE(time, msg) values (current time, MQRead())**

Basic Messaging

- Getting messages - table functions
 - **MQReadAll, MQReadAllCLOB** - non-destructive read
 - **MQReceiveAll, MQReceiveAllCLOB** - destructive read
 - MQReadAll()
 - MQReadAll(receive-service)
 - MQReadAll(receive-service, policy)
 - MQReceiveAll(num-rows)
 - MQReceiveAll(receive-service, policy, correlid, num-rows)
- Returns
 - msg - Varchar(4000) - contents of the MQSeries message
 - Correlid - Varchar(24) - correlation ID used to relate messages
 - Topic - Varchar(40) - topic that the message was published with
 - QNAME - Varchar(48) - queue name where the message was received
 - MSGID - Char(24) - MQSeries unique identifier for this message
 - MSGFORMAT - Varchar(8) - format of the message, as defined by MQSeries



Basic Messaging

- Table Function Examples

- select * from table (MQReadAll()) t
- select msg from table (MQReadAll()) t
- select varchar(msg,20) from table (MQReceiveAll(10)) t
- select count(*) from table (MQReadAll()) t
- select count(*) from table (MQReceiveAll()) t
- select * from table (MQReadAll(10)) t where t.CorrelID='AB'
- select * from table (MQReceiveAll('myService','myPolicy','AB',10)) t
- select t.msg, e.age from employee e, table(MQReadAll()) t where t.msg = e.lastname
- insert into msg_history values (select * from table (MQReadAll()) t)
- create view M as (select msg from table (MQReadAll()) t)

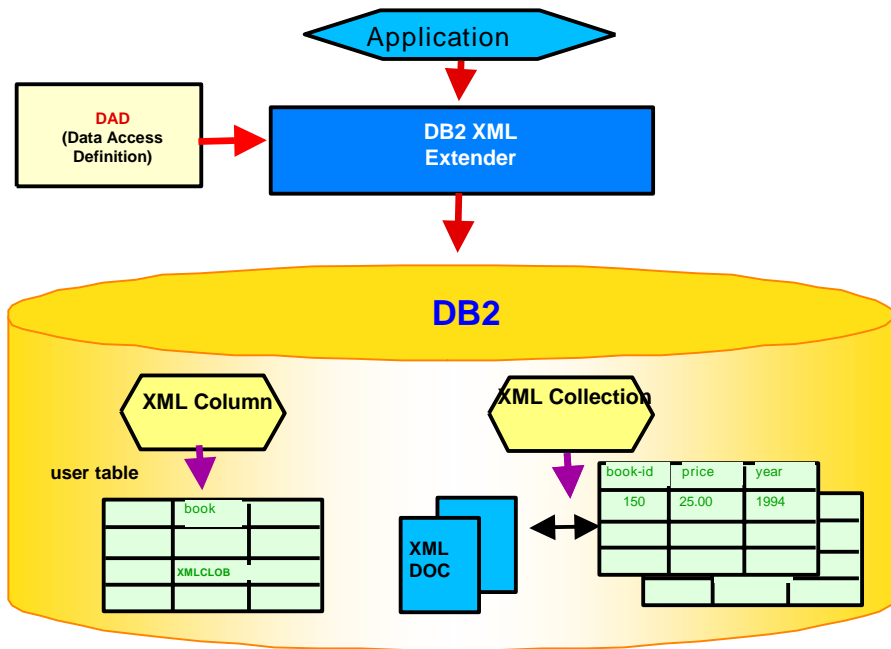
Publish / Subscribe

- MQPublish
 - MQPublish (msg)
 - MQPublish (publisher-service, msg)
 - MQPublish (msg, topic)
 - MQPublish (publisher-service, service-policy, msg, topic, correlid)
- Examples
 - values MQPublish('my favorite stock quote')
 - values MQPublish('200','IBM:MyPortfolio')
 - **CREATE TRIGGER new_employee AFTER INSERT ON employee
REFERENCING NEW AS n FOR EACH ROW MODE DB2SQL
VALUES MQPublish('HR_INFO_PUB', current date || ' ' || lastname || ' '
|| DEPARTMENT,'NEW_EMP')**

Publish/Subscribe

- MQSubsubscribe
 - MQSubscribe(topic)
 - MQSubscribe(subscriber-service, policy, topic)
- MQUnSubscribe
 - MQUnSubscribe(topic)
 - MQUnSubscribe(subscriber-service, policy, topic)
- Examples
 - values MQSubscribe('myPortfolio')
 - values MQSubscribe('myPortfolio:IBM')
 - values MQSubscribe('emp_subscription','NEW_EMP')
 - values MQUnsubscribe('myPortfolio')

