

# ORACLE'S SOLUTIONS FOR THE DISTRIBUTED ENVIRONMENT

*An Oracle White Paper*  
*06 2001*

# ORACLE'S SOLUTIONS FOR THE DISTRIBUTED ENVIRONMENT

EXECUTIVE OVERVIEW .....	4
INTRODUCTION.....	4
RAPIDLY CHANGING BUSINESS ENVIRONMENT .....	4
DISTRIBUTED ENVIRONMENT.....	5
ORACLE TECHNOLOGIES FOR DISTRIBUTED ENVIRONMENTS.....	6
ORACLE-TO-ORACLE .....	6
LOCATION TRANSPARENCY.....	7
SQL AND COMMIT TRANSPARENCY .....	7
DISTRIBUTED QUERY OPTIMIZATION.....	8
ADVANCED QUEUING.....	8
ASYNCHRONOUS APPLICATION INTEGRATION.....	9
EXTENSIBLE INTEGRATION ARCHITECTURE .....	10
HETEROGENEOUS DATA INTEGRATION.....	10
APPLICATION LOCATION TRANSPARENCY.....	11
UNIQUE MESSAGE MANAGEMENT BENEFITS.....	11
UNIQUE FEATURES FOR E-BUSINESS.....	11
MANAGEMENT WITH ORACLE ENTERPRISE MANAGER.....	12
ADVANCED REPLICATION.....	12
MULTIMASTER REPLICATION.....	13
MATERIALIZED VIEWS.....	14
DATA SUBSETTING .....	15
DEPLOYMENT TEMPLATES.....	15
DISCONNECTED COMPUTING .....	15
CONFLICT RESOLUTION.....	15
REPLICATION MANAGEMENT TOOL IN ORACLE ENTERPRISE MANAGER.....	16
GENERIC CONNECTIVITY AND TRANSPARENT GATEWAYS.....	16
HETEROGENEOUS TRANSPARENCY .....	16
CONNECTING DISPARATE SYSTEMS.....	17
SQL TRANSLATIONS.....	17
DATA DICTIONARY TRANSLATIONS .....	17
DATATYPE TRANSLATIONS.....	18
HETEROGENEOUS CONNECTIVITY PROCESS ARCHITECTURE .....	18
HETEROGENEOUS SERVICES(HS) .....	18

AGENT.....	19
GENERIC CONNECTIVITY.....	20
ORACLE TRANSPARENT GATEWAYS.....	20
CONCLUSION.....	20

# ORACLE'S SOLUTIONS FOR THE DISTRIBUTED ENVIRONMENT

## **EXECUTIVE OVERVIEW**

As companies grow and expand they are often faced with the challenge of operating in a disparate environment. The Oracle9i platform provides a robust and complete solution that addresses all the needs that may arise when operating in a distributed environment. It includes communication between applications/users on the Oracle database using queues, data replication and distributed data access in both homogeneous and heterogeneous environments.

## **INTRODUCTION**

As companies grow and expand they are faced with the challenge of effectively and economically operating in a disparate environment. To thrive in today's dynamic and fast paced business environment, data sharing and integration is required across applications and departments.

The Oracle9i platform provides a robust and complete solution that addresses all the needs that may arise when operating in a distributed environment. It includes communication between applications/users on the Oracle database using queues, data replication and distributed data access in both homogeneous and heterogeneous environments.

This paper will cover integration of Oracle databases using distributed SQL, Advanced Queuing which handles the message queuing functionality of the Oracle database, Replication which is used for protecting the availability of critical applications and Oracle Transparent Gateways which allows transparent data access to non-Oracle systems from an Oracle environment.

## **RAPIDLY CHANGING BUSINESS ENVIRONMENT**

Let's use company XYZ as an example to demonstrate today's rapidly changing business environment. Company XYZ is a multinational company with sales offices in all major cities worldwide. Over the years each region has developed its own information system with customized applications based on different databases.

With this fragmented data model, the company is faced with a enormous challenge to collaborate and access information across offices. Although each

region can provide an accurate read on its piece of the business, to have a global and consolidated view of the company as a whole, all the systems, even though diverse, need to be integrated and work seamlessly.

The company has several sales channels: online via their web site, a direct sales force, as well as through partners. The company needs to be able to track and fulfill orders from all sources including their partners in a timely manner. It is critical that the company's web site be available 24 x 7.

Although the company has offices in many countries, the product catalog must be the same at each office. The catalog information is stored centrally in a database and has to be disseminated to each office periodically.

Its sales force application requires data to be periodically synchronized between the central database system and the remote sites, which are often disconnected from the central database. Members of a sales force must be able to complete transactions while disconnected from the central database for example place sales orders, from customer sites.

It is not feasible for the company to replace all the existing diverse applications with one application, since this would be too costly and disruptive. Even if this would be possible company XYZ has no control over the applications that its partners use, thus it still has to operate in an environment with diverse system.

In order for company XYZ to be successful it has to be able to quickly, efficiently, and economically access/exchange data with all its offices and partners. This requires the diverse systems to be integrated and work together seamlessly. Essentially the company has to operate in a distributed environment.

## **DISTRIBUTED ENVIRONMENT**

A distributed environment is a network of disparate systems that seamlessly communicate with each other. Each system in the distributed environment is called a node. The system to which a user is directly connected is called the local system. Any additional systems accessed by this user are called remote systems.

A distributed environment allows applications to access/exchange data from the local and remote systems. All the data can be simultaneously accessed and modified.

While a distributed environment enables increased access to a large amount of data across a network, it must also hide the location of the data and the complexity of accessing it across the network.

There are several reasons why a company will be operating in a distributed environment. Sometimes this happens by choice, for example company XYZ chooses to exchange data with their partners and sometimes it is simply the state of affairs, that results from corporate mergers and acquisitions.

In order for company XYZ, to operate successfully in its distributed environment, it essentially has to be able to:

- Exchange data between homogeneous databases
- Exchange data between heterogeneous databases
- Replicate between databases
- Communicate between applications
- Exchange information with customers, partners, and suppliers

### **ORACLE TECHNOLOGIES FOR DISTRIBUTED ENVIRONMENTS**

As with company XYZ, most companies are faced with the need to integrate many diverse systems.

Oracle offers a complete and robust solution for global enterprises like company XYZ. It has solutions to address all the needs that a typical multinational company might face. These solutions, all part of the Oracle database, provide a highly reliable, secure and consistent solution for operating in an environment with disparate systems.

Oracle has great support for communicating between Oracle databases using distributed SQL, while maintaining location transparency and data integrity. To address the needs of applications that access non-Oracle system, Oracle offers Oracle Transparent Gateways and Generic Connectivity. To protect against failovers and to ensure the availability of critical applications, Oracle Advanced Replication can be used. And finally for exchanging information with customers, partners, and suppliers and coordinating business processes there is Advanced Queuing.

Each of the solutions and how they address company XYZ's needs are described in detail in the sections following.

### **ORACLE-TO-ORACLE**

A homogenous distributed database system is a network of two or more Oracle databases that reside on one or more machines. An application can simultaneously access or modify the data in several databases in a single distributed environment.

An Oracle distributed database system can be made transparent to users making it appear as though it is a single Oracle database. Consequently, developers and users of the system do not have to deal with complexities that would otherwise make distributed database application development challenging and detract from user productivity. Company XYZ, can use this distributed SQL feature to make all its Oracle databases look like one and thus reduce some of the complexity of the distributed system.

Oracle uses database links to enable users on one database to access objects in a remote database. A local user can access a link to a remote database without having to be a user on the remote database.

### **LOCATION TRANSPARENCY**

An Oracle distributed database system has features that allow application developers and administrators to hide the physical location of database objects from applications and users. Location transparency exists when a user can universally refer to a database object such as a table, regardless of the node to which an application connects. Location transparency has several benefits, including:

- Access to remote data is simple, because database users do not need to know the physical location of database objects.
- Administrators can move database objects with no impact on end-users or existing database applications. Typically, administrators and developers use synonyms to establish location transparency for the tables and supporting objects in an application schema. For example, the following statements create synonyms in a database for tables in another, remote database.

```
CREATE PUBLIC SYNONYM emp
FOR scott.emp@sales.us.americas.acme_auto.com
CREATE PUBLIC SYNONYM dept
FOR scott.dept@sales.us.americas.acme_auto.com
```

Now, rather than accessing the remote tables with a query such as:

```
SELECT ename, dname
FROM scott.emp@sales.us.americas.acme_auto.com e,
scott.dept@sales.us.americas.acme_auto.com d
WHERE e.deptno = d.deptno;
```

