



DataExchange Deployment Strategies

Version 2.5

October 2003

Pervasive Software Inc.

Table of Contents

- DATAEXCHANGE OVERVIEW 3**
- DATAEXCHANGE TECHNOLOGY 4**
 - DataExchange Software 4
 - Replication Network 5
 - Operating Modes 5
 - Replication Template 6
- DATAEXCHANGE DEPLOYMENT STRATEGIES 7**
 - One-way Two Databases 7
 - One-way Multiple Databases 7
 - Two-way 9
 - Advanced Features 10
- CONCLUSION 11**
- FOR MORE INFORMATION 11**

DATAEXCHANGE OVERVIEW

Not long ago, database deployments fell under one of two distinct configurations – centralized databases on one large server accessed by all users, or individual database servers with limited views of the data. While these uses served their purposes, both entailed certain limitations. A central database is a single point of failure, and remote users with low bandwidth or sporadic connections challenge the configuration. In decentralized structures, maintaining an up-to-date view of the company database for all users is an equal challenge. Collectively, these trends point to new requirements on data availability: your customers require that mission-critical data must be where users need it when they need it. And, it must be safe and secure at all times.

Pervasive DataExchange™ offers an alternative to such large client-server deployments by supporting distributed applications which operate locally with users but exchange data with all systems in the network, thus giving users the speed of local access with the redundancy of multiple distributed systems. Rather than having one large system serve offices in New York and London, each location runs its own local application and has the databases synchronize throughout the day. Another feature of DataExchange is that it encrypts data before sending it to other locations. This means that you can replicate data across public networks, which eliminates the need for large, costly private networks. Combining this with a Virtual Private Network adds an even higher level of security.

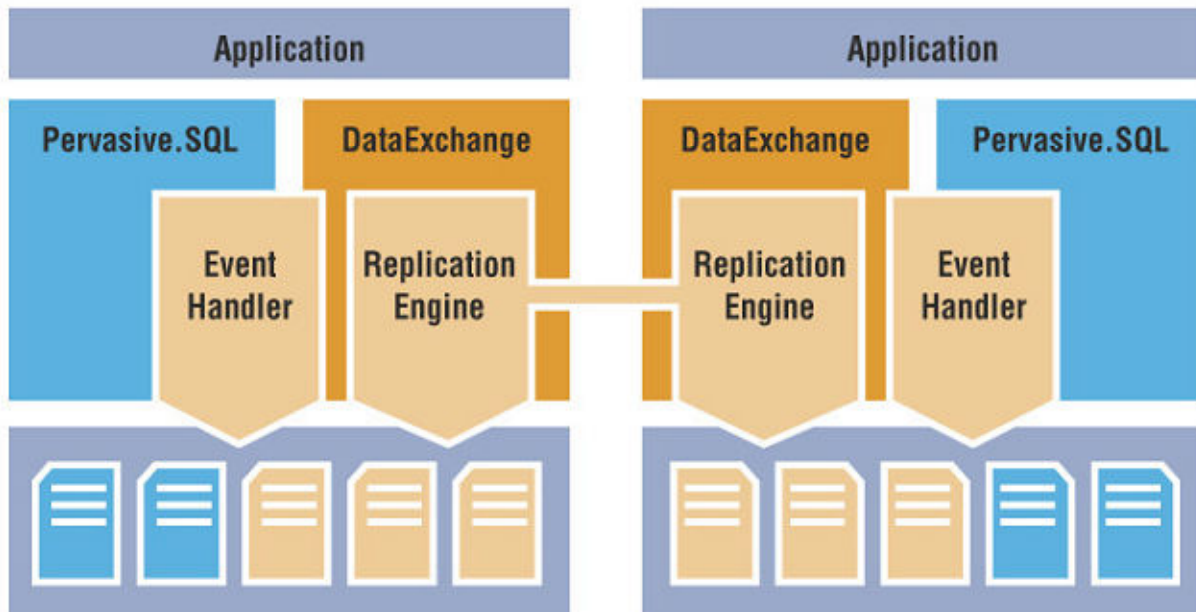
Pervasive DataExchange™, Pervasive's database replication solution, addresses these issues. It reliably moves data between two or more Pervasive.SQL™ databases to maintain warm backup systems, drive data into reporting servers, or synchronize multiple remote databases.

DATAEXCHANGE TECHNOLOGY

DATAEXCHANGE SOFTWARE

The DataExchange technology works by capturing and sharing changes from one Pervasive.SQL database to other databases in a DataExchange replication network. Each database is augmented with DataExchange, which adds two software components to a Pervasive.SQL database, the Replication Event Handler and the Replication Engine.

- The **Replication Event Handler (REH)** plugs directly into the Pervasive.SQL database engine. Specifically, the REH is a set of DLLs that operate within the database engine, so if the database is running, the replication event handler is running as well. The database engine activates the REH when there is a change event (insert, update, delete). The REH then makes note of the event in one of its private control tables.
- The **Replication Engine** is a separate process that performs the actual replication task. It reads the control tables to determine what records have changed since the last replication session. It then groups these changes into packets and shares them with the other replication engines participating in a DataExchange network. These other engines then apply the updates to their own databases. The replication process occurs either continuously, at scheduled intervals, or on demand. The replication engine does not require constant network connectivity, so DataExchange is suitable for scenarios with intermittent connectivity, like remote users with dialup connections.



In addition, DataExchange includes a full suite of design, maintenance and monitoring tools. DataExchange comes in two editions. The Real-Time Backup Edition is optimized for quickly and easily maintaining a backup server. It pairs high performance with minimal installation and maintenance. The Data Synchronization Edition is optimized for more complex replication scenarios. It supports multiple participating databases and data flowing in multiple directions.

REPLICATION NETWORK

A Replication Network refers to a collection of databases spread across two or more computers that exchange information between duplicate database configurations that resides at each site. While there may be dozens of database servers all running on an internal or external network within a company, a “Replication Network” only refers to those servers that exchange data and does not take into consideration the actual IT network with which they all communicate over. Additionally one server may potentially be part of two separate replication networks. The following diagram illustrates three separate replication networks that exist within a single company. Databases within the networks exist at three physical locations, the corporate headquarters, a regional office, and a sales person’s personal laptop. The three databases depicted in the IT department of the headquarters all reside on one computer and are part of all three DataExchange networks.



Also note that the Customer DataExchange network is comprised of more than two databases. Replication networks can exist across LAN, WAN or a combination of any TCP/IP based networking including the public Internet. DataExchange encrypts all data prior to transmitting for security across private and public networks.

OPERATING MODES

DataExchange operates in two different modes, transactional and relational. The fundamental difference between the modes is visibility into the data. In transactional mode, data is accessed through Btrieve calls and as such each record is its own entity. In relational mode, data is accessed

through SQL calls, giving the replication engine the ability to view the columns comprising each record. Both modes have their plusses and minuses, which will be described briefly below.

Transactional mode is the easiest to set up, deploy and maintain. This mode provides direct access to the data via the Btrieve engine and as such realizes the true speed of Btrieve. The only requirement for data is that each table must have its system keys activated. System key is a Pervasive.SQL database managed unique identifier. Because the data is accessed as a record level entity, transactional mode does not have the advanced features that are available in relational mode. Transactional mode supports both one-way (unidirectional) replication and two-way (bidirectional) replication.

Relational mode places more requirements on a database but provides more control in the way data is replicated. Relational mode requires data to be accurately represented by DDFs, something legacy Btrieve databases may not have. Additionally, changes to the schema of a database after replication has been enabled may require reconfiguration; this can be a complex task depending on the number of sites replication is deployed to. However, the ability for the replication engine to see data within a record at a column value level can provide Work Sets and Fragments to be used during replication. More on this in the Deployment Strategies section below. Relational mode supports only two-way (bidirectional) replication.

REPLICATION TEMPLATE

Before a database can be added to a replication network, you must first define the rules for replicating it. If you are using the Real-Time Backup Edition, these rules are necessarily simple, and are usually configured using the Dxdeploy utility. If you are using the Data Synchronization edition or are configuring a complex real-time backup scenario, you will use the more powerful DataExchange Replication Designer. With the designer, you identify the tables in the database you wish to replicate and provide rules for managing the synchronization process. The designer combines these rules with other internal information to create a DataExchange *template*. A template records the configuration of a replication network. It is then used to activate the member databases of the replication network. In the example network, a template is created for the Customer Database. This template is then used to activate each of the databases at each of the different locations. After the databases are activated they may replicate with one another based on the rules defined in the template

For transactional mode, the template simply defines which tables in the database you wish to replicate (it is not required that all the tables in a database replicate, its up to you). All the databases in a transactional mode replication network send all of their records to the other sites.

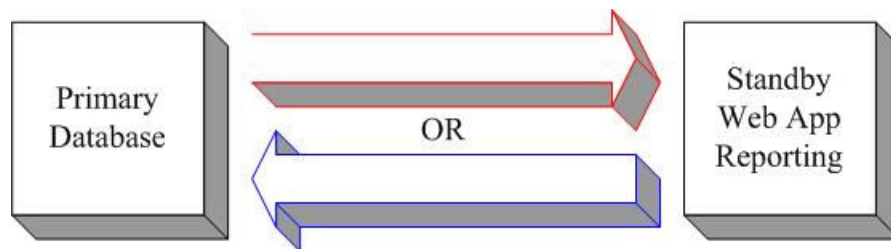
A relational mode template can be much more complex. Because it can see the columns comprising a record, it can be configured to make data dependent replication decisions. You can subscribe to particular sets of data or even replicate data on a sub-record level. In the example network above, the Regional office can subscribe to the customer database at the Headquarters and replicate only the customer information for customers in their region. The outside sales person may then subscribe to only the customers that they work with from the regional office database. The rules allowing data to be divided this way are defined in the Replication Designer and stored in the Replication Template.

DATAEXCHANGE DEPLOYMENT STRATEGIES

Now that all the parts of a DataExchange are defined we can put the pieces together to create data management solutions. The following examples will proceed from the easiest to setup and maintain to the more powerful and complex.

ONE-WAY TWO DATABASES

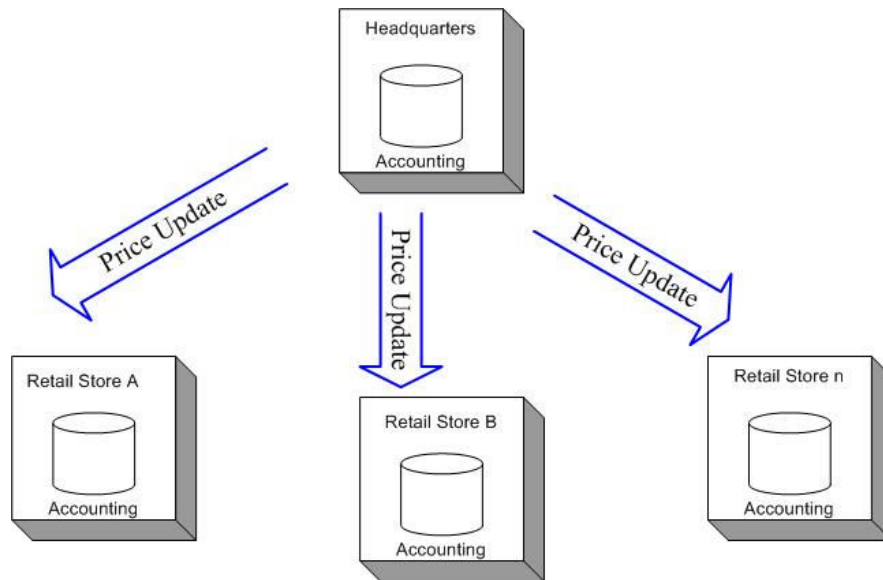
One-way replication using transactional mode is the easiest to set up and maintain. This is the only mode supported by the DataExchange Real-Time Backup Edition. This solution is used most often to deploy a warm-standby server in the event that your primary database goes down. Changes made to the primary database are constantly replicated to the backup. In the event of a disaster, clients may connect to the standby while the primary is being repaired. When the primary is made available again, the changes on the standby may be replicated back. This solution is also useful for creating a secondary database in which lower priority task can be executed against, such as running reports or providing data to web applications. The activity that takes place on the secondary machine will not impact the performance of users on the primary system. In each of these cases data is replicated in only one direction at a time. It is important that changes be made to only one database at a time when using this type of deployment.



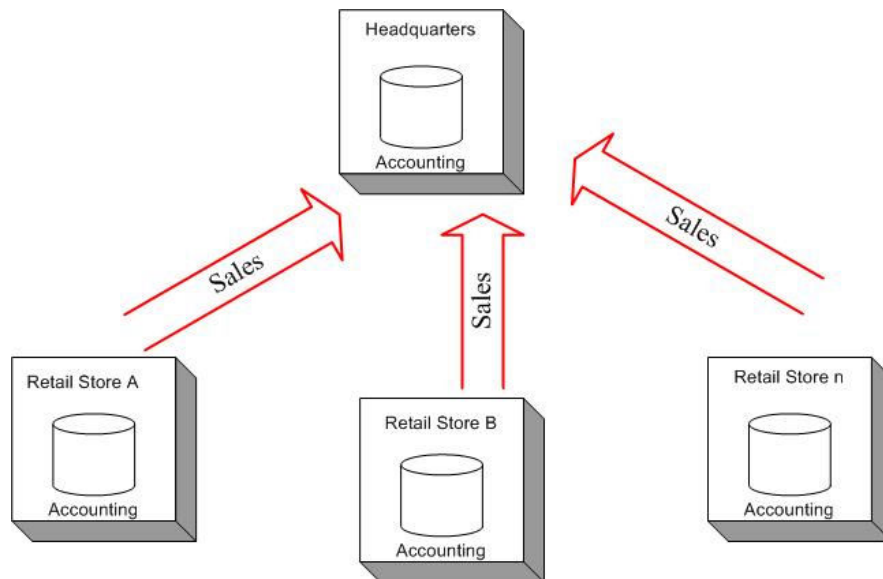
ONE-WAY MULTIPLE DATABASES

One-way replication in transactional mode is not limited to two databases. This mode allows complex publish and subscribe replication networks. An example use is synchronizing data between disconnected offices

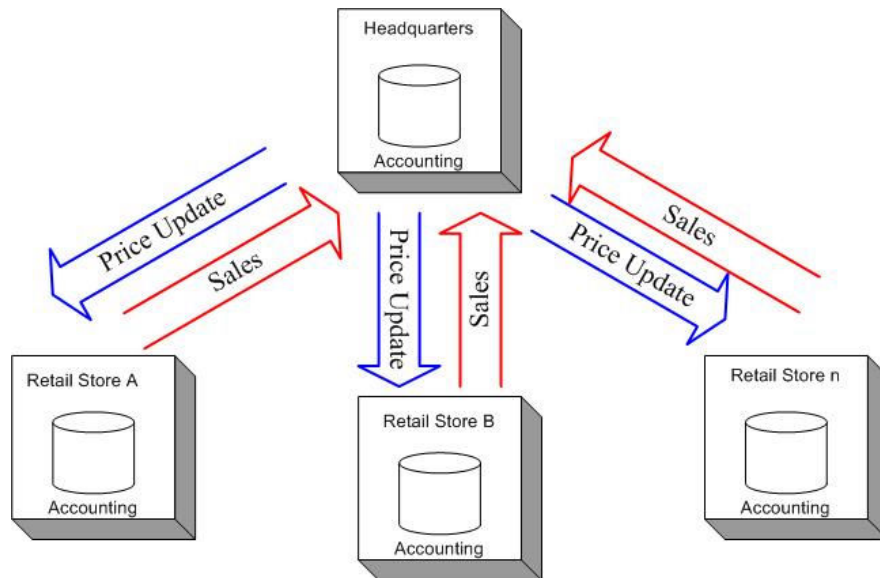
In this example of a 1-to-n replication deployment, a headquarters database sets the price for all items sold at local retail stores. As changes are made to the headquarters database, they are replicated to the local databases at each of the retail stores.



Replication can go in the other direction as well. In this example of an n-to-1 deployment, the different retail stores continuously replicate their sales data back to the headquarters.



Recall earlier we mentioned that you don't have to replicate all the tables in a database. You can use tables from the same database and create two completely separate replication networks.

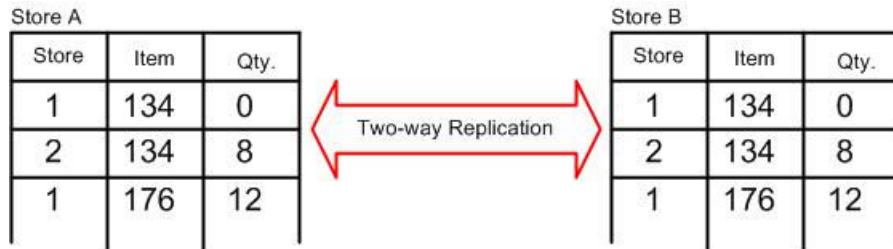


In all the examples listed here, data is replicated in only one direction. This is important to note because the rules of one-way replication eliminate the possibility for conflict in the database. For instance if one of the retail stores set the price of a particular item to \$1 and the price for that item is set to \$2 at the headquarters, the retail store's will be overridden by the price set by the headquarters. The site that is replicating (transmitting) the data will always override the partner sites it is replicating to. It should also be noted that the Sales tables in the above example would need to include a column that describes the store from which the data is being replicated. A's sales would not want to be overwritten by B's or any other stores data. By including this "location" column in the database schema, each stores data can co-exist in the same Sales table

TWO-WAY

All of the deployment strategies up to this point have been one direction; price tables at the headquarters database replicating to retail stores, and stores' sales tables replicating to the headquarters. The following examples illustrate how tables within the same database can be operated on at all locations at the same time. NOTE: Because users at different locations could potentially add or modify the same record within the same replication cycle, care must be taken to minimize this possibility of "conflict". A database's schema or application logic must take into account that data is being entered in the database at another location. For a deeper explanation of these issues see the white paper [Application Development for Replication](#).