DB2 XML Extender



- Store/retrieve whole XML documents
 - As XML column (entire document) or collection of fields
 - As externally managed files using DB2 UDB Data Links Manager
- Compose or decompose and store/retrieve portions
 - stored procedures `dxxGenXML/dxxShredXML` for collection of fields
- Document Access Definition (DAD)
 - Shape
 - Scope
- XML data types
 - XMLVARCHAR, XMLCLOB, XMLFILE (file name)
- Search: fast and powerful search/indexing on XML

XML Extender



- DTD Repository
 - Store your DTDs in DB2
 - Reference one DTD from
 - Many XML documents
 - More than one XML column and XML collection
 - Use DTDID to validate input XML documents at insertion
- Validating Input or Generated XML Documents
 - Validation specified in the DAD, supported in both XML Column and Collection features
- GUI based Administration Wizard
 - Enable your database, tables, column, collection for XML
 - Create DAD

DB2 XML Column for MQSeries

- MQReadXML (or MQReadXMLCLOB)
 - Places an XML message at the head of an MQSeries queue, without removing it from the queue, in an XMLVarchar (or an XMLCLOB) row
- MQReadXMLAll (or MQReadAllXMLCLOB)
 - Returns a DB2 table containing XML message data without removing messages from the queue. The messages are placed in an XMLVarchar column (or an XMLCLOB column)
- MQReceiveXML (or MQReceiveXMLCLOB)
 - Removes an XML message at the head of an MQSeries queue, and places it in an XMLVarchar (or an XMLCLOB) row
- MQReceiveAllXML (or MQReadAllXMLCLOB)
 - Removes XML messages from an MQSeries queue, and places them in an XMLVarchar column (or an XMLCLOB column)



DB2 XML Column for MQSeries

- Enqueuing Functions
 - MQSendXML, MQSendXMLFile
 - MQPublishXML
- Examples
 - `select MQSendXML(myFavoriteXMLColumn)`
`from aTable`
 - values MQReadXML()
 - values
MQSendXMLFile('c:\xml\myFavoriteXMLFile')