An application can issue a much simpler query that does not have to account for the location of the remote tables.

```
SELECT ename, dname
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

In addition to synonyms, developers can also use views and stored procedures to establish location transparency for applications that work in a distributed database system.

## **SQL AND COMMIT TRANSPARENCY**

Oracle's distributed database architecture also provides query, update, and transaction transparency. For example, standard SQL statements such as SELECT, INSERT, UPDATE, and DELETE work just as they do in a non-distributed database environment. Additionally, applications control transactions using the standard SQL statements COMMIT, SAVEPOINT, and ROLLBACK

Unlike a transaction on a local database, a distributed transaction involves altering data on multiple databases. Consequently, distributed transaction processing is more complicated, because Oracle must coordinate the committing or rolling back of the changes in a transaction as a self-contained unit. In other words, the entire transaction commits, or the entire transaction rolls back. Oracle ensures the integrity of data in a distributed transaction using the two-phase commit mechanism. In the prepare phase, the initiating node in the transaction tasks the other participating nodes to promise to commit or roll back the transaction. During the commit phase, the initiating node asks all participating nodes to commit the transaction. If this outcome is not possible, then all nodes are asked to roll back.

All participating nodes in a distributed transaction should perform the same action: they should either all commit or all perform a rollback of the transaction. Oracle automatically controls and monitors the commit or rollback of a distributed transaction and maintains the integrity of the global database (the collection of databases participating in the transaction) using the two-phase commit mechanism. This mechanism is completely transparent, requiring no complex programming or other special operations to provide distributed transaction control.

## **DISTRIBUTED QUERY OPTIMIZATION**

Distributed query optimization is an Oracle feature that reduces the amount of data transfer required between sites when a transaction retrieves data from remote tables referenced in a distributed SQL statement. Distributed query optimization uses Oracle's cost-based optimization to find or generate SQL expressions that extract only the necessary data from remote tables, process that data at a remote site or sometimes at the local site, and send the results to the local site for final processing. This operation reduces the amount of required data transfer when compared to the time it takes to transfer all the table data to the local site for processing. Using various cost-based optimizer hints such as DRIVING\_SITE, NO\_MERGE, and INDEX, you can control where Oracle processes the data and how it accesses the data.

## **ADVANCED QUEUING**

Company XYZ faces the insurmountable task of integrating business processes internally and beyond. Like many other companies they have developed a variety of autonomous and distributed applications to automate business processes, and

manage business tasks. However, these applications need to communicate with each other, coordinating business processes and tasks in a consistent, reliable, secure, and autonomous manner. They also need to efficiently exchange information with customers, partners, and suppliers over low-cost channels such as the Internet, while preserving a traceable history of events--a requirement previously satisfied through now obsolete paper forms.

For loose application coupling such as that required by company XYZ, Oracle offers Advanced Queuing (AQ), a technology far ahead of the traditional message-oriented middleware which addresses only a sub-set of these needs. AQ's integration with the database brings unprecedented levels of functionality, operational simplicity, security, reliability, availability, and scalability to the world of message queuing. In addition, AQ supports multiple communication channels, including the Internet, to cater to the needs of e-businesses.

Advanced Queuing provides database-integrated message queuing functionality and asynchronous communication needed for application integration. It leverages the functions of the Oracle database so that messages can be stored persistently as well as propagated between queues on different machines and databases. Since Oracle Advanced Queuing is implemented in database tables, all the operational benefits of high availability, scalability, and reliability are applicable to queue data. Standard database features such as recovery, restart, and security are supported in Advanced Queuing, and queue tables can be imported and exported.

In an integrated environment, messages travel between the Oracle database server and the applications and users. Using Oracle Net Services (formerly Net8), messages are exchanged between a client and the Oracle database server or between two Oracle databases. Oracle Net Services also propagates messages from one Oracle queue to another. Advanced Queuing operations can also be performed over the Internet using transport protocols such as HTTP, HTTPS, or SMTP. In this case, the client, a user or Internet application, produces structured XML messages. During propagation over the Internet, Oracle servers communicate using structured XML.

### **ASYNCHRONOUS APPLICATION INTEGRATION**

AQ enables applications to communicate asynchronously. It offers multiple ways for applications to produce (enqueue) messages and consume (dequeue) messages.

Producer applications can enqueue a message for post consumption. This allows for a window of execution. With this feature, messages will be visible to the consumer application after the specified delay. Messages can also be created with an expiration time. In this case if the message is not consumed before its expiration the message is moved to the exception queue and no longer available for consumption. It is possible to specify the priority of the enqueued message. An enqueued message can also have its exact position in the queue specified.

On the consuming end, a consumer application has various options to consume the message. They can process the messages as they arrive, on a first come first server basis, on a priority-basis, or by position . A DEQUEUE can be issued against an empty queue. To avoid polling for the arrival of a new message, a wait time can be specified for the arrival of a message, keeping control of the application. If the message still does not arrive after the specified wait time control is passed back to the application.

The consumer application also has the listen capability. The listen call is a blocking call that can be used to wait for messages on multiple queues. It can be used by a gateway application to monitor a set of queues. An application can also use it to wait for messages on a list of subscriptions. If the listen returns successfully, a dequeue must be used to retrieve the message.

Alternatively, the consumer application can choose to be notified when a message arrives. The asynchronous notification feature allows the consumer application to receive notification of a message of interest. It can be used by the consumer application to monitor multiple subscriptions. The consumer application does not have to be connected to the database to receive notifications regarding its subscriptions. These notifications can be an OCI callback function, a PL/SQL functions, or even an email.

## **EXTENSIBLE INTEGRATION ARCHITECTURE**

AQ offers an extensible architecture for integrating distributed applications. It offers multiple models of communication between applications, dynamic addition of subscribers and two models for subscribers to receive messages.

The two models of communication are: point-to-multipoint and publish/subscribe.

In the point-to-multipoint model, the producer application can control who gets the message by specifying the recipient list for the message.

In the publish/subscribe model, the producer application has no control over who gets the message. Publisher and subscriber applications communicate via queues and need not know about each other. It is the equivalent of a radio station not knowing exactly who the audience is for a given program. The dequeuers are subscribers to multiconsumer queues. The publish/subscribe paradigm offers an extensible way to integrate additional applications in the future.

Subscribers can receive messages based on two criteria: content-based and subject-based. AQ also offers unique content-based subscriptions. The subscriber application will receive a message if the message content has the specified condition of the subscriber. For subscribers receiving messages based on the subject, the subject of the message will have to contain the specified condition of the subscriber.

Due to the extensible architecture of AQ, applications can be added to a queue without affecting the other applications in the queue.

## **HETEROGENEOUS DATA INTEGRATION**

One of the limitations of most of the messaging systems is the limited type system. With AQ, messaging applications can use the extensive type support offered by the Oracle database. This includes support for Oracle object types and the XML Type supported in Oracle9i. The data model used for messaging operations can be the same as that of the application to be integrated.

Generally applications to be integrated are designed independently of each other. So, the messages they understand are different from each other. To integrate these applications, messages have to be transformed. There are various existing solutions to handle these transformations. AQ provides a transformation infrastructure that can be used to plug in transformation functionality from Oracle Application Interconnect or other third-party solutions such as Mercator without losing AQ functionality. With this feature, customers can define transformation mappings from one data model to another. Transformations can be specified as PL/SQL call back functions, which are applied at enqueue, dequeue, or propagation of messages. These PL/SQL callback functions can call third-party functions implemented in C, Java, or PL/SQL. XSLT transformations can also be specified for XML messages.

## **APPLICATION LOCATION TRANSPARENCY**

The applications to be integrated need not be co-located. They may run on different Oracle databases. Messages can be automatically propagated between queues which may be on the same or different Oracle databases. The destination queue is yet another subscriber to the source queue. AQ propagation does not use distributed two-phase commit. Hence, it does not have the in-doubt transactions in case of failures and is also network efficient. AQ offers guaranteed and fail-safe propagation. Propagation is also tunable according to application needs. In Oracle9i, propagation can be done over the Internet using a new XML based API, IDAP.

In Oracle9i, AQ is also integrated with Oracle Internet Directory (OID). Queue and event meta-data can now be stored and looked up from LDAP (OID).

## **UNIQUE MESSAGE MANAGEMENT BENEFITS**

Providing message queuing from the database offers unique benefits. It brings transactional behavior with very high reliability to the messaging operations. In case of failures, messaging operations are recovered in the same manner as other database operations.

With AQ, messaging and database operations can be performed in the same transaction. They can use the same security model for messaging and database operations. This is a unique benefit that tremendously reduces application complexity. Application developers do not need to bother with different resources

for transaction management, different security schemes, and different data models.

In many situations, for example, when sending financial information, the messaging operations require not only guaranteed delivery but also reliable auditing. With AQ, messaging operations are automatically audited. An entire message history can be queried using a SQL view. This history can also be used for extracting tracking information and business intelligence.

### **UNIQUE FEATURES FOR E-BUSINESS**

Businesses interact with their customers and partners over the Internet. In Oracle9i, AQ operations - enqueue, dequeue, notifications, propagation - can be performed over the Internet. Now, a very simple and high performance web application can directly place an order securely over the Internet. This new Internet propagation feature enables integration of applications of business partners over the Internet. Oracle9i security features ensure that only authorized users can access critical data. An email notification feature sends an email notifications for high-priority activities. A new transformation feature can be used for verification and transformation of incoming and outgoing business messages.

E-business integration is incomplete without the integration of legacy applications. To integrate with IBM mainframe applications, messages can be automatically propagated to and from AQ to MQ Series. Similarly for integrating with Tibco-based messaging applications, messages can be automatically propagated to and from AQ and Tibco Rendezvous.

### **MANAGEMENT WITH ORACLE ENTERPRISE MANAGER**

Oracle Enterprise Manager (OEM) can manage AQ. AQ administrative functionality has been provided through the OEM console. The OEM console can be used to create queue tables, create queues, browse through AQ messages, archive or purge AQ messages, add AQ subscribers, and manage propagation. The OEM console also shows the topology for the propagation of messages between queues at the database and queue level.

The OEM Diagnostics and tuning pack supports alerts and monitoring for AQ queues. Alerts can be sent when the number of messages for a particular subscriber exceeds a threshold. Alerts can be sent when there is an error in propagation. In addition, queues can be monitored for a number of messages in ready state or a number of messages per subscriber, etc.

