Advanced Queuing in Oracle

By Ritu Singal

ritu_singal@delhi.tcs.co.in

TATA Consultancy Services

Feb 2003

Introduction

Enterprises are investing heavily in their information technology infrastructure because they realize huge benefits in terms of lower business costs, better information and better communications. They have developed a variety of autonomous and distributed applications to automate business process, 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.

Advanced Queuing (AQ) is the message queuing functionality of the Oracle database that satisfies the above needs. Traditional message-oriented middleware address only a sub-set of these needs. Oracle AQ provides message queuing as an integrated part of the Oracle server. Oracle AQ provides this functionality by integrating the queuing system with the database itself

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.

While using Oracle Advanced Queuing we do not have to install additional middle-ware dealing with inter-process communication and transaction monitoring. We can directly use an existing and well-known database and can benefit from given functionalities like online backup or transaction processing. Alternatively other simple and non-queue based messaging techniques can be used like the Java RMI, Or more complex approaches like CORBA.

Messages represent critical business events. Consequently, the queuing technology has to have the same functionality and operational characteristics as the infrastructure for mission critical applications:

- The creation and consumption of messages must handled with the highest level of integrity.
- Messages must be protected against system failures of various kinds, including disasters.
- Message content needs to be accessible both for operational management purposes and for business decisions. This retrieval must be flexible and efficient.

Transaction processing systems have the desired characteristics to meet the first two requirements. In fact, many TP monitors have included queuing subsystems.

Database technology, with powerful query capability, can meet the third requirement in addition to the integrity and durability requirements.

Conventional file system-based messaging systems do not fully meet these requirements, especially for mission critical environments.

Because database technology matches up well with these requirements, a new generation of messaging products, based on database systems, is emerging.

There are two basic approaches:

- Create a queuing system as an application to the database, i.e., use the database "as is" for storage of messages.
- Build queuing into the database infrastructure.

The use of the database "as is" provides significant benefits:

- Transaction support without necessarily requiring the overhead of two-phase commits.
- Integrated storage management.
- Recovery management.
- Restart management including disaster recovery.
- Full type and query support.
- Retention of messages allowing automated tracking.
- "Message warehouse" support.

The drawback of this approach is that databases are not inherently designed to be message queuing systems. This puts fundamental scalability and functionality limitations on messaging systems that use the database "as is."

The following list specifies the demands that a message queuing solution places on a database that cannot be handled without changing the database. It also describes how these requirements are addressed by **Oracle Advanced Queuing (AQ),** a part of the Oracle8TM object relational database management system.

- ***Message consumers may need to be informed immediately about the arrival of a new message.***

If an application requests a message and no message exists, the database will return immediately with a 'Not Found' condition. This forces the application to 'poll' the database/queue for arrival of new messages. Polling is not scalable and its inherent latency is often not acceptable. AQ allows applications to specify how long they are willing to wait. AQ informs waiting applications as soon as a new message becomes visible, e.g., posting is done at the end of the commit process of the transaction that created the message.

- ***From a set of messages, consumers may like to see the highest priority message that has not yet been selected for processing.***

  Messages are typically ordered by importance, e.g., priority and/or creation time. To retrieve messages according to the desired order, a SELECT ORDERED BY SQL statement has to be used. While a message is being processed, it must be locked to ensure transactional semantics. Since all consumers using this queue will typically get the rows in the same order, only one application at a time can be processing a message. To solve this, Oracle internally added the ability to skip those messages that are already being processed by other applications. This enables the concurrent consumption of messages from multiple applications.

- ***Statistics are required to provide information about the current status of the queue, e.g., the number of messages in a queue or the average response time of the currently active messages.***

  Statistics create "hot spots" that cannot be handled with the traditional locking technology. Oracle AQ creates a work-list of all message creations and consumption in a transaction. After commit completion, the statistics are updated based on the information in the work-list. This is a typical escrow approach, which also deals with application and system failures.

- ***Message management requires modifications to the index technology.***

  Messages are frequently added and deleted. This tends to create deep index structures even for relatively small numbers of records. The index algorithms have been enhanced to properly handle this problem.

- ***Messages need to be propagated between queues.***
  Executing the propagation as part of the database server removes the need for process context switching, which would occur between the database server and propagator.

- ***It is necessary to keep track of the number of attempts to process a message-and the count has to be retained through application and database failures. This causes problems with standard database techniques, because the "increase failure count" must be committed even if the rest of the transaction is rolled back.***

  If a transaction starts with the consumption of a message, which is almost always the case, Oracle AQ creates an internal counter. If the transaction rolls back, the database is reset and the 'attempt' count is updated and committed independently. If the attempt count reaches the user-defined maximum value, the message is moved to an exception queue.

- ***Messages may need to be created even if a transaction is rolled back.***
  Applications may be unable to process messages, however, the application may need to create a message about this condition. Oracle AQ allows applications to

create a message that is 'outside' of the current transaction and immediately visible. A following abort request will rollback all the changes, however, the message will still be delivered.

Key features for Advanced Queues

Following are some of the key features offered by Advanced Queuing technique.

- **Asynchronous application integration**

AQ enables applications to communicate asynchronously. It offers multiple ways for applications to produce (enqueue) message and consume(dequeue) messages. Producer application can enqueue a message for post consumption. With this feature, messages will be visible to the consumer application after the specified delay. Message can be created with an expiration time. If the message is not consumed before its expiry the message is moved to the exception queue. On the consuming end, consumer application has various mechanisms to consume the message. They can process as the messages as they arrive. For that purpose, they can use wait for the messages to arrive. Alternatively, it can be notified when the messages arrive. These notifications can be an OCI callback function, a PL/SQL functions, or even an email. AQ also offers FIFO and priority based ordering of messages.

- **Extensible Integration Architecture**

AQ offers an extensible architecture for integrating distributed applications. It offers multiple models of communication - point-to-multipoint and publish/subscribe. In the point-to-multipoint model, the producer application specifies the recipient list for the message. The publish/subscribe paradigm offers an extensible way to integrate additional applications in the future. Publisher and subscriber applications communicate via queues and need not know about each other. AQ also offers unique content-based subscriptions. Subscriber application can specify interest based on message content.
AQ offers extensible way to handle exceptions. The erroneous message is automatically moved to an exception queue. These exceptions can be handled separately. Exceptions can occur when the message is expired or message can not be successfully dequeued.

- **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 XMLType supported in Oracle9i. The data model used for messaging operations can be the same as that of the application to be integrated.

  In Oracle9i, applications with different data models can be integrated using the transformation feature. With this feature, customers can define transformation mappings from one data model to another. Later, they can use this mapping, while performing various AQ operations.

- **Application Location Transparency**

  The integrated applications 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 Oracle*9*i, propagation can done over the Internet.
  In Oracle*9i*, 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 bother about 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 Oracle*9i*, AQ operations - enqueue, dequeue, notifications, propagation - can be performed over the Internet. Now, a very simple and highly 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. 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 queuetables, 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.

The scenario given now tells how usage of advanced queues eliminated the inherent latency of batch processing, thus giving an online approach.

Batch to online processing using AQ

ANZ INSURANCE
The ANZ Insurance is one of the largest insurance company.

IT ENVIRONMENT
The ANZ Insurance has two applications:

1. Front office *customer service* application used by

- Sales people to sell new policies to customers.
- Customer service representatives to change customer policies and provide account information.
- Back office *billing/accounting* application used to send bills to customers and maintain their account status

- Both are custom applications running on Oracle databases. The two applications are owned and maintained by the sales and finance departments respectively. The applications are modified independent of each other.

  Usage:
  - Whenever a sales person sells a new policy to a customer, the information is captured locally in the customer service database and also passed on to the billing application.
  - Frequently the sales person also needs to know a customer's account status to offer them suitable discounts.
  - The customer service representative also needs a customers account status to answer their queries.

Prior Solution

1. Classical *file transfer based batch processing* was used to update the billing database with new information from the customer service database. The extract and load program was run at the end of the day.