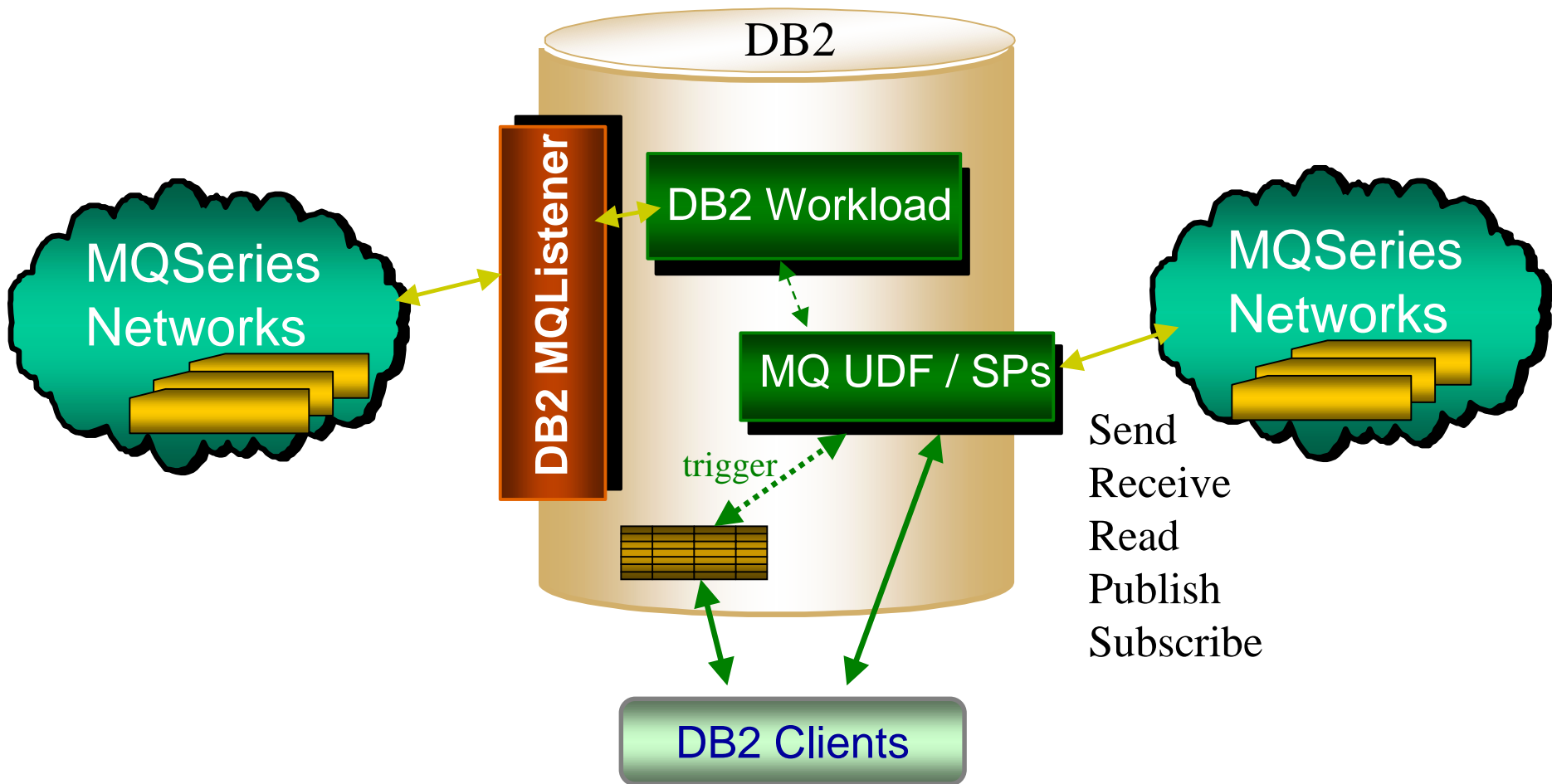
Handling XML Messages

- XML - MQSeries Stored Procedures
 - dxxmqShred - decompose a single XML message
 - dxxmqShredAll - decompose all XML messages in a queue
 - dxxmqGen - generate XML messages from existing tables
 - dxxmqRetrieve - generate XML messages from existing tables using a collection
 - dxxmqInsert - decompose a single XML message using a collection
 - dxxmqInsertAll - decompose all XML messages using a collection

Handling XML Messages

- DB2 Stored Procedures may be invoked from most languages and environments including:
 - SQL, C, C++, Java, Cobol, ...
 - Data Access Beans (servlets, JSPs, session beans, ...)
 - Session Beans (VAJ 3.5.3, WebSphere Studio Advanced V5 Beta)
 - SOAP (WebSphere 4.0, WebSphere Studio Advanced)
 - MQSI (must currently handcraft a SQL node)

DB2 / MQ Integration Futures



Platform Support Matrix

Description	DB2 Win/Unix	DB2 390
<p>Core MQ UDFs: (MQSend, MQReceive, MQRead, MQReceive CLOB, MQReadCLOB, MQReceiveAll, MQReadAll, MQRecieveAllCLOB, MQReadAllCLOB)</p>	<p>V7.2: Win, AIX, Solaris, HP Coming : Linux, transactional</p>	<p>V7 APAR: PQ59549 Both 1 and 2 phase commit</p>
<p>Extended MQ UDFs: (MQPublish, MQSubscribe, MQUnsubscribe)</p>	<p>V7.2: Win, AIX, Solaris, HP Coming: Linux, transactional</p>	<p>coming</p>
<p>XML MQ UDFs and SPs</p>	<p>V7.2: Win, AIX, Solaris, HP Coming, Linux, transactional</p>	<p>Coming</p>
<p>DB2- MQ Listener</p>	<p>Early beta Win/AIX</p>	<p>Phase 1: Win/Unix listener to 390 - Coming</p>



Summary

- Simple MQ/DB2 Integration available now
 - More function/platforms rolling out
- Many Usage Scenarios
 - Selective publication of data
 - Automated publication of data
 - Application to application Connectivity
 - Data subscriptions

Where to get more information

- DB2 Developers Domain www.ibm.com/software/data/developer/
 - DB2 MQ XML Functions: Using MQSeries and XML Extender from DB2 Applications
 - Using the MQSeries(R) Assist Wizard
 - Using MQSeries from DB2 Applications
- Redbooks
 - SG24-6282-00 - MQSeries Publish/Subscribe Applications
 - SG24-6513-00 - Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager