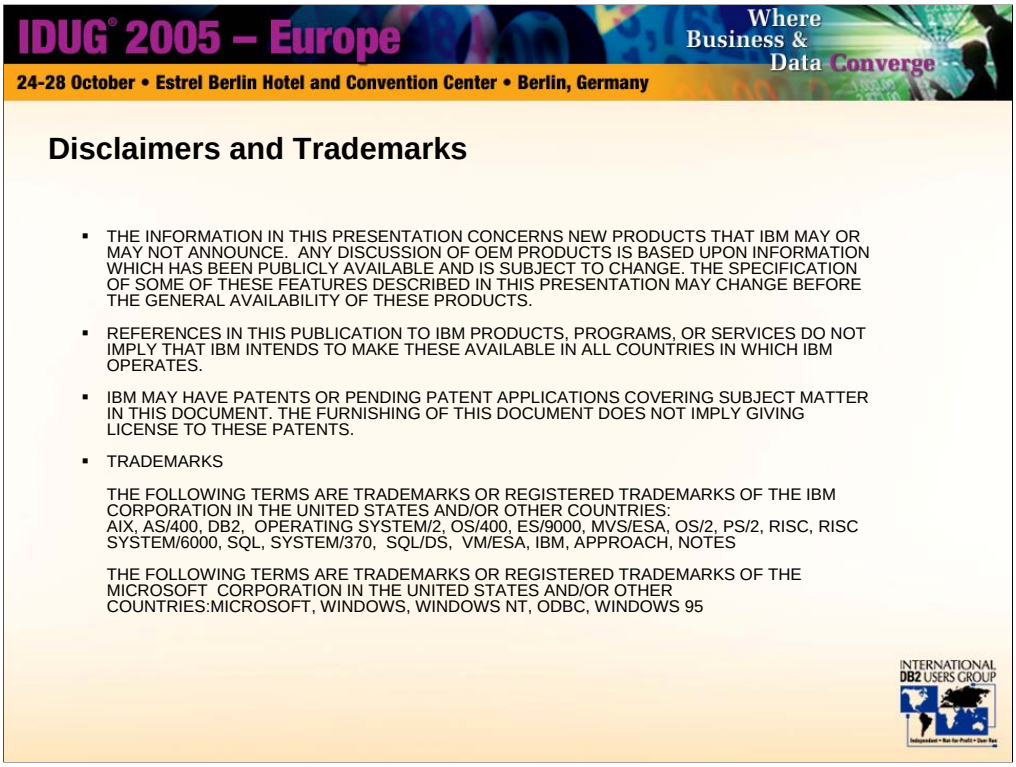
IDUG® 2005 – Europe
Where Business & Data Converge
24-28 October • Estrel Berlin Hotel and Convention Center • Berlin, Germany

Platform: Beyond DB2

WebSphere II Q Replication - It's Here! What is it?

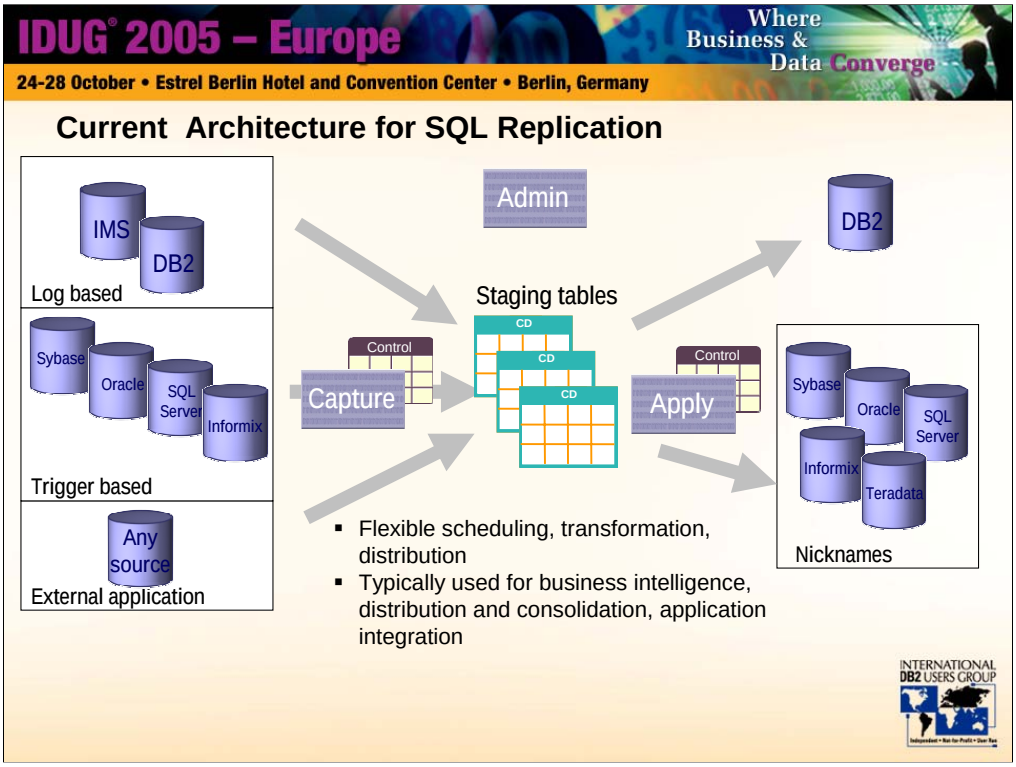
Madhu Kochar
Sr. Development Manager, II Replication/Classic Federation/Event Publishing

Session: E02
October 24, 2005 1:30-2:30



Some sneak preview information may be provided. All standard disclaimers apply….

This presentation is an introduction to the new queue based replication architecture which was made available in September 2004 as part of DB2 Information Integrator V8.2 (note that we have recently been rebranded to the name WebSphere Information Integrator) . Topics include the motivation for this new architecture and a lot of the technical details regarding how it works, what options will be available, and examples of applications of the technology.



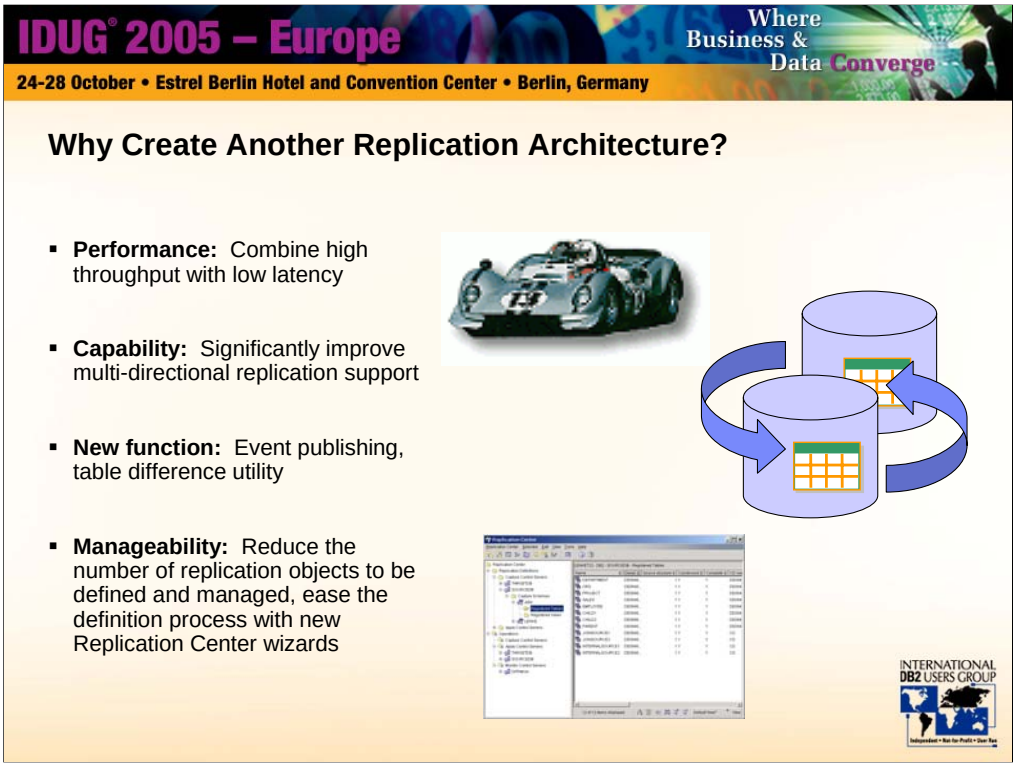
•Sold under the name DB2 Datapropagator, as part of DB2/Websphere Information Integrator, or as part of DB2 for LUW, this is the architecture that has been available for more than 10 years. We are now calling this SQL Replication to make it distinct from Q Replication (the new queue based architecture)

•A Capture program or trigger captures changes and moves them into a staging table, called a changed data (CD) table.

- •A single staging table can serve as source for multiple subscriptions or multiple staging tables can be created for a single source depending on the application requirements.
- •The staging table typically resides on the same system as the source table
- •Staging table format is published to enable applications or ISV to provide capture function

•The Apply program fetches data from the staging tables using client/server db2 communications and applies it to the target tables using standard SQL statements

- •One or more apply programs can subscribe to a CD table
- •One apply program can replicate data to one or more target tables
- •Target tables can be user copies, history tables, or staging tables
- •Apply program handles column and row subsetting, performs SQL transformations, manages commit scope based on subscription sets and table vs transaction consistent delivery ➔ note that RI cannot be guaranteed for foreign sources as ordering across tables is unknown from trigger capture mechanisms
- •Apply program references foreign source and target tables and control tables via nicknames

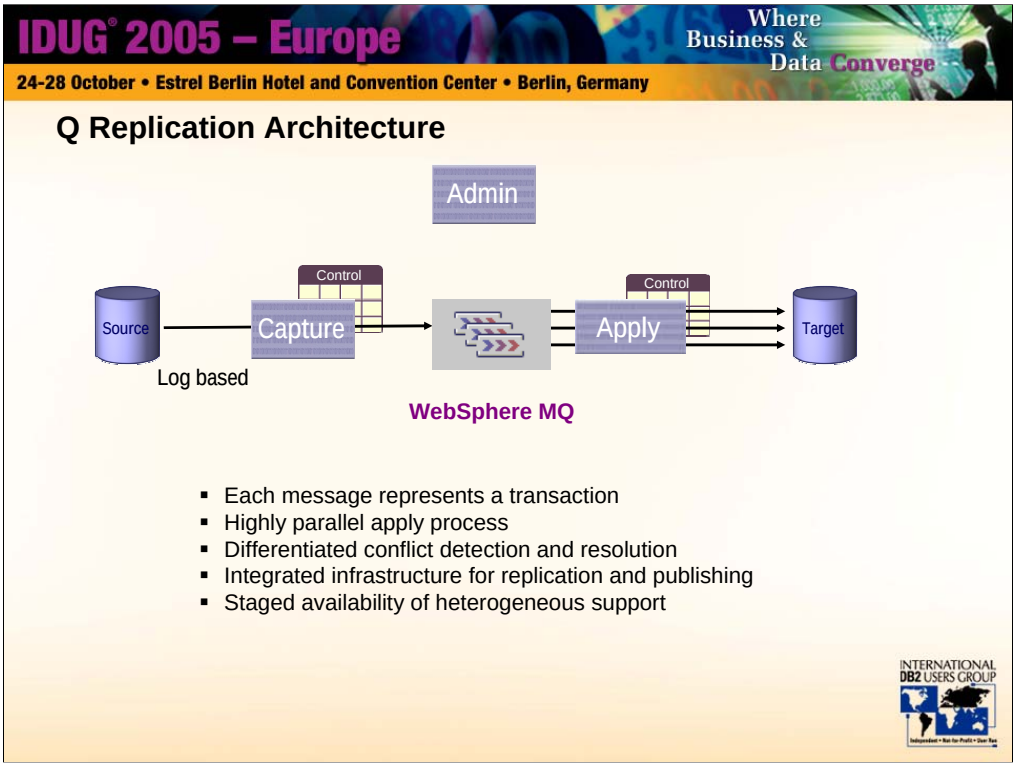


There is a growing demand for high speed low latency replication, primarily for the purposes of meeting high availability requirements. The implementations are varied, and include geographically distributed peer to peer applications, workload balancing, and primary/secondary failover configurations. In addition to speed, these implementations require robust methods for conflict detection and resolution.

We also see the need for a solution that is easy to manage - requiring reduction in the numbers of objects to create and manage, and with easier methods to create and manage those objects.

In creating this new architecture, we also see an opportunity to create an infrastructure that can serve application messaging/publishing in addition to replication.

Please note that this architecture supplements the existing SQL architecture, it does not replace it.



•Capture program stages data in queues

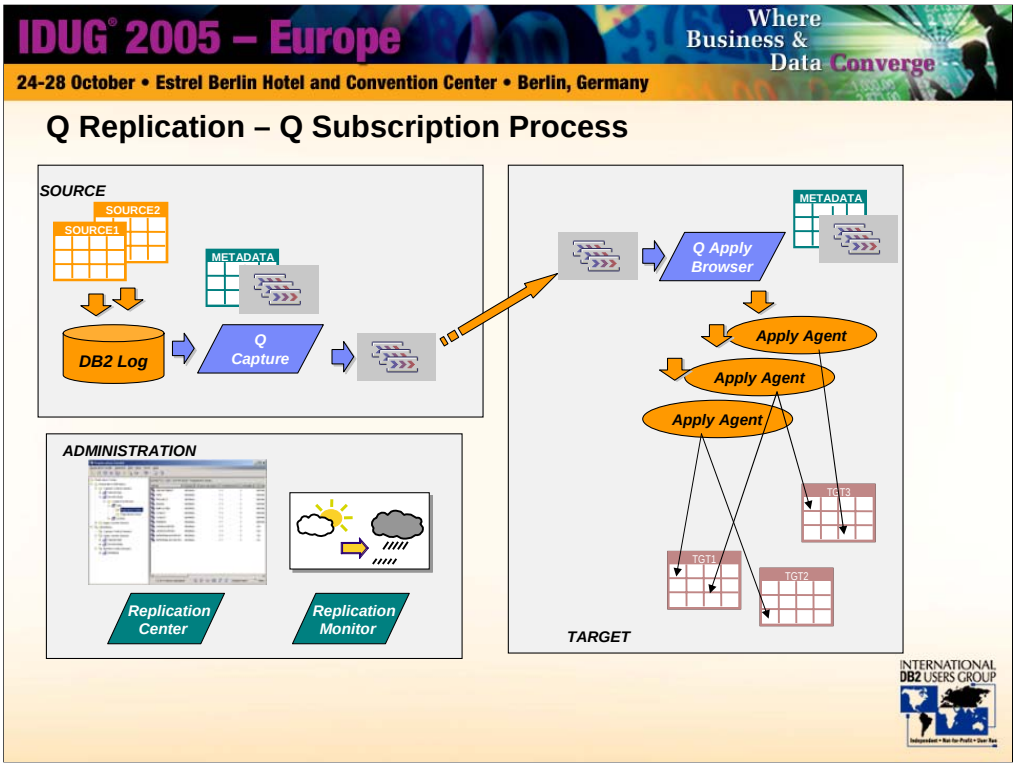
- •Each message represents a transaction
- •One or more data transport queues per source/target database pair

•Apply is significantly re-architected

- •Highly parallel in how it applies transactions to tables
- •Data is always applied per source transaction units
- •Data is applied such that source commit order is observed where necessary for data integrity

•Conflict detection very robust, including ability to handle deletes and key changes

•Data can also be published in XML format for external applications, using the same capture infrastructure



This animation takes you through the implementation details of Q replication.

(1) First you install the programs and set up infrastructure for queues and control table metadata

(2) Subscribe to those tables of interest

(3) Changes to those tables will appear on the DB2 recovery log

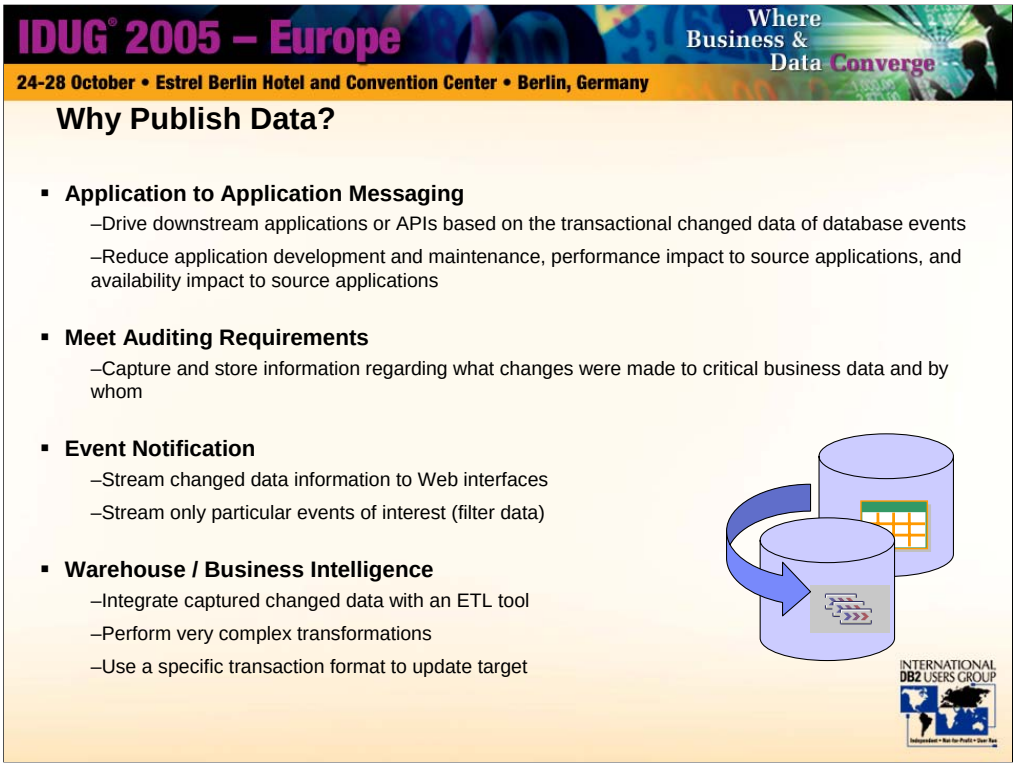
(4) The changes will be read by Q Capture and stored in memory

(5) Committed transactional data will be put to the data transport queue

(6) At a commit interval the data will be sent by MQ to the target receive queue

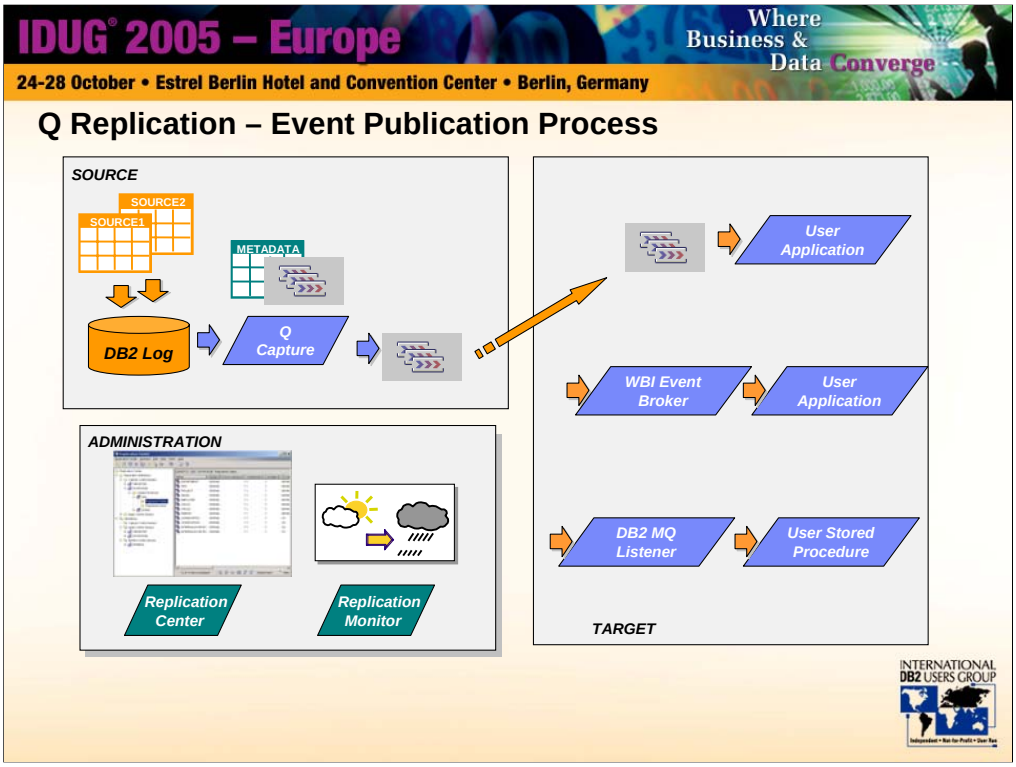
(7) Q Apply browser reads transactions from the queue, examines and tracks dependencies between transactions, and feeds transactions to Apply agents

(8) The alert monitor program keeps an eye on the both SQL and Q Replication server metadata and statistics.

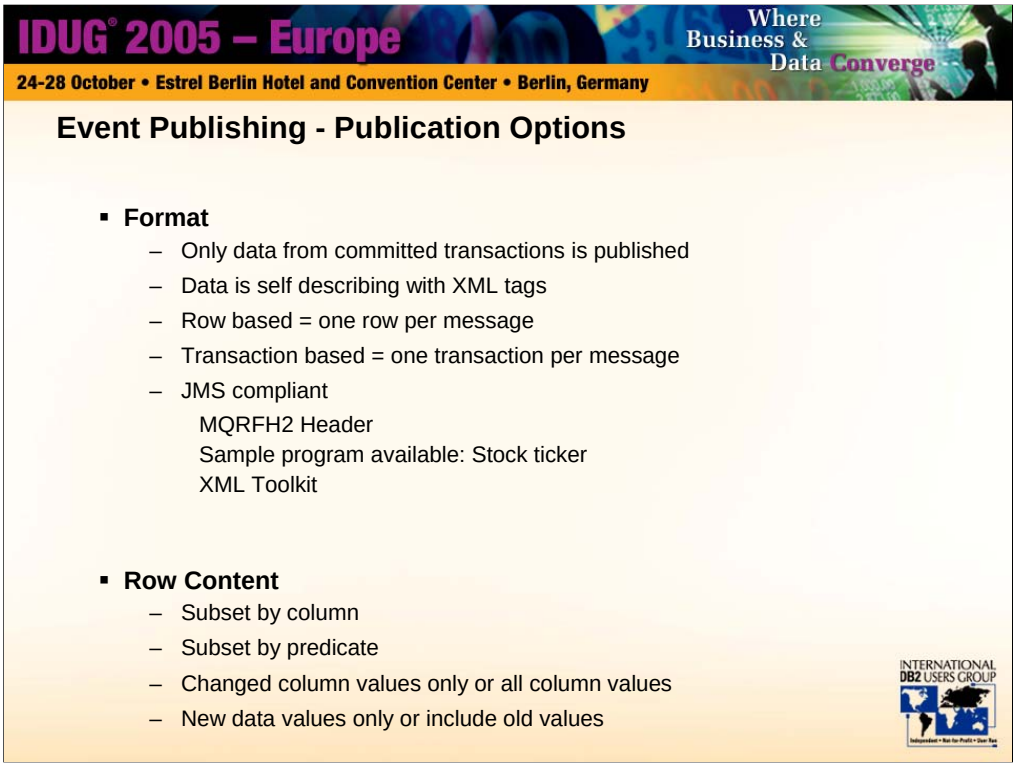


More and more customers are using message queuing to provide application to application communication. When the need exists to combine database activity with application messaging, then a strong advantage can be gained by using an asynchronous log based infrastructure to post messages that coordinate with database events. This eliminates the cost of 2 phase commit, or works in databases that cannot support a 2 phase commit. This also avoids availability concerns posed by an application that would otherwise require both the message queue server and the database to be available.

Database changes can be posted to a queue and then sent to downstream applications for further processing. Examples: streaming stock prices or wholesale item prices, moving data from an order database to shipping and/or billing databases, notifying all systems of customer information changes….

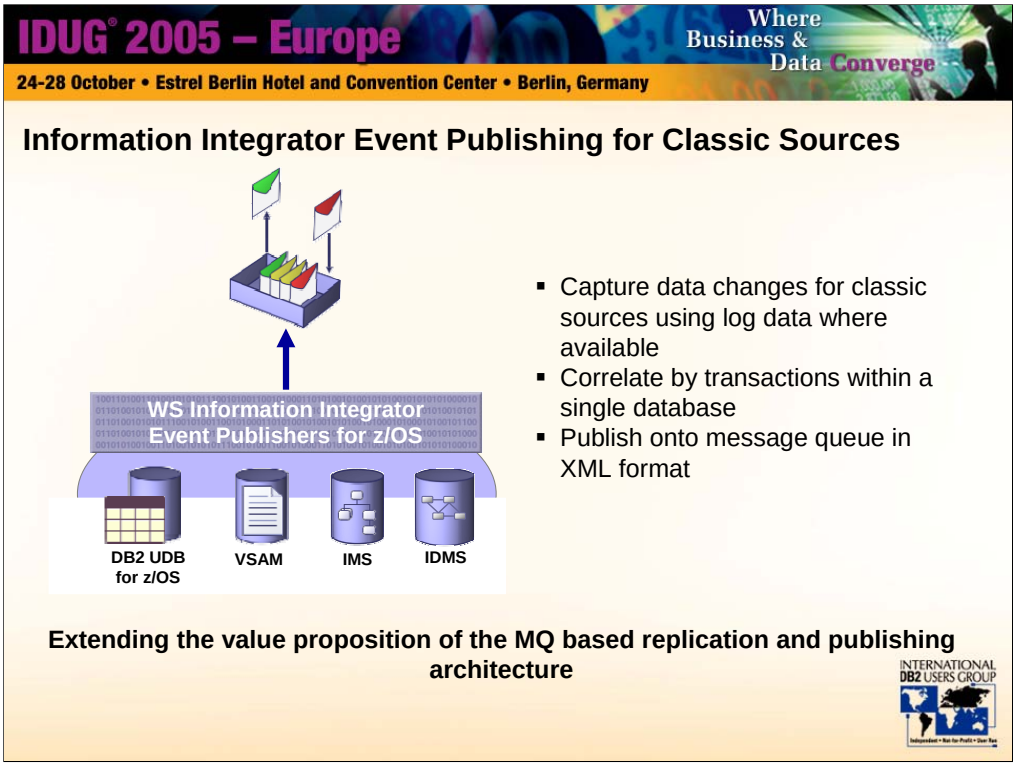


This animation is showing various configuration suggestions for event publishing. In addition to receiving published data directly from a user application, the data could first be brokered by the Websphere Business Integration Event Broker (formerly known as MQSI – MQ Series Integrator), and then passed on to other applications, or the data could be brokered by the MQ Listener function of DB2 on LUW or z/OS, and then passed on to your user written stored procedure.

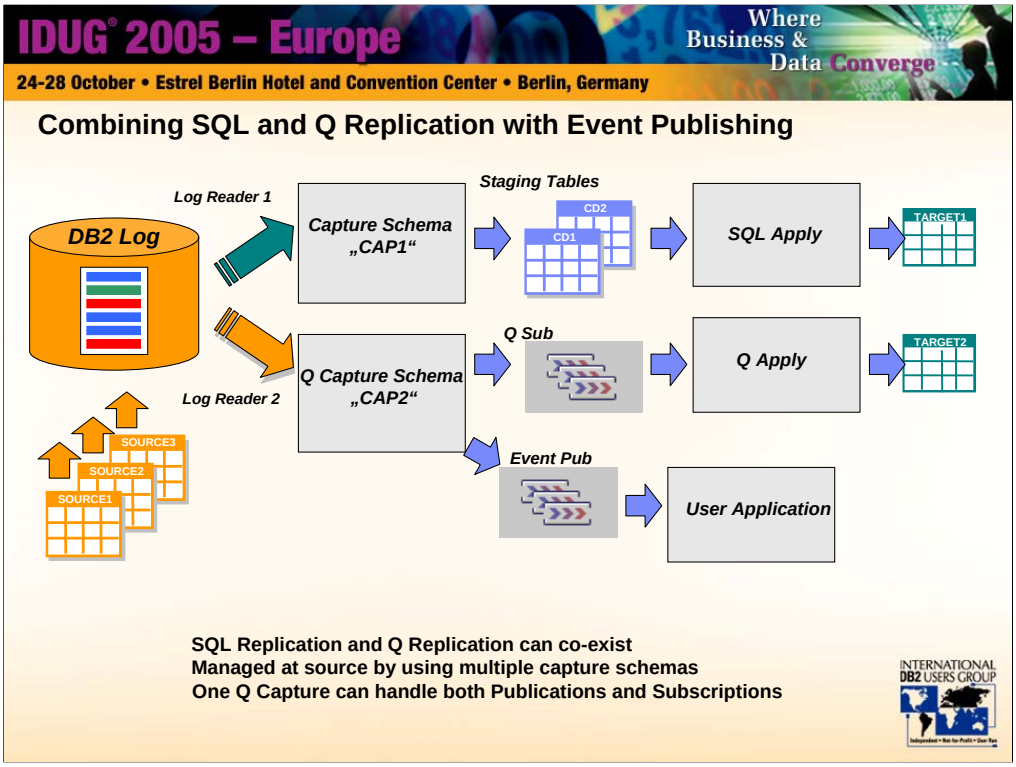


Data is captured in the same way that it is captured in Q Replication – transactional data is stored in memory until a commit record has been seen on the log. Then the committed data is translated into UTF-8, tagged with descriptive XML tags, and is put to a queue. You can choose for the messages to be made up of individual row changes, or of all associated row changes that were in a transaction.

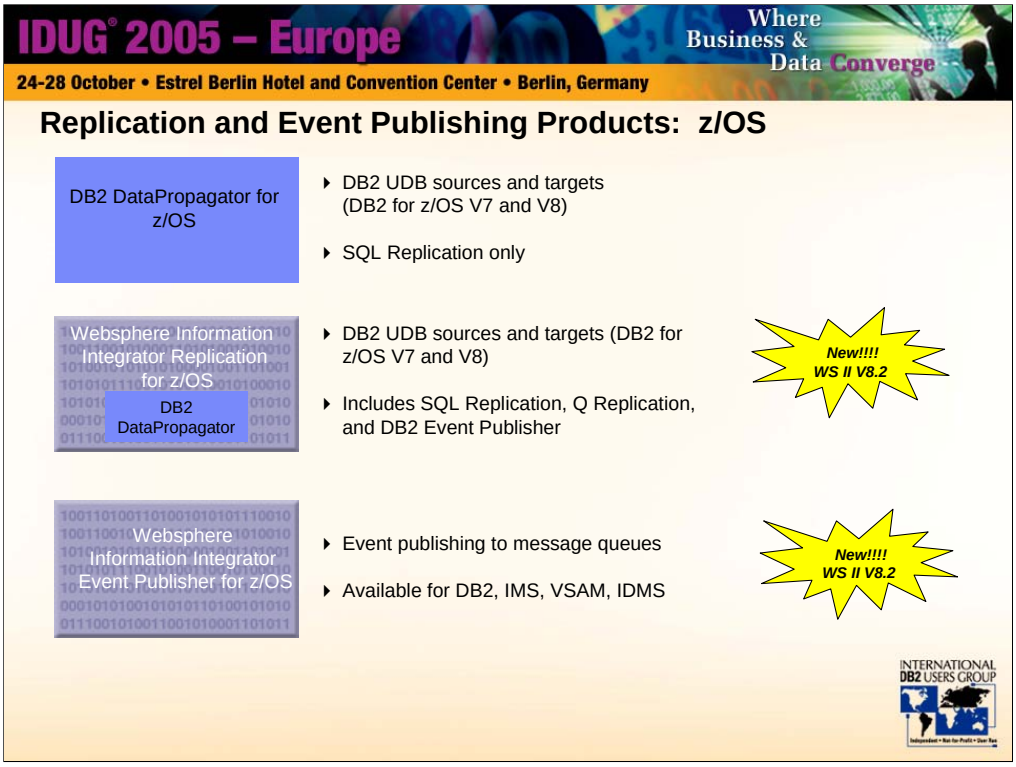
Additionally, you have multiple options that dictate how much data is put into each row operation published: new/old values, all columns or only changed columns, etc.



In addition to the event publisher for DB2, we are extending this technology by also offering event publishing from other classic data sources. The first to be offered will be event publishers for IMS and CICS VSAM. All publishers provide MQ messages in the same UTF 8 XML format. Most options that are offered for DB2 event publishing are also offered for the classic sources. The major differences are (a) the non relational data must first be mapped to a relational format through a mapping tool, and (b) the classic capture does not offer filtering at the logical "row" level – this must be performed at the application level.



This picture shows the relationships between SQL Replication, Q Replication, and Event Publishing. They can all co-exist. Q Replication and Event Publishing can be performed using the same Q Capture program (schema), but they do require separate queues. SQL Replication requires a separate Capture program (schema) – one Capture program cannot perform both SQL and Q Replication. One reason for this is that the Q Replication has been carefully designed to completely avoid 2 phase commit operations between DB2 and MQ.



DB2 Data Propagator for z/OS 8.2 is the newest release of the product that has been available for many years under the same name. There are a small number of new features available in this release – most significant is the addition of table reconciliation utilities.

The new queue based replication architecture is in all cases marketed and sold as part of the Websphere Information Integrator brand. When purchased on z/OS, this package includes the SQL replication, the queue based replication, and the event publisher for DB2 on z/OS. Note that on z/OS this package does NOT include Websphere MQ, which must be purchased separately. The prerequisite version is 5.3.

It is also possible to purchase individually the Event Publishers for DB2, IMS, or VSAM. These are also part of the Websphere Information Integrator brand of products.

Highly recommend z/OS 1.4 or later.

Can run with OS/390 2.10 or later

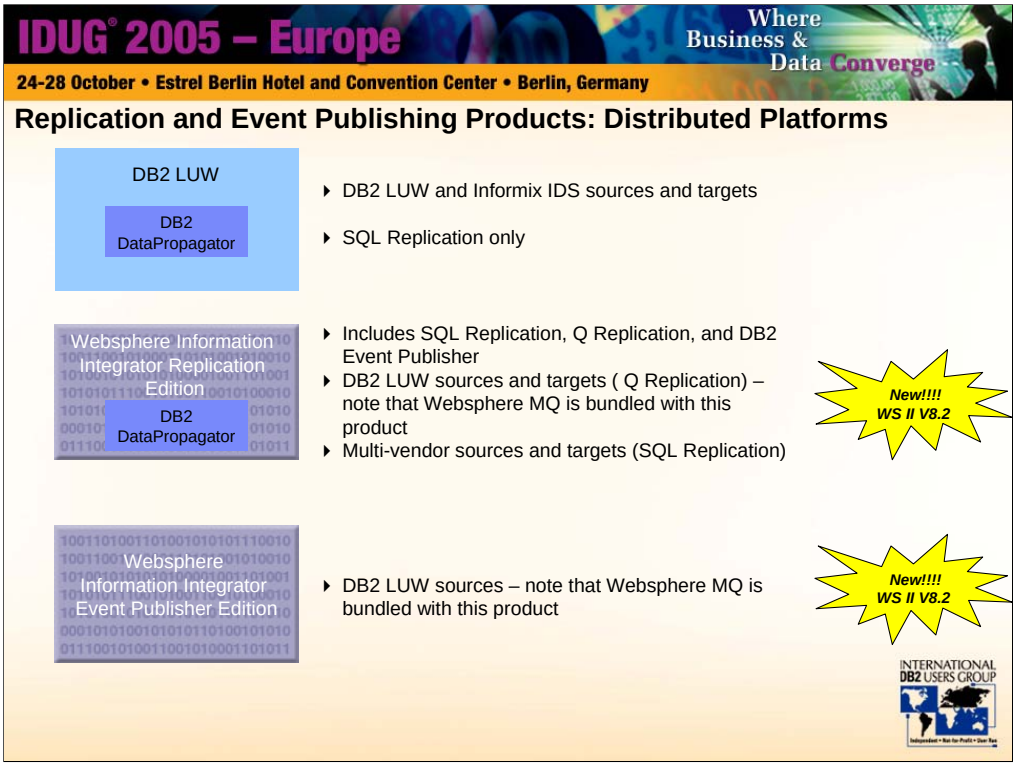
DB2 Universal Database for z/OS and OS/390 Version 7.1or later with PQ85495

WebSphere MQ for z/OS Version 5.3.1

XML Toolkit for z/OS and OS/390 Version 1.4.0 (for Q replication and Event publishing)

DB2 Administration Server (DAS) for z/OS (Replication Center)

390 Enablement (Replication Center)



There has been some confusion is this space and this slide attempts to help sort out this confusion. The newest release of DB2 is now available as V8.2, and had the code named "Stinger". There has been a lot of well deserved excitement about this release, and included in the buzz has been the new queue based replication. This new replication is actually marketed and sold as part of the Websphere Information Integrator brand. Q Replication leverages DB2 V8.2. Q Replication only works with an 8.2 release of DB2 LUW.

There are a number of WS II packages that include replication(and event publishing) – it is available in the standard, advanced, and replication editions. This package includes the SQL replication, the queue based replication, and the event publisher for DB2 LUW. Note that all of these multiplatform packages DO include Websphere MQ 5.3 in the bundle.

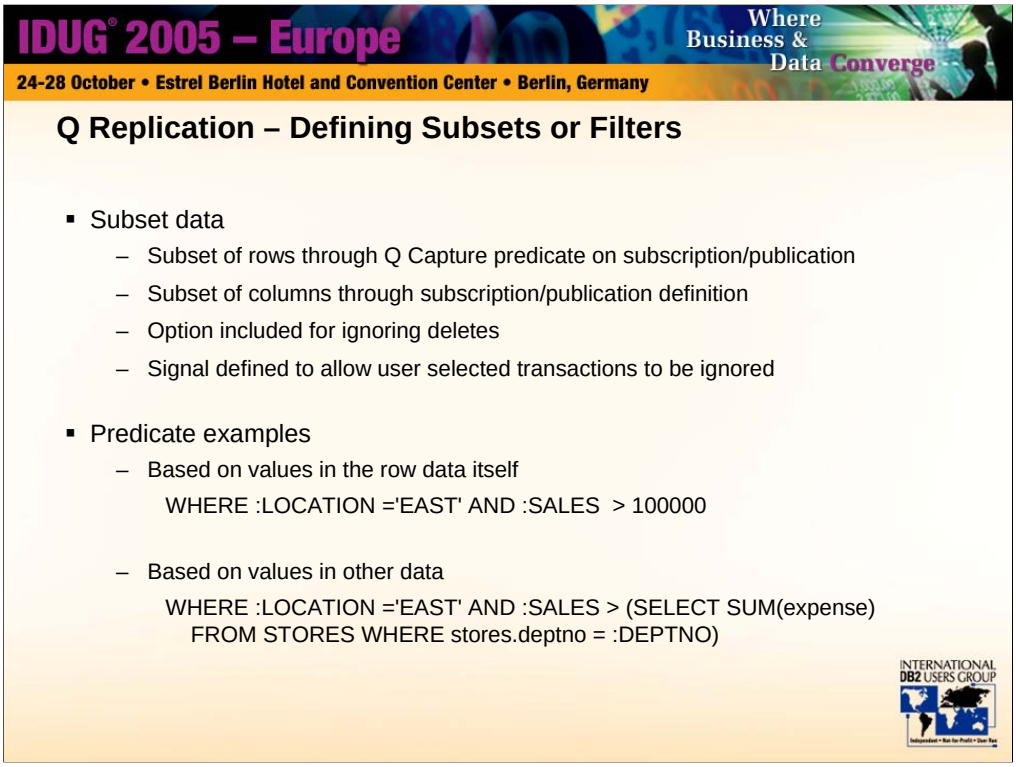
It is also possible to purchase individually the Event Publisher for DB2 LUW – this is the WS Information Integrator Event Publisher Edition.

DB2 Universal Database for LUW V8.2

Q Replication inherits the prerequisites of the database.

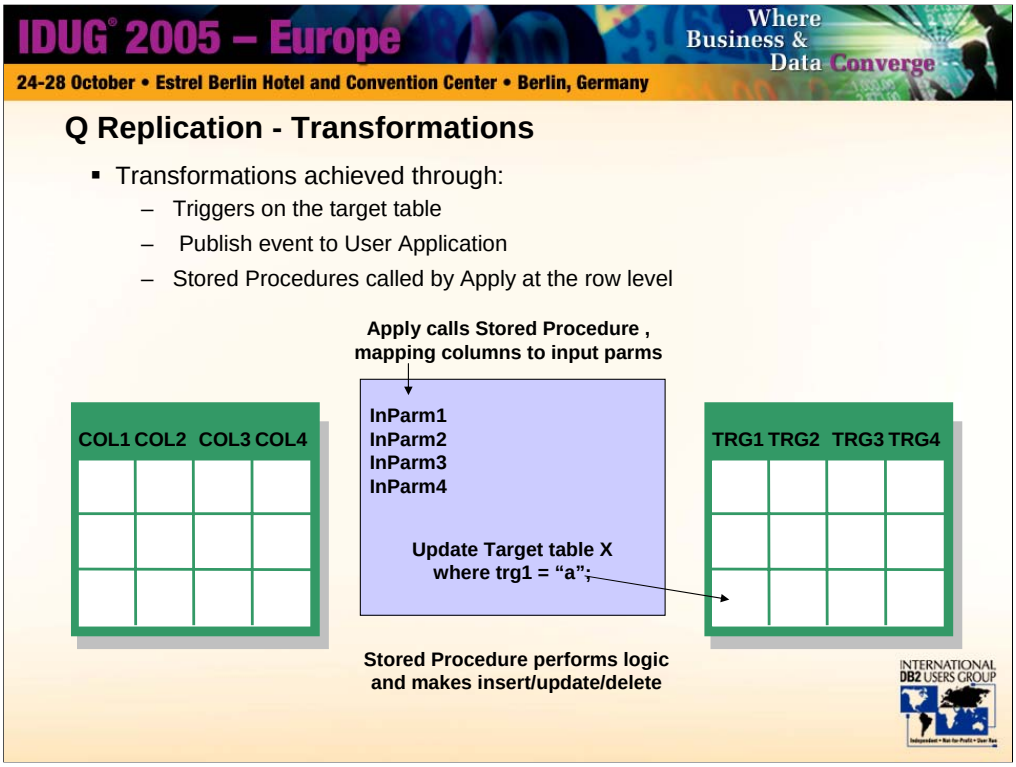
WebSphere MQ Version 5.3.1 is included in the WebSphere Information Integrator Package

MQ Server does not run on all 64 bit platforms



Column and row filtering is provided. The predicate is evaluated on the Q Capture side versus evaluation during the Apply process in SQL Replication. This allows the Q Capture to understand when a predicate column value has changed, and selectively convert certain updates to deletes or inserts as necessary, automatically. An option is available to suppress the replication of any deletes, and another option is available to mark transactions so that they will not be replicated.

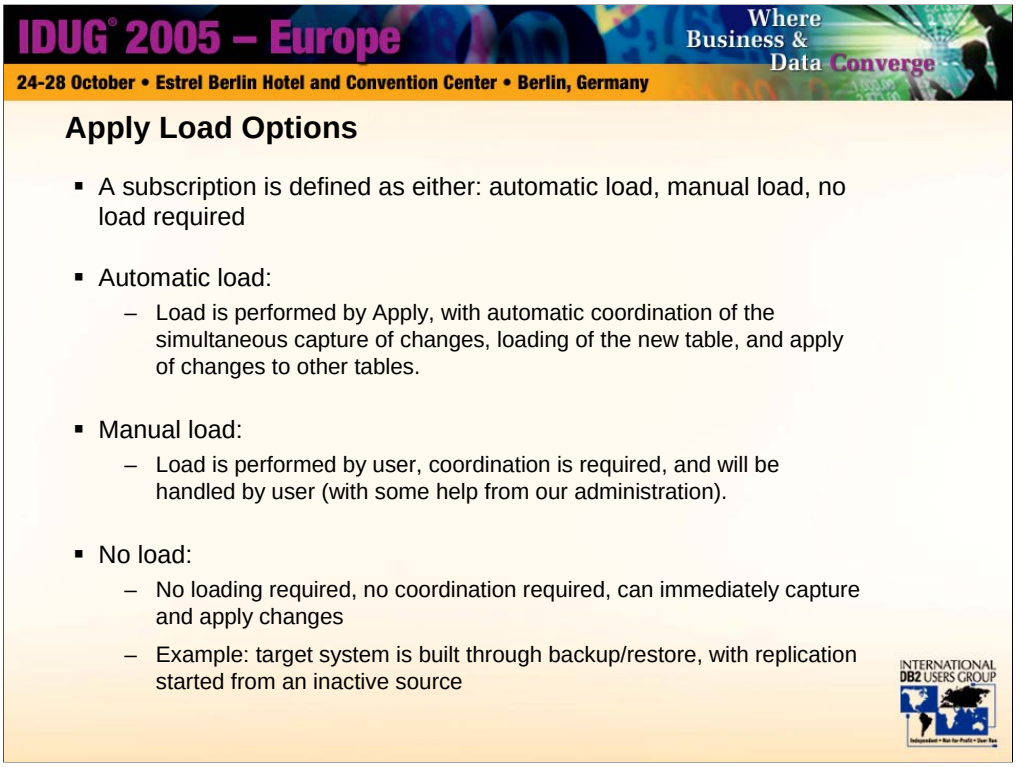
Evaluation can be made of the data in the row itself, and this is relatively fast. It can also include lookups in other tables, but this can dramatically affect performance.



One of the big benefits of SQL Replication is that transformations can be made very easily using SQL expressions. This capability is clearly different in the Q Replication case, where SQL is not being used. In Q Replication we provide a transformation exit capability in the way of a stored procedure call. The column values are passed in as the input parameters to the stored procedure, and the actual update of the table is made by the stored procedure, after any manipulations have been performed.

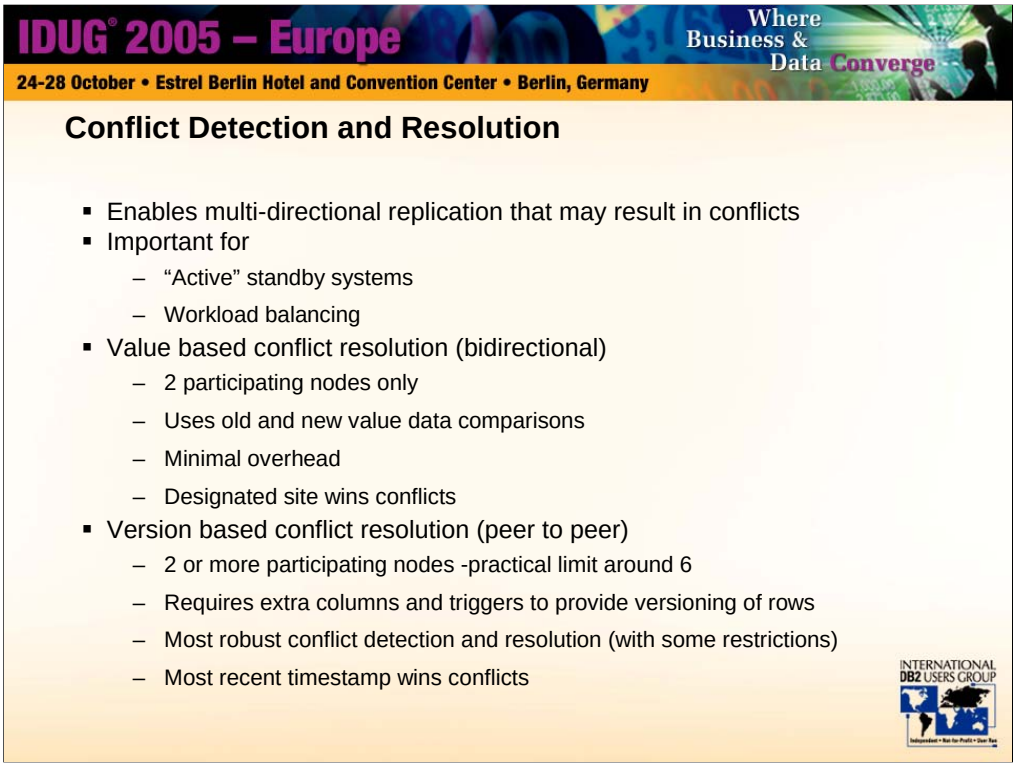
As previously described, the Event Publishing capability can be used to perform more extensive transformations.

Also, triggers and/or user defined functions could be defined on the target table to perform basic manipulations.



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”.

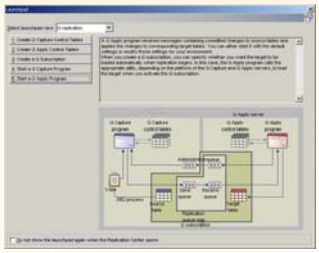


When multiple databases are allowed to update exactly the same records at exactly the same time, conflict detection and resolution must be incredibly robust. Here we recommend a version based method that can detect conflicts of all types. However, any robust method will require the addition of extra columns in the involved tables, and a mechanism (we offer triggers) to maintain these versioning columns.

When databases and applications are carefully set up such that only one copy is ever to be updated at a time, then it might be possible to live without conflict detection and resolution entirely. Or, it might be that one database will be a backup to the other, and backup scenarios may require conflict detection and resolution during switch and switchback timeframes. For this case, the peer to peer version based method can be used, but we also provide an alternate method, consuming less overhead, that may be entirely adequate for the application.

Replication Administration

- Replication Center GUI
  - –Launchpads, Wizards, Online Help
  - –Definitions, Operations, Monitoring



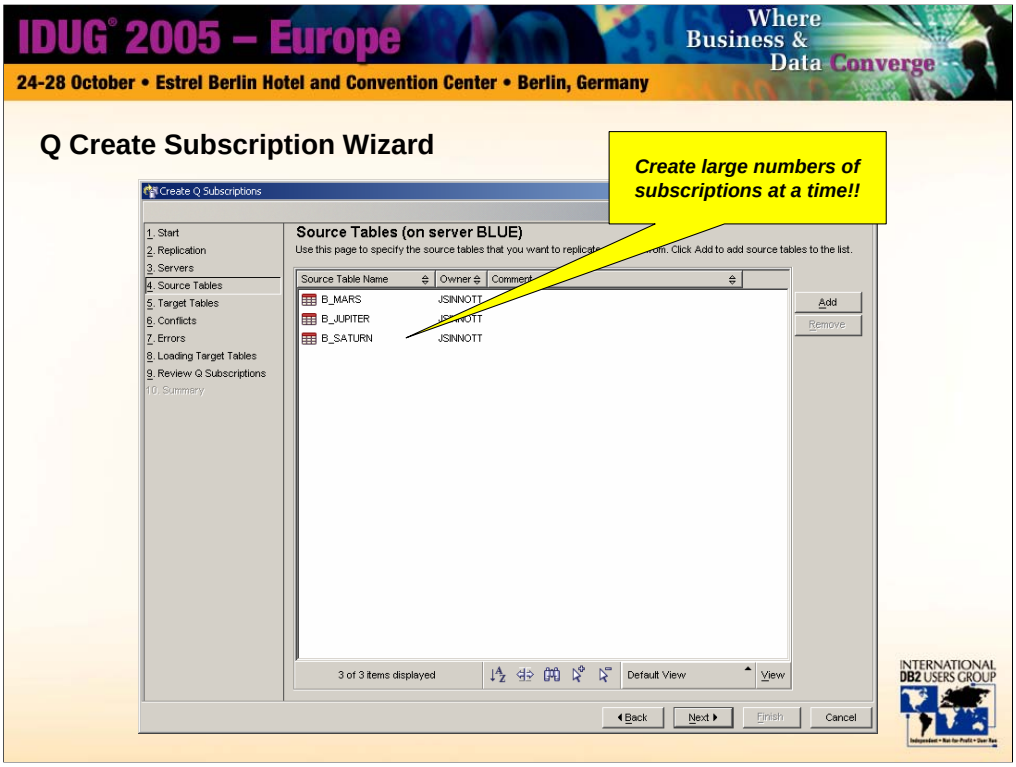
- Command Line Interface
  - –Scripts or interactive mode
  - –Example:

```
C:\asnclp
REPL > CREATE QSUB USING REPLQMAP ...
REPL > CREATE SUBSCRIPTION SET SETNAME ...
REPL > CREATE MEMBER IN SETNAME ...
```

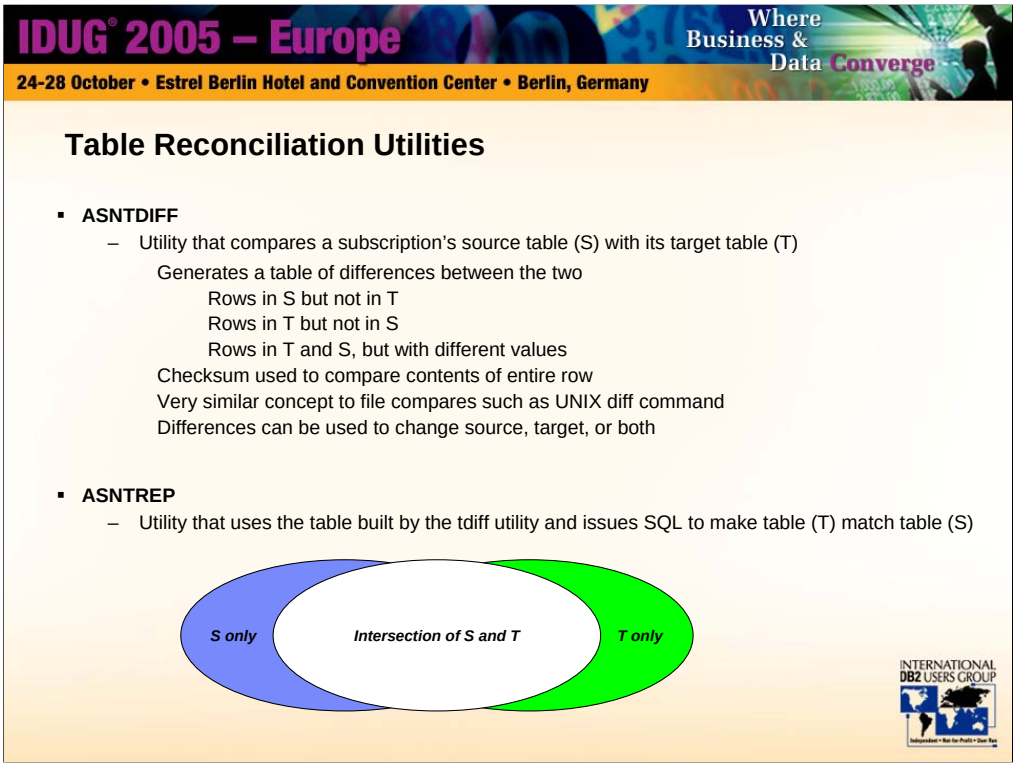
- Java API's
  - –Typically used when replication is embedded

As per V8.1, the administration is constructed on java APIs that can be called by a graphical user interface, a command, or a program. It is strongly recommended that the novice user become more familiar with the product first by using the GUI, which has launchpads, wizards, and online help to get the user up and running very quickly. The more experienced user may prefer and find it faster to create command line scripts . Only vendors embedding the replication product would typically use a programmatic interface to the APIs.

Manageability of replication has been improved in Q Replication in several ways: one queue can be defined as the staging area for literally thousands of source objects. There is only one replication mapping object to be defined – a subscription, rather than a registration plus a subscription. When many subscriptions will share common attributes, they can be built all at the same time using the mass subscription wizard.

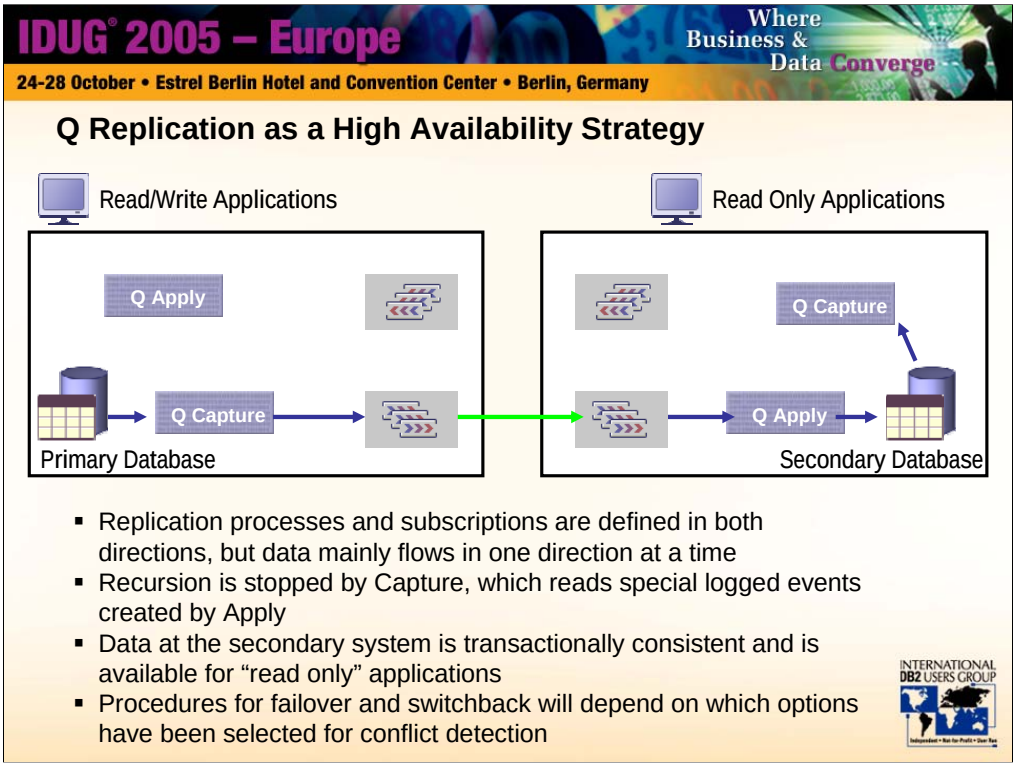


Here is the Create Q Subscriptions wizard. Note that multiple objects are being defined at once. The wizard walks the user through the various steps required to create subscriptions, and adjusts what steps are presented based on the user's choices , eg unidirectional vs bidirectional or peer to peer.

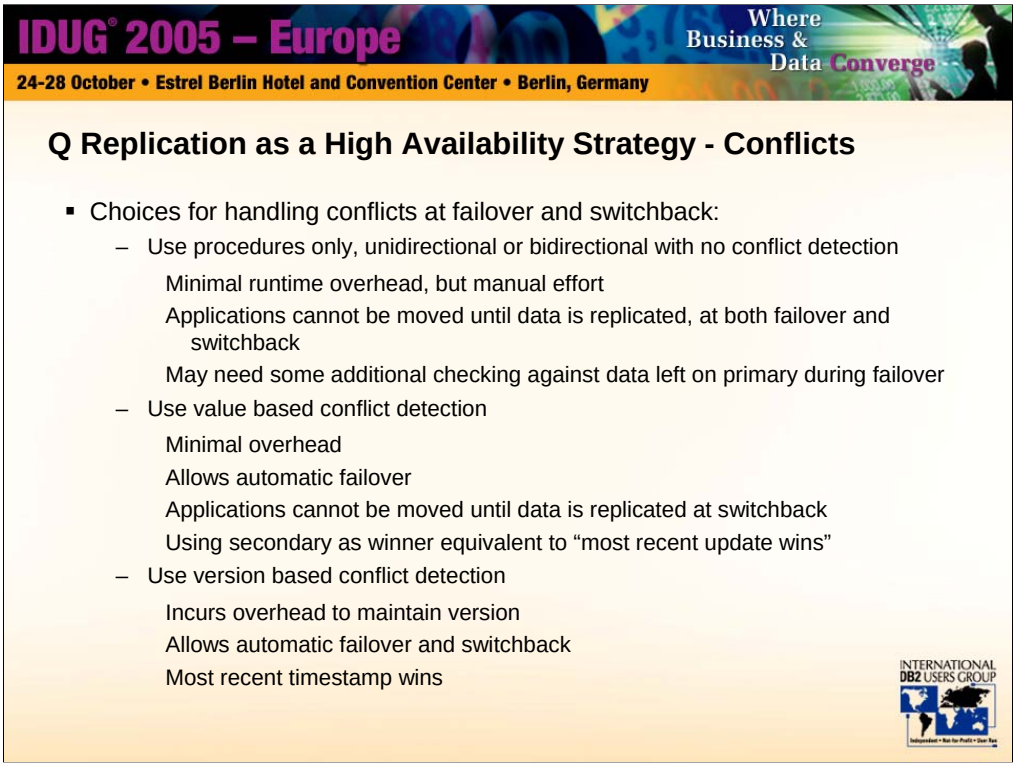


Tdiff has been very popular with early adopters, both external customers and also within IBM. One early project user ran 4 tdiff's in parallel (partitioned data via predicates) on one very large table to reconcile differences. This table included LOBs (2 LOBs per row), 4.5 GB of LOB data in source/target tables, and they achieved 4.2 GB/sec differencing speed.

Note: Q Replication provides 3 utilities (1) monitor (mentioned on prior slide), (2) tdiff/trep (discussed on this slide), and (3) q analyzer (not specifically discussed). Q Analyzer is a replication utility that is used as a serviceability aide. It looks at replication definitions/metadata and produces reports highlighting any detectable problems or inconsistencies in those definitions. This is available via download.



In the next few charts, we will explore the use of replication as a high availability strategy. First we look at the basic setup. Replication is configured for either bidirectional or peer to peer, depending on the choice the user makes on how conflicts need to be handled. In either case , the replication is typically configured so that data changes can be captured in both directions so that the replication is ready immediately to handle a switch to the alternate server. No matter how the system is configured, recursion of updates is stopped at the Capture process, based on information provided by the Apply processes.



So now that we have looked at the exposures, let's discuss the methods that can be used to handle these exposures.

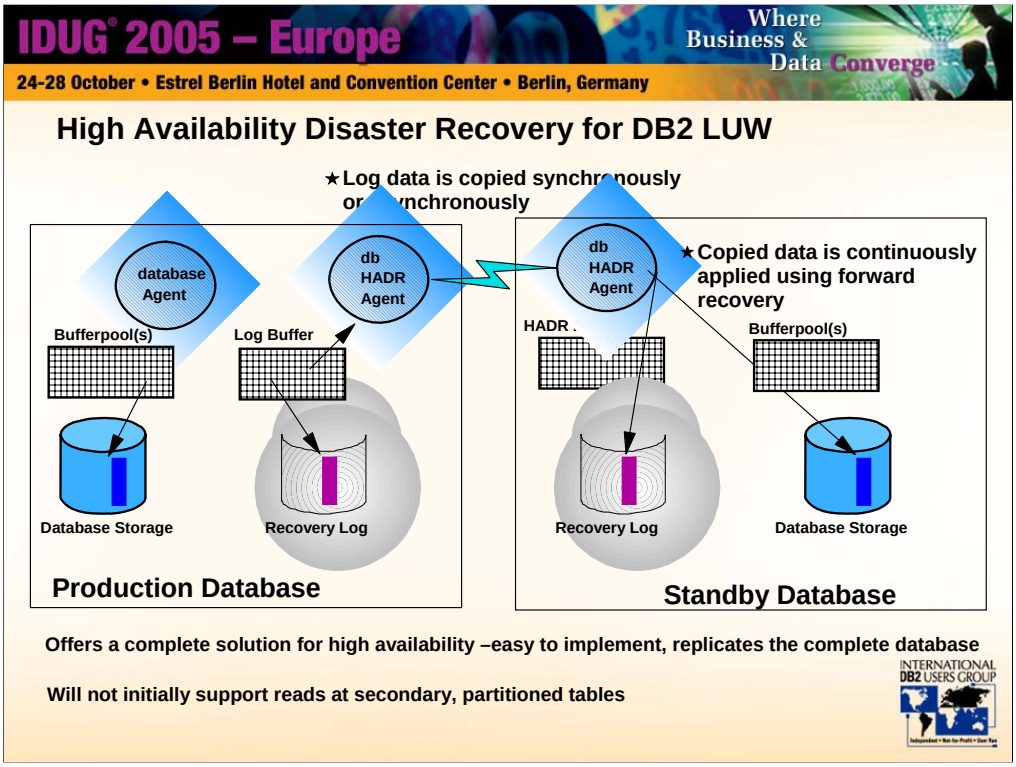
(1) Set up replication with no conflict checking (allowed only in bidirectional). Use procedures to control failover and switchback. During failover, do not allow write applications at the secondary until all queued transactions have been applied.

During switchback, use a method to resynchronize the databases to check for any conflicts between data stuck on primary at failover and data changes that have occurred since then. Move write applications to primary when procedures have verified the resynchronization.

(2)Set up replication to use bidirectional , basic conflict checking and resolution. Set the secondary server as the "winner". Allow failover to happen automatically, and conflicts will be resolved such that the newer transactions take precedence.

During switchback, old data transactions from the primary will similarly be resolved such that newer transactions will take precedence. During switchback, completely apply all secondary data changes and briefly quiesce the workload to allow this before moving write applications back to the primary.

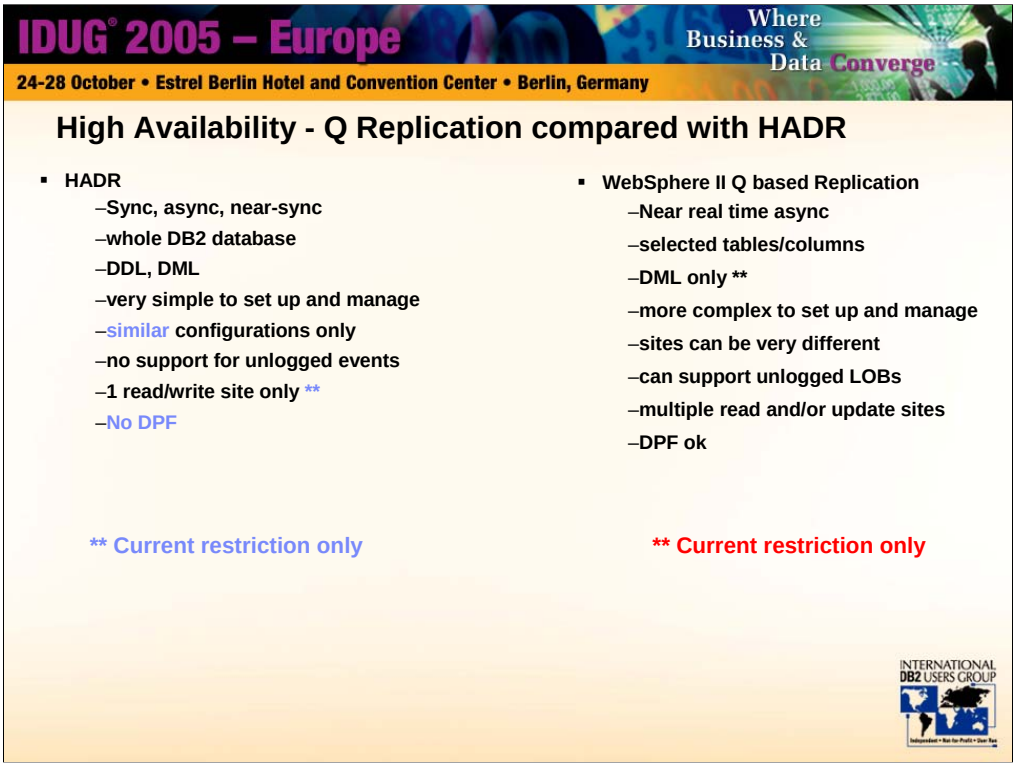
(3)Set up replication to use peer to peer, advanced conflict checking and resolution. Write applications can be moved at will from server to server, or even be directed to both servers simultaneously. No procedures are necessary. Most recent transactions will take precedence over older transactions.



When considering a high availability solution for DB2 on LUW, consider also log shipping and the soon to be offered HADR capability shown on this slide. In HADR, the data is replicated at a physical level, such that forward recovery is used to apply changes as seen in log records which have been sent from the primary to the standby.

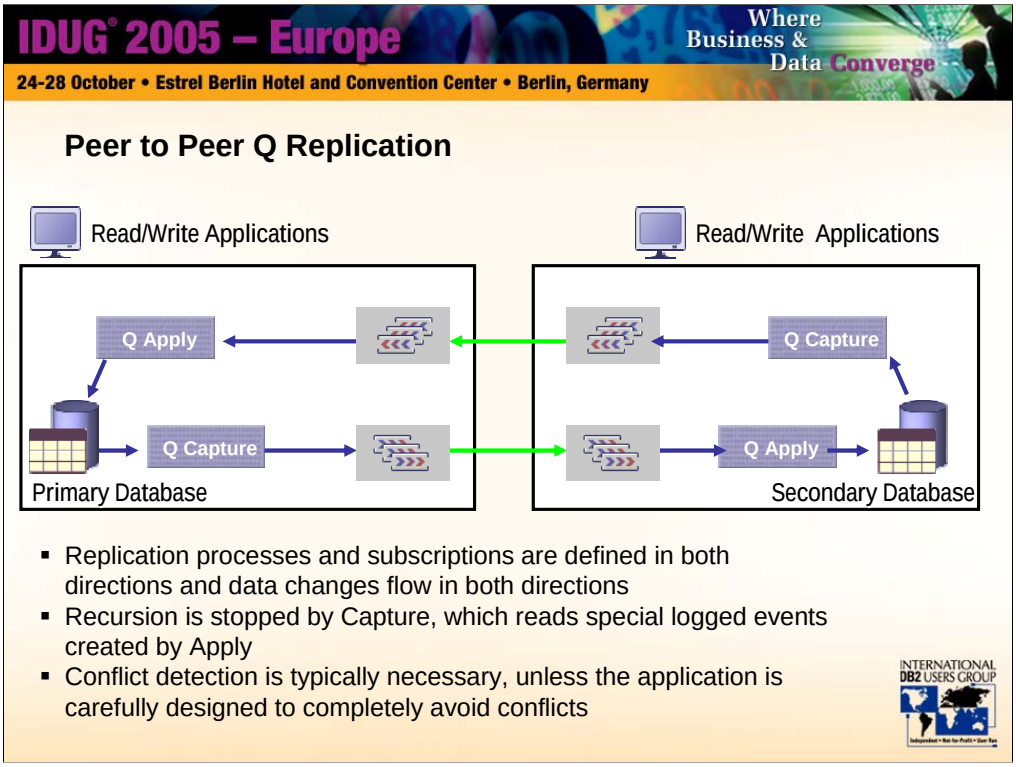
For a much more complete discussion of this topic, see the session from this same conference: **SL3: Spotlight Session - DB2 UDB for LUW's HA Strategy, Vision and Execution**
speaker: Dale McInnis of the IBM Toronto Labs.

There are advantages and disadvantages to the many high availability methods available (HADR, Q Replication, hardware), and you will want to carefully consider the benefits against your priorities. Some of these are examined in the following slide.



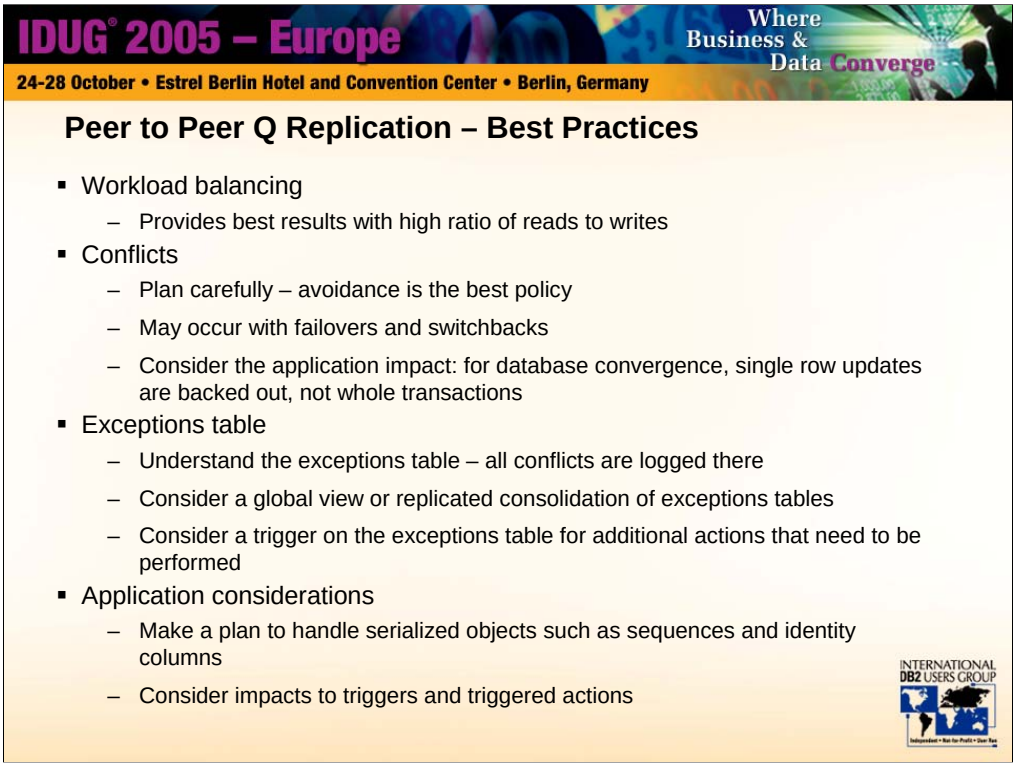
To net out this comparison, HADR is simpler to set up and manage because it covers the whole database. There are no filters or options to define. You do not have to consider issues such as identity columns and sequences, and you do not have to worry about adding new tables or altering existing tables – this is all covered.

So why do some users choose logical database replication? The most popular advantages are: (1)speed of takeover – the secondary in a replication configuration is live, (2) because the secondary database is live, it can be used for multiple purposes, and (3) because this is the only "logical" method available for high availability, it affords the most flexibility in hardware and software – primary and secondary systems can vary a great deal from one another.



Peer to peer configurations are used to allow for workload balancing, often to bring a database closer to the user for better performance and availability. This is certainly the case for many online applications that are used all around the globe.

Peer to peer replication might be used in a configuration in which the same data rows may be updated simultaneously at multiple sites at the same time, or it may be that the same database is updated simultaneously at multiple sites at the same time, but in such a controlled fashion that it is not expected that more than one site would ever update any one individual row. However, it is also frequently the case that in a peer to peer configuration, it is desirable that if any one site becomes unavailable, then the applications that use this site failover to an alternate site. At this point the characteristics are identical to those previously discussed in the earlier slides on high availability. So even when applications are carefully planned and designed, it is typically necessary to have a strong plan to handle the failover and switchback cases, inclusive of conflict detection and resolution.



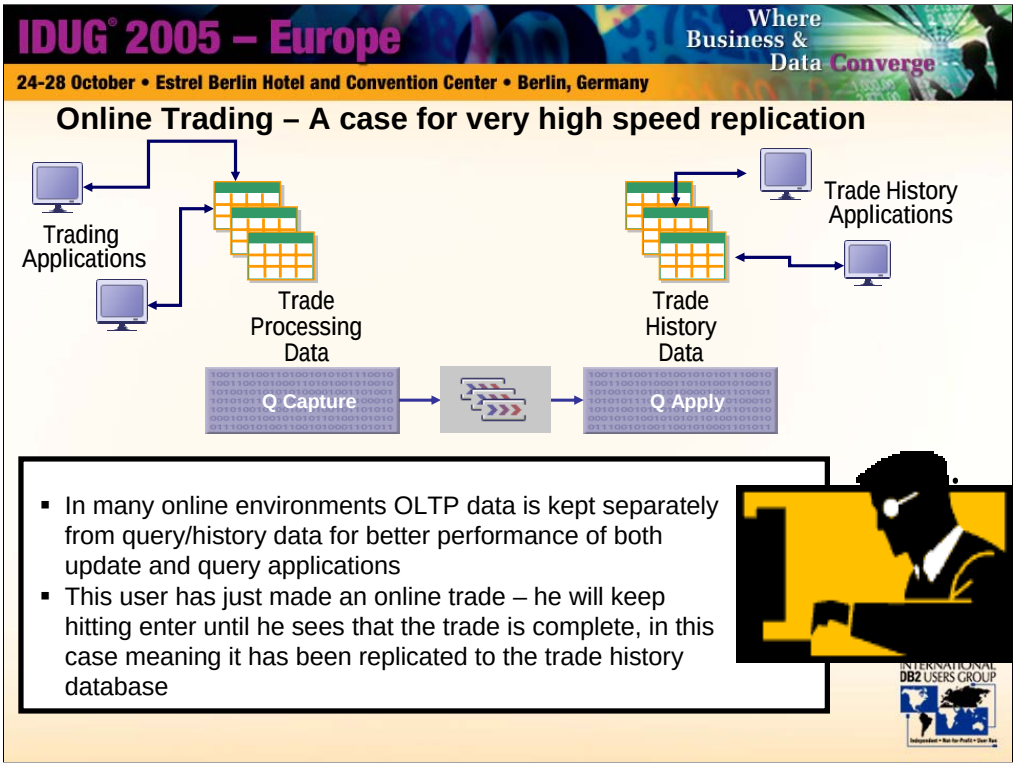
If a workload is mostly read, then very good workload balancing can be achieved through the use of multiple servers. For write operations, however, the replication of those writes means that the workload is not reduced or shared, it is increased – that is to say, each write will be performed at least twice, or more if there are more than 2 peers.

Even with the very best replication conflict detection and resolution mechanism, the important thing for any application designer to consider is what a conflict means to their business. Does it mean selling the same item twice? Does it mean that there is an exposure such that more money can be withdrawn than exists in an account? Because of these kinds of exposures or issues, it is typical that peer to peer replication that allows multiple updaters/owners of a row is more often used in applications that involve user profiles, preferences, and retail sales than in trading or financial applications.

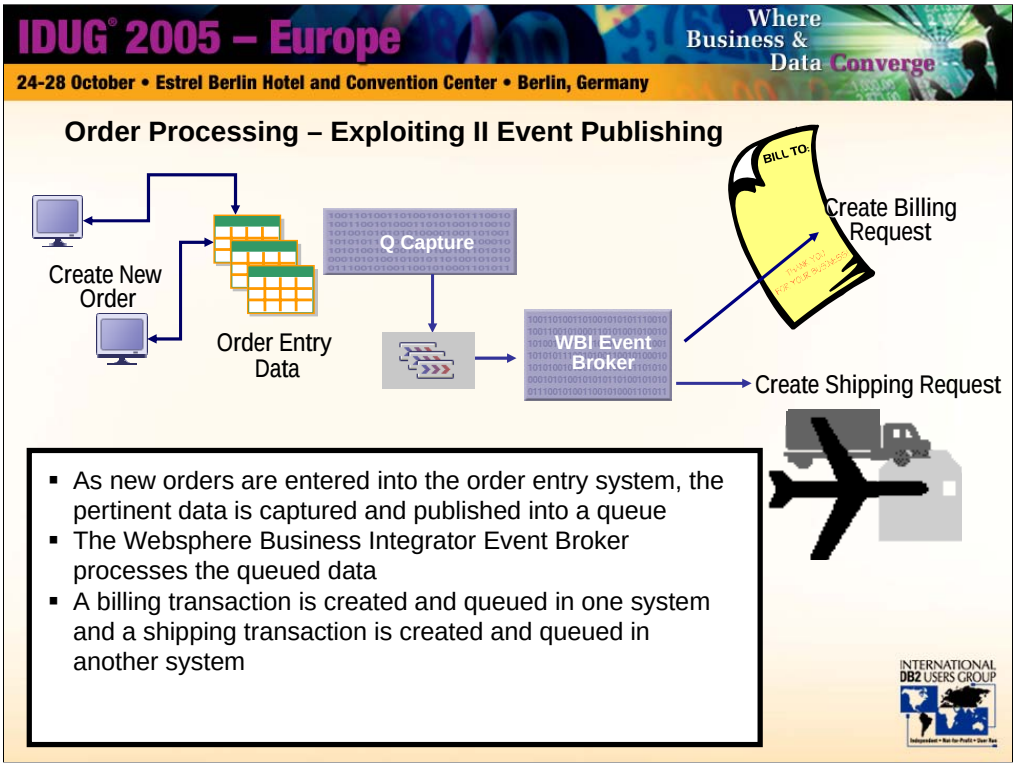
It is very easy to show that databases can diverge if conflicting rows are backed out at a transaction level, and we have therefore chosen to back out only the conflicting update – our primary aim is to create converged peer databases. The effect of a partially backed out transaction must therefore also be taken into consideration.

Use the apply exceptions table to find information on all rows that have been backed out based on conflicts.

Consider using methods to isolate sequential data such as allowing each peer its own set of unique entries (e.g. even/odd)

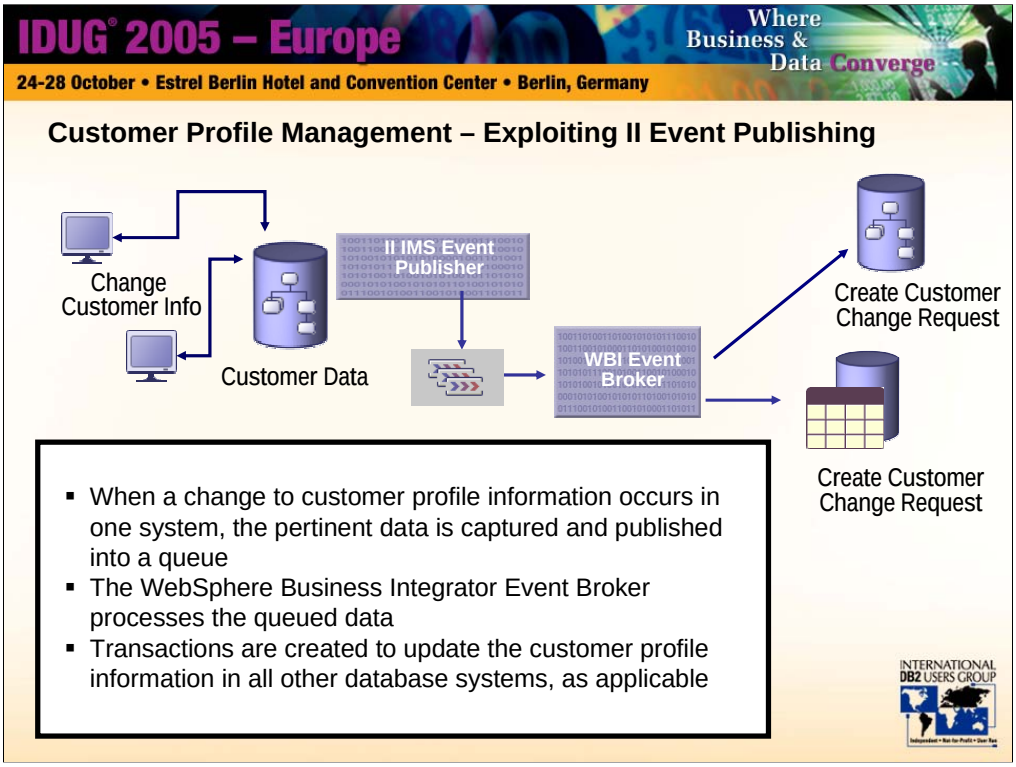


In many cases related business data is kept in two or more databases, possibly on different platforms, possibly from different vendors, and possibly in different formats. Different applications benefit from having the data in exactly the place where it is located and in that format, or it may just be that way because this is how the application was packaged. Replication has been bridging these gaps for years, but the new twist is that users have grown ever more demanding of their online environments. When a user makes an online trade, performs an online banking operation, uses frequent flyer points to buy or upgrade a flight, they want to verify that this action has occurred. Instant gratification is expected, and may be all the more desired because of insecurity of their internet actions. The user is not aware, and does not want to be aware, that multiple databases are used to run the business behind their actions. Many businesses are continuing to satisfy this database gap with replication, but they want it to be very fast, to provide the best seamless operation for their online customers.



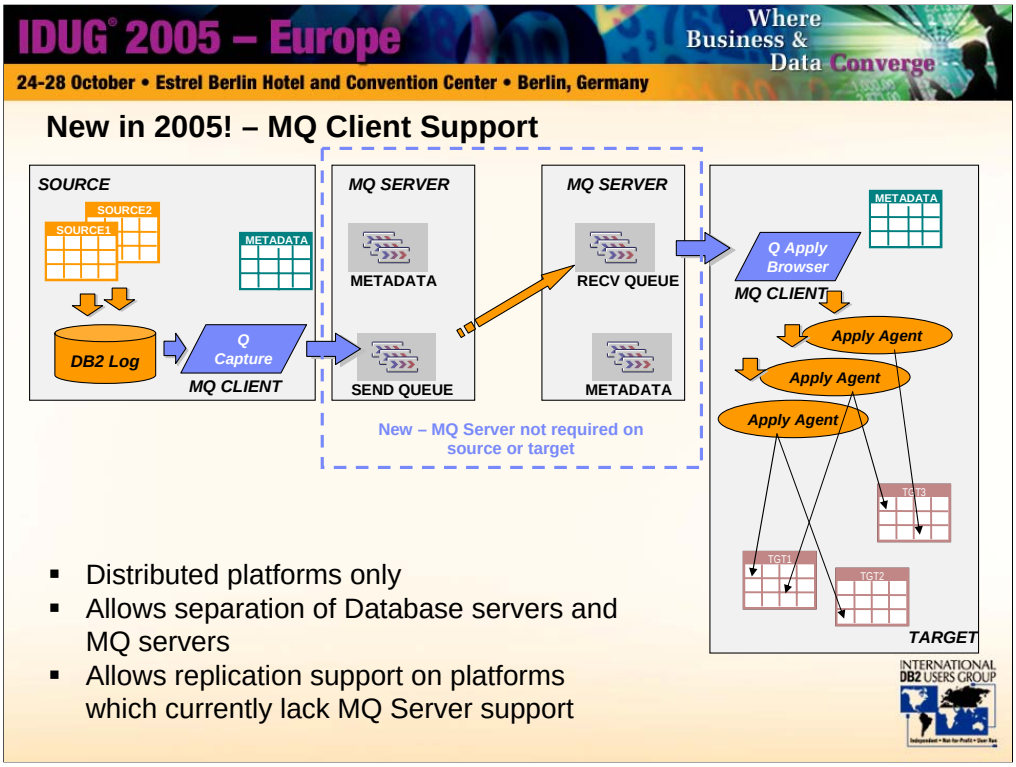
Business data needs to flow, within a company or between companies. There are many methods for creating this flow of data, and many businesses have turned to application messaging to perform this function. This is perhaps the very heart of service oriented architectures.

But here are 2 compelling reasons to use event publishing as the preferred method for application messaging – (1) the messages are created asynchronously from the originating application, reducing the performance impact on that application, and (2) the application is similarly shielded from any loss of availability of the message queue or service. New orders can continue to be created while the connectivity to the in-house billing system or the external shipping system is lost, even if queues are temporarily overfilled. When the connectivity is regained and message queues are again available, the orders can be processed and sent on.

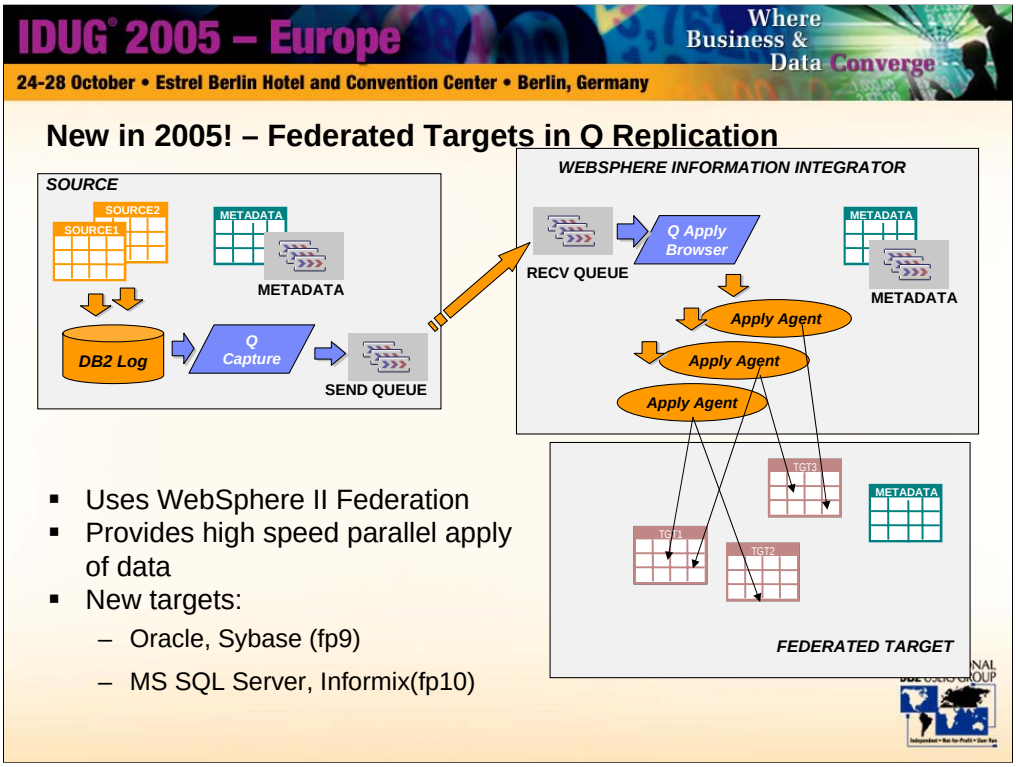


Here is another very common business issue – many systems exist, with a customer potentially defined in all or some of these systems. When such a customer has a change of information – change of name, address, status – it is highly desirable that this change be made in every system in which this customer exists within the business. This is a potent customer satisfaction issue.

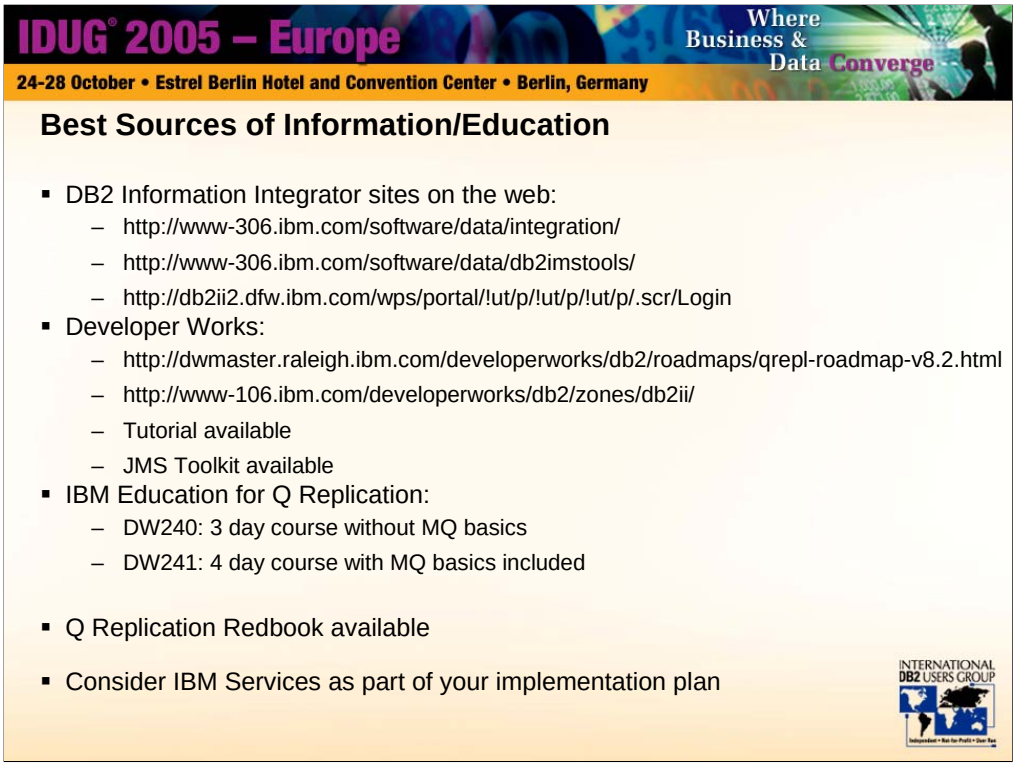
Using the event publisher capability, very disparate systems can be linked together and benefit by sharing such important events. Depicted here is the ability to capture an event that occurs in an IMS database and publish that event to both a second IMS database as well as to a DB2 database, where each of these databases represent various factions of the business.



This support was introduced in Fixpak 8.



Support for Oracle and Sybase target tables is available with fixpak 9 and support for Microsoft and Informix target tables will be available in fixpak 10. This support requires Websphere Information Integrator Replication Edition (also standard and advanced editions) and connector support to the desired database.



Or just google “db2 replication roadmap”. Currently the first hit listed is a page that shows you 2 DB2 replication roadmaps – one for Q Replication and one for SQL Replication. These roadmap pages are great for helping you find the information that you are looking for.

WebSphere II Q Replication – It's here! What is it?
Session: E02

Madhu Kochar
IBM
kochar@us.ibm.com