### **ADVANCED REPLICATION**

Three of the challenges faced by company XYZ can be resolved by using Advanced Replication. Replication is the process of copying and maintaining database objects, such as tables, in multiple databases that make up a distributed database system. The same data is available at multiple locations. For example, the

EMP table may be available at db1, db2, and db3. It provides users with fast, local access to shared data, and protects availability of applications because alternate data access options exist. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations.

As with company XYZ many of the Internet vendors need to have their web site available 24 x 7. Large-scale Internet vendors are increasingly turning to Oracle Advanced Replication to provide the long-term scalability, availability and performance for their e-business sites. Replication provides the perfect solution for this situation. The primary objective of replication in most environments is availability: keeping the data available to users without interruption of service. Replication accomplishes this by peer-to-peer synchronization, a replication type referred to as Multimaster replication, where two or more sites, have replicas of the same data.

To address Company XYZ's requirement of periodically disseminating its product catalog to its regional offices and to enable its sales force to place orders from customer sites, Oracle offers a replication type referred to as Materialized view replication. In both these cases the goal is not to replicate all the data at the master site, but rather a portion of it. A materialized view contains a complete or partial copy of a target master from a single point in time.

Replication can also improve the performance of company XYZ's web site. By locally replicating remote tables that are frequently queried by local users, for example the inventory table, the amount of data going across the network is greatly reduced. By having the local users access the local copies instead of one central copy, the distributed database does not need to send information across a network repeatedly, thus helping to maximize the performance of the database application. Either multimaster replication or materialized views can be used in this case. The replication type depends on the business needs such as how fast changes need to be propagated to the other sites.

Multimaster replication and Materialized views are covered in more detail in the following sections.

## **MULTIMASTER REPLICATION**

Multimaster replication is two or more sites, referred to as master sites, working together to maintain exact copies of the same tables. These replicated tables can be updated at all sites. Multimaster replication provides complete replicas of each replicated table at each of the master sites. Applications can update any replicated table at any site in a multimaster configuration. Each site automatically communicates with the other sites to maintain the data of all table replicas, and ensure global transaction consistency and data integrity.

Multimaster replication can be useful to protect the availability of a mission critical database. For example, company XYZ can use multimaster replication to

replicate all its data critical for their web site to establish a failover site should the primary site become unavailable due to system or network outages, thus meeting the requirement of 24 x 7 availability. In contrast with Oracle's standby database feature, the replicated site can also serve as a fully functional database to support application access while the primary site is operational.

To keep all locations as up to date as possible, Replication supports both real-time and near real-time replication. With asynchronous multimaster replication(near real-time), each change made by a user is captured and stored locally, and then later forwarded and applied at each remote location. Synchronous replication(real-time), where changes applied to any master table are propagated and applied directly to all other master tables is also available.

Multimaster replication uses deferred remote procedure calls (RPCs) as the underlying transport mechanism to propagate and apply changes. Changes to tables in a multimaster configuration are applied in a transactionally consistent manner to ensure data and referential consistency. Changes are propagated either continuously, at a user specified time, at regular intervals or on-demand. With asynchronous replication failures of any one master site will not block propagation of changes between other master sites. If a remote system is unavailable, the deferred RPCs propagating changes to that system remain in their local queue for later execution.

If two sites update the same data element within the same replication interval, an update conflict will occur. To ensure data integrity, Oracle has the ability to detect and resolve conflicts, so that the data element has the same value at every site. Users may choose one of Oracle's many built-in conflict resolution routines, such as "latest timestamp" or "site-priority", to resolve potential conflicts. Users can select different methods for different tables, or even different groups of columns within a table. Users can also create their own routines to employ resolution rules tailored to their particular business needs. Replication has been carefully designed to provide the optimal performance required by users of multimaster replication, while still ensuring data integrity throughout the replicated environment.

## **MATERIALIZED VIEWS**

A materialized view is an updatable or read-only copy of a master table or portion of a master table from a single point in time.

Read-only materialized views can be used for company XYZ to periodically propagate the updated product catalog to the various sales offices since the product catalog will not be updated at the sales offices but rather only at the master site.

Updatable materialized views allows users to insert, update, and delete rows of the target master table or master materialized view by performing these operations on the materialized view.

In Oracle9i, multitier materialized views were introduced. A multitier updatable materialized view is one that is based on an updatable materialized view, instead of on a master table. Multitier replication provides increased flexibility of design for a distributed application. Using multitier materialized views, applications can manage multilevel data subsets where there is no direct connection between levels.

In contrast to multimaster replication where tables are continuously updated by other master sites, materialized views are periodically updated from one or more masters through individual batch updates, known as a refreshes, from a single master site or master materialized view site. There are three types of refresh available:

- Fast refresh uses materialized view logs to update only the rows that have changed since the last refresh.