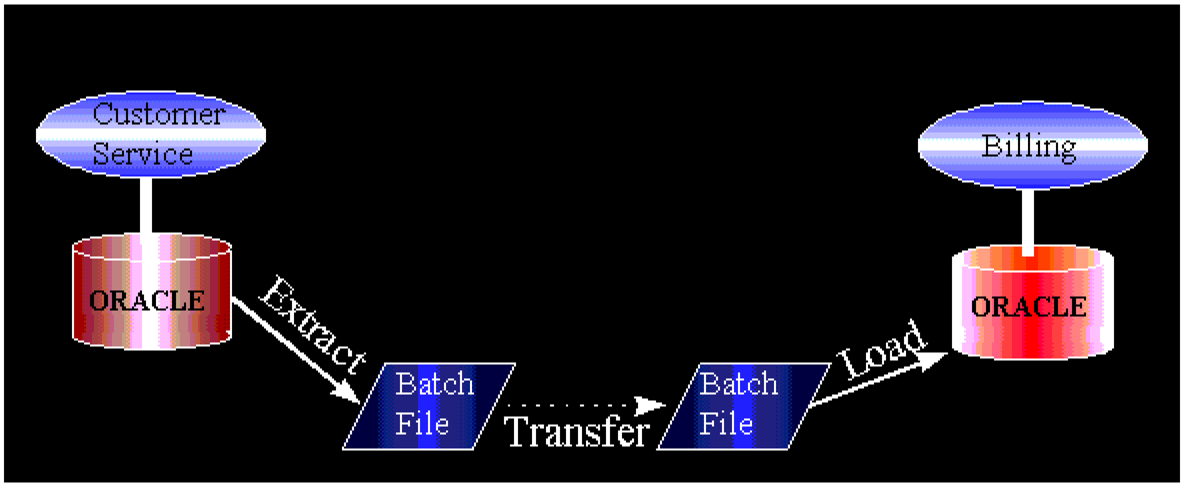

Figure 1: Batch processing using file transfers.

1. Sales people and customer service representatives used remote data access and reporting tools to retrieve billing and account information.

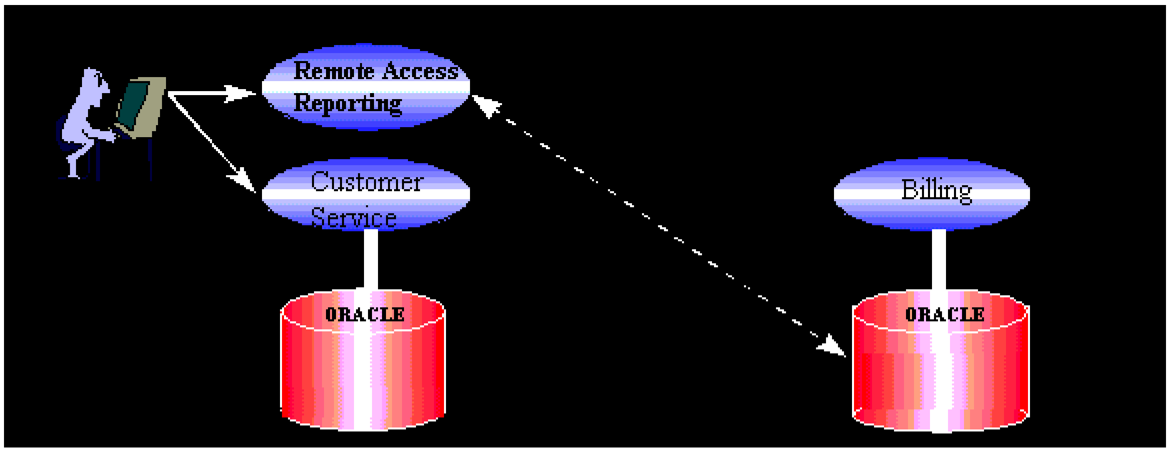

Fig2: Remote access to billing and accounting information.

PROBLEM

1. As a result of the inherent latency in batch processing, sales and customer service did not have access to the most current account and billing information. Access to the latest information was a key business requirement.
2. Batch processing using file transfer is cumbersome, time consuming and error prone. Before the loading phase, the batch file had to be validated. Errors that were detected had to be captured in a separate error file which was returned.
3. Due to a lack of transaction support, any failures during the load or extract phase resulted in the programs having to be rerun.
4. The remote access and reporting tools used by sales and customer service went directly to the billing database a major security concern. Furthermore these direct queries could interfere with billing application, adversely affecting its performance.

As already discussed database “as is” approach puts fundamental scalability and functionality limitations. And is therefore not the best approach.

AQ BASED SOLUTION

The customer was able to address all the above problems through a message based solution using Oracle8 AQ. Messaging enabled the integration of the applications in a loosely coupled manner - eliminating the latency of batch processing, without the tight dependency of synchronous communication.

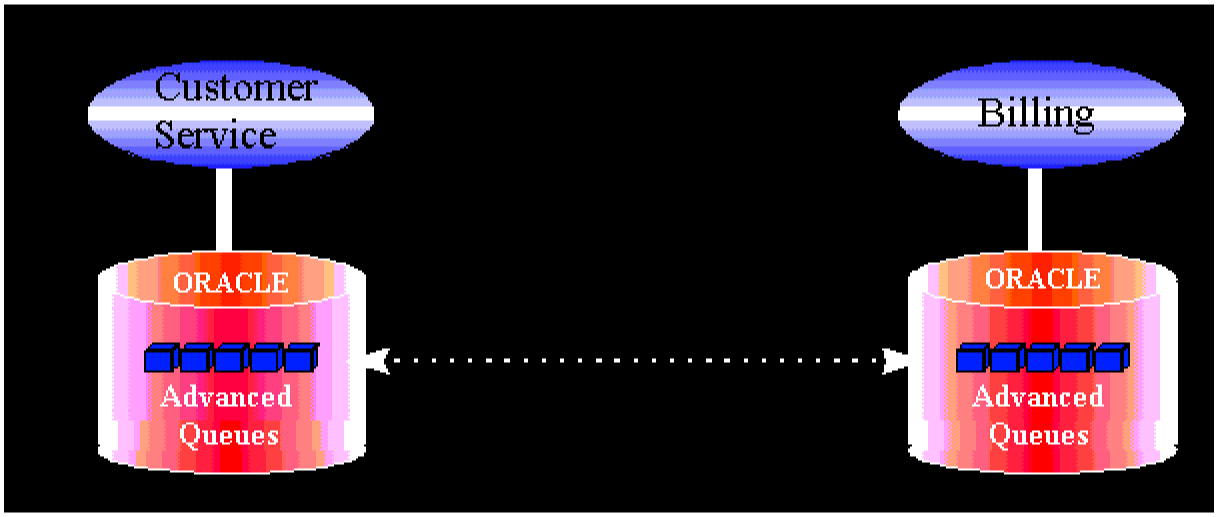

Figure 3: Messaging solution-using AQ.

ARCHITECTURE OUTLINE

AQ queues were setup in both the databases. The AQ propagator was used to transparently move messages between the queues. Job queues were used to process the messages.

Four queues were setup in each of the databases.

- Two sets of queues are used to handle new policies and policy changes.
- Two sets are used to retrieve customer billing and account information.

Handling new policies

Two queues are used for this purpose - one to handle new policies and policy updates and the other to handle rejected policies.