Two-way replication allows changes made to each node in a replication network to both send and receive changes, in effect keeping exact copies of a single database at multiple locations. In the example above, the retail stores sent their sales information to the headquarters. The headquarters therefore had a complete account of the inventory for each particular store. However, if someone called Store A asking for a product that that was out of stock, they would not be able to tell the customer that Store B had this item in stock; only the headquarters has this information. Employees could then direct the customer to the nearest store stocking the desired item..



By enabling two-way replication between all the stores, data could be shared such that the complete inventory picture is available at all stores. Databases at the stores constantly send and receive data, keeping all the store databases synchronized.

ADVANCED FEATURES

You can begin to see that the two-way replication described above could, over time, grow the size of the database at each store to a size that is not efficient for that store to do its work. In some cases there may not be a need to share all the data, only particular segments of data that is relevant to each location. For instance it would not be necessary for personnel information to be shared among all the stores, each store should only replicate data that pertains to their location. Using the Data Synchronization Edition with advanced features, it is possible to set up each location to only “subscribe” to certain data as it pertains to that location while at the same time allowing other data in the database to be fully replicated. The following items are only available in the relational replication mode.

- **Work Sets** describe subdivisions of information within the database based on a hierarchical data structure defined through Primary/Foreign keys. While designing the template for DataExchange, you may include this hierarchical information into the rules of replication. After locations are activated into the replication network, each site may then “subscribe” to any particular data and all the subsequent related data based on this structure. In the example above, a column in the “Store” table serves as the primary key for the related records in the “Personnel” table. Each site subscribes to their particular “Store” value and then only the Personnel records related to that store are replicated back and forth between those two sites. This is useful in conserving bandwidth during replication and as providing security to non-relevant data.
- **Fragments** group data within a record by that record’s columns. By default all columns in any given table together form a base fragment so that if any information in that fragment is changed, the entire record replicates. You can subdivide columns into multiple fragments to minimize the amount of data replicated and also to minimize the possibility of the same data within a given record to conflict during replication. For example a table may have the columns “FirstName, MiddleInitial, LastName” grouped in one fragment and “Address, City, State, Zip” grouped in another. If the Address info in any of these records is changed, only the second fragment gets replicated. This conserves the amount of data that is required to be transmitted thus saving bandwidth.
- **Transaction Sets** group tables together using referential integrity rules. By default replication process tables alphabetically. All records from each table are replicated one at a time until all tables are complete. If your application is utilizing referential integrity, transaction sets allow

the enforcement of RI to take place within the replication process. Using the hierarchical description as in Work Sets, base records are replicated, then all related records, regardless of which table these related records exist in. All record updates are applied within a single transaction to ensure that RI is enforced. In the example above, the Store table would be replicated, then the relevant Personnel records. There would never be a situation where a Personnel record existed without a proper parent Store record.

The advanced features available in the Data Synchronization Edition do provide a highly configurable replication environment, but the cost is performance and maintainability. Because DataExchange accesses data through the SQL interface, there is additional processing required than if the data was accessed on the native Btrieve interface. Also, multiple table references and complex RI schemas add overhead to replication sessions as well. In the event that the schema of the database changes, something as simple as adding or dropping a column from a single table, all sites in that replication network must be reactivated with updated rules based on that new schema. Only use this mode if advanced features are needed.

CONCLUSION

The advantages of implementing Pervasive DataExchange into your application are numerous. It only requires a modest amount of planning to extend these advantages to your current production environment. This paper touches on the various replication deployment strategies the benefits each provide.

Decide which features best suit your needs, the type of replication you are looking for and apply the techniques outlined above to realize the power of your application with Pervasive DataExchange.

If you have further questions or would like a more in-depth consultation on the incorporating Pervasive DataExchange in your application, please contact the DataExchange group as detailed below. We look forward to working with you to realize the untapped potential of replicating your data across your business locations.

FOR MORE INFORMATION

For more information about Pervasive DataExchange, please visit our Web site at www.pervasive.com/dataexchange, email us at info@pervasive.com or locate one of our sales offices at www.pervasive.com/company/contact/index.asp.

©2003 Pervasive Software Inc. Pervasive, Pervasive.SQL, Btrieve, and the Pervasive logo are registered trademarks of Pervasive Software Inc. All other product names may be trademarks of their respective companies. All rights reserved worldwide.

Disclaimer: The performance demonstrated in this document is for reference purposes only and does not constitute a performance guarantee even if under similar configurations.