- Complete refresh updates the entire materialized view.
- Force refresh performs a fast refresh when possible. When a fast refresh is not possible, force refresh performs a complete refresh.

#### **DATA SUBSETTING**

Materialized views allow data replication based on column and row-level subsetting. Data subsetting enables replication of information that pertains only to a particular site. For example, company XYZ's regional sales office in Japan, might only replicate customer data for Japan, from the customer table, thereby cutting down on unnecessary network traffic.

Data subsetting is defined when a materialized view is created. By using a WHERE clause, a subset of the rows of the master table can be obtained. Using deployment templates, column subsetting can be accomplished by specifying particular columns in the SELECT statement of the materialized view.

#### **DEPLOYMENT TEMPLATES**

Deployment templates simplify the task of deploying and maintaining many remote materialized view sites. A collection of materialized view definitions can be created locally at the master site. Deployment templates can be used to quickly and easily deploy materialized view environments to support sales force automation and other mass deployment environments. By using parameters in the materialized view definitions custom data sets can be created for individual users without changing the deployment template. This technology is great for rolling out a database infrastructure to hundreds or thousands of users.

#### **DISCONNECTED COMPUTING**

Materialized views do not require a dedicated network connection. This is an ideal solution for sales applications running on a laptop. For example, in the case of company XYZ when a member of the sales force has completed the day's orders,

the salesperson can simply dial up to the network and refresh the database, thus transferring the orders to the main office, and obtaining the most current information from the master site.

### **CONFLICT RESOLUTION**

Replication conflicts can occur in a replication environment that permits concurrent updates to the same data at multiple sites. For example, when two transactions originating from different sites update the same row at nearly the same time, a conflict can occur.

Asynchronous multimaster and updatable materialized view replication environments must address the possibility of replication conflicts. The system data does not converge until the conflict is resolved in some way. When data conflicts occur, a mechanism is required to ensure that the conflict is resolved in accordance with your business rules and to ensure that the data converges correctly at all sites.

In addition to logging any conflicts that may occur, Oracle Replication offers a variety of prebuilt conflict resolution methods to define a conflict resolution system for your database that resolves conflicts in accordance with your business rules. Users can also write their own conflict resolution routines.

### **REPLICATION MANAGEMENT TOOL IN ORACLE ENTERPRISE MANAGER**

Oracle's Enterprise Manager(EM) provides a graphical user interface (GUI) for setting up, managing, and monitoring a replication environment. It includes wizards that guide you through many important operations and reports for monitoring. It can be used to manage both multimaster and materialized view replication environments. The EM console also shows the topology for the replication environment.

### **GENERIC CONNECTIVITY AND TRANSPARENT GATEWAYS**

As seen with company XYZ, many companies are faced with the challenge to integrate data residing in disparate systems. Heterogeneous data access is a problem that affects a lot of companies. Many of them, like company XYZ, run several different database systems. Each of these systems stores data and has a set of applications that run against it. Consolidation of this data in one database system is often hard - in large part due to the fact that many of the applications that run against one database may not have an equivalent that runs against another. Until such time as migration to one consolidated database system is made feasible, it is necessary for the various heterogeneous database systems to interoperate. The challenge is to quickly, efficiently, and economically deploy data that may exist on many disparate systems through a single application, providing a comprehensive view of the data, regardless of the database or operating system. Oracle offers two connectivity solutions that will enable company XYZ to

seamlessly integrate the different systems and provide a consolidated view of the company as a whole. They are: Generic Connectivity and Oracle Transparent Gateways.

### **HETEROGENEOUS TRANSPARENCY**

Both Generic Connectivity and Oracle Transparent Gateways provide the ability to transparently access data in non-Oracle systems from an Oracle environment. As with an Oracle distributed database environment, location transparency can be extended to objects residing in non-Oracle systems as well. Therefore users can create synonyms for the objects in the non-Oracle and refer to them without having to specify its physical location. This transparency eliminates the need for application developers to customize their applications to access data from different non-Oracle systems, thus decreasing development efforts and increasing the mobility of the application. Instead of requiring applications to interoperate with non-Oracle systems using their native interfaces (which can result in intensive application-side processing), applications can be built upon a consistent Oracle interface for both Oracle and non-Oracle systems.

Consider the scenario where a customer uses Oracle Transparent Gateways to access heterogeneously stored data with the plan to migrate the heterogeneous data to an Oracle system. If a database application is developed to interoperate with both an Oracle and a non-Oracle system using their native interfaces, once the migration to a homogeneous Oracle environment is complete, the database application would need to be altered to operate in that environment.

### **CONNECTING DISPARATE SYSTEMS**

Although the user interfaces for different non-Oracle systems based on SQL standards may appear to work identically, there may be subtle (and not so subtle) differences between these non-Oracle systems. These differences may prevent the disparate systems from interoperating effectively.

For smooth interoperability between disparate systems, SQL translations, data dictionary translations and data type translations are required, even if the non-Oracle systems are based on SQL standards. Both Generic Connectivity and Oracle Transparent Gateways have the ability to translate one system's dialect to another.

### **SQL TRANSLATIONS**

Even though a relational data store may be based on SQL standards, there may be subtle differences between manufacturers in the implementation. For example, if you wanted the following result set to be in uppercase letters, you would execute the following SQL statement in an Oracle environment:

```
SELECT TO_UPPER(ename) FROM emp;
```

In a non-Oracle environment, however, you might execute the following SQL statement to retrieve the same results:

```
SELECT UPPERCASE(ename) FROM emp;
```

The heterogeneous solution should automatically translate the dialect of the foreign system to that of the local system (where the transaction originated), eliminating the need for the user to utilize more than one system's dialect. This solution should also occur transparently.

#### **DATA DICTIONARY TRANSLATIONS**

Metadata, or information about a database environment, is extremely useful to DBAs managing a database environment. Different manufacturers have their own methods of storing this data in a data dictionary and displaying this data. A mechanism is required that would allow a query of the metadata at a remote disparate system to be displayed in the format of the local system.

For example, a DBA might issue the following SQL statement to view all tables in an Oracle database:

```
SELECT * FROM sys.dba_objects WHERE object_type = 'TABLE';
```

In a non-Oracle environment, the following SELECT statement might yield the same results:

```
SELECT * FROM catalog_objects  
WHERE object_name LIKE '%EP%'  
AND object_type = 'TABLE';
```

Just as for SQL translations, the heterogeneous solution should automatically translate a data dictionary query in the dialect of the local database to that of the target remote database. This translation involves rewriting the local SELECT statement into a query that will produce the same results by querying the remote system.

#### **DATATYPE TRANSLATIONS**

The final area that affects the interoperation of disparate systems involves translating one manufacturer's datatype to another manufacturer's. For example, the heterogeneous solution should transparently handle translating a DB2 PACKED DECIMAL datatype to an Oracle NUMBER datatype.

When a DESCRIBE statement is issued for a particular remote object, it should be described with the datatypes of the local database.

#### **HETEROGENEOUS CONNECTIVITY PROCESS ARCHITECTURE**

Gateway technology is composed of two parts: a component that has the generic technology to connect to a non-Oracle system, which is common to all the non-

Oracle systems, called Heterogeneous Services (HS) and a component that is target specific, called an Agent. Heterogeneous Services in conjunction with the Agent enables transparent access to non-Oracle systems from an Oracle environment.

#### **HETEROGENEOUS SERVICES(HS)**

Heterogeneous Services provides the generic technology for connecting to non-Oracle systems and is the processing power of both solutions. Generic Connectivity and Oracle Transparent Gateways are based on Heterogeneous Services. As an integrated component of the database, Heterogeneous Services can exploit features of the database, such as the powerful SQL parsing and distributed optimization capabilities. As of Oracle9iAS, Heterogeneous Services is also part of Oracle9iAS Database Cache thereby enabling connection to the non-Oracle system from the middle tier using an agent.

Heterogeneous Services extend the Oracle SQL engine to recognize the SQL and procedural capabilities of the remote non-Oracle system and the mappings required to obtain necessary data dictionary information. Heterogeneous Services provides two types of translations: the ability to translate Oracle SQL into the proper dialect of the non-Oracle system as well as data dictionary translations which displays the metadata of the non-Oracle system in the local format. For situations where no translations are available, native SQL can be issued to the non-Oracle system using the pass-through feature of Heterogeneous Services.

Heterogeneous Services also makes it possible for non-Oracle systems to be integrated into Oracle database server transactions and sessions. When a non-Oracle system is accessed for the first time over a database link within an Oracle user session, an authenticated session in the non-Oracle system is transparently set up. At the end of the Oracle user session, the authenticated session in the non-Oracle database system is transparently closed. Additionally, one or more non-Oracle systems can participate in an Oracle distributed transaction. When an application commits a transaction, Oracle's two-phase commit protocol accesses the non-Oracle database system to coordinate transparently the distributed transaction. Even in those cases where the non-Oracle system does not support all aspects of Oracle two-phase commit protocol, Oracle can (with some limitations) support distributed transactions with the non-Oracle system.

The pass-through SQL feature of HS provides the facility to issue native SQL directly against a non-Oracle system without being interpreted. This flexibility enables you to execute functions or procedures on the non-Oracle system that are not supported by the Oracle Transparent Gateway.

#### **AGENT**

An agent is the process through which an Oracle server connects to a non-Oracle system. The agent process consists of two components. These are agent generic

code and a non-Oracle system-specific driver. An agent exists primarily to isolate the Oracle database server from third-party code. In order for a process to access the non-Oracle system, the non-Oracle system client libraries have to be linked into it. In the absence of the agent process, these libraries would have to be directly linked into the Oracle database and problems in this code could cause the Oracle server to go down. Having an agent process isolates the Oracle server from any problems in third-party code so that even if a fatal error takes place, only the agent process will end.

An agent can reside in the following places:

- On the same machine as the non-Oracle system
- On the same machine as the Oracle server
- On a machine different from either of these two

The capabilities, SQL mappings, datatype conversions, and interface to the remote non-Oracle system are contained in the Agent. The agent interacts with Heterogeneous Services to provide the transparent connectivity between Oracle and non-Oracle systems. There are two types of agents: HS agents and Transparent Gateway agents.

There are two types of HS agents: HS ODBC which uses an ODBC driver to talk with the non-Oracle system and HS OLEDB which uses an OLEDB driver. HS agents are part of the Oracle9i and Oracle9iAS, so they are installed by default with these products. Generic Connectivity use these agents to connect to the non-Oracle systems.

Transparent Gateway agents are part of Oracle9i and Oracle9iAS however they are licensed separately. Unlike the HS agents which has to be on the same machine as the Oracle database, the Transparent Gateway agents can be installed on any machine. They can be on the same machine as the Oracle database or on the same machine as the non-Oracle system or on a third machine as a stand alone. Each configuration has its advantages and disadvantages. The issues to consider when determining where to install these agents are network traffic, operating system platform availability, hardware resources and storage. Oracle Transparent Gateways use these agents to connect to the non-Oracle systems.

#### **GENERIC CONNECTIVITY**

Generic Connectivity is a feature of Oracle9i and Oracle 9iAS. It is a generic solution that uses an ODBC or OLEDB driver to access any ODBC or OLEDB compliant non-Oracle system. It addresses the needs of data access to many data stores for which Oracle does not have a gateway solution. This feature enables transparent connectivity using industry standards such as ODBC and OLEDB. Generic connectivity makes it possible to access low-end data stores such as Foxpro, Access, dBase and non-relational targets like Excel.

## **ORACLE TRANSPARENT GATEWAYS**

In contrast to Generic Connectivity which is a generic solution, Oracle Transparent Gateways are tailored solutions, specifically coded for the non-Oracle system. They provide an optimized solution, with more functionality and better performance than Generic Connectivity. Generic Connectivity relies on industry standards, whereas Oracle Transparent Gateways accesses the non-Oracle systems using their native interface. The Transparent Gateways are also end-to-end certified. Oracle has Transparent Gateways to many sources, Sybase, DB2, Informix, Microsoft SQL Server, Ingres, Teradata, to name a few.

## **CONCLUSION**

Using company XYZ, as an example it is clear that with today's rapidly changing business environment companies must operate in a distributed environment. Looking at the needs that arise for companies in such situations it is obvious that Oracle offers a complete and robust solution to address each of these needs in a reliable, and secure environment.

Distributed SQL is used to transparently communicate between the Oracle databases. Generic Connectivity and Transparent Gateways extend that capability to non-Oracle systems. Advanced Queuing extends the communication capability beyond just databases to applications. Additionally advanced replication significantly improves the availability of these applications by, load balancing as well as disconnected computing.



ORACLE'S SOLUTIONS FOR THE DISTRIBUTED ENVIRONMENT

06 2001

Author:

Contributing Authors:

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

[www.oracle.com](http://www.oracle.com)

Oracle is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation. All other product and service names mentioned may be trademarks of their respective owners.

Copyright © 2001 Oracle Corporation

All rights reserved.