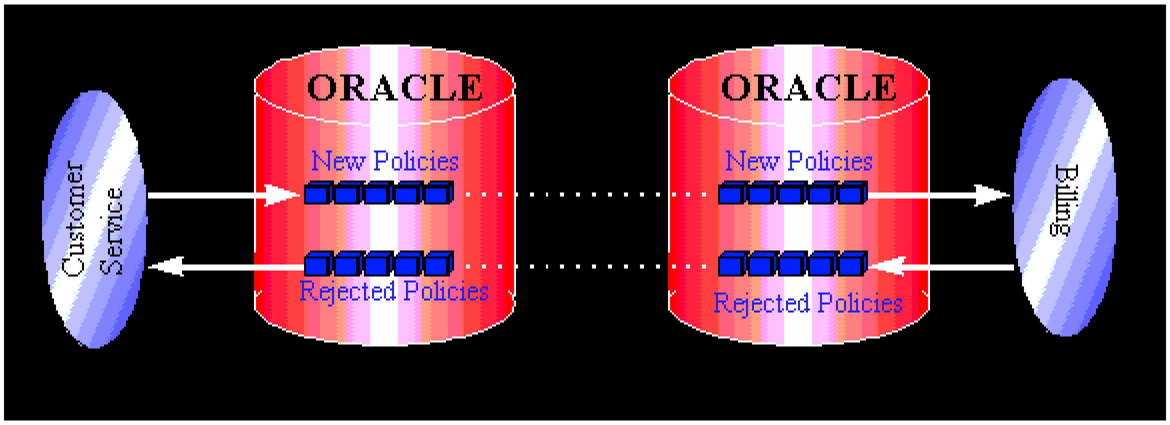

Figure 4: Two sets of queues are used to synchronize the two databases.

The flow of data is as follows:

- New policies and policy updates are enqueued into the local "new policy" queue by the customer service application.
- The AQ propagator transfer the messages into the "new policy" queue in the billing database.
- The billing application processes validates the new policies and updates the database appropriately. The rejected policies are enqueued into the local rejected policies queue.
- The customer service retrieves the rejected messages from its local "rejected policies" queue and processes them.

Access to account and billing information

Queues are also being used to provide account and billing information to sales and customer service representatives. This information is required synchronously, since the customer is waiting on-line. Messaging can be used to achieve this effect quite efficiently. Two sets of queues are used for this purpose as shown in Fig 5.

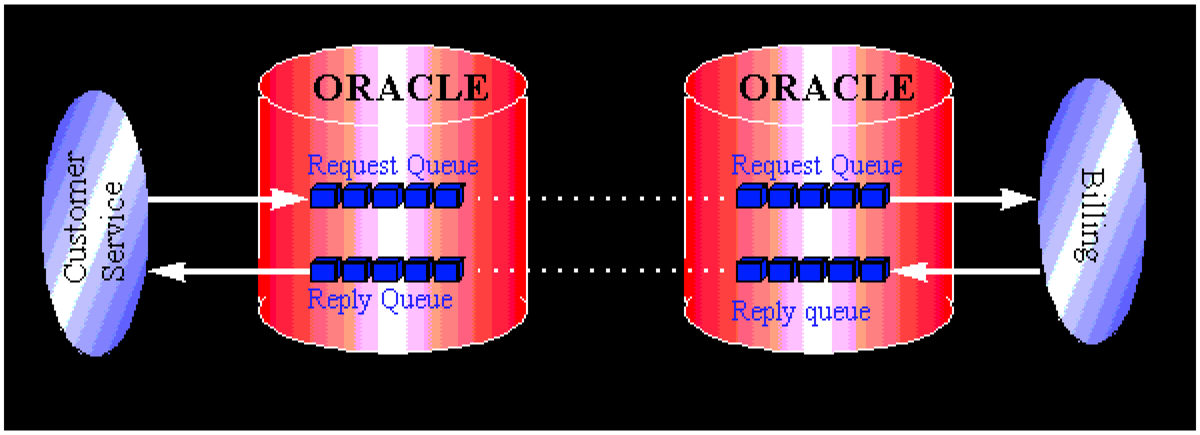

Figure 5: Using Request/Reply queues for synchronous communication

The customer service application places the request in the local request queue and then waits for the reply to show up in the local reply queue. The synchronous request/reply model of communication is achieved within the message queuing paradigm.

AQ FEATURES USED

- **Propagation:** The AQ Propagation facility is used to move messages between the queues in the two databases. The propagator servicing the "new policy queues" and the rejected policy queue is scheduled to run periodically, while the propagator serving the request/reply queues is scheduled to run with zero latency.
- **Retention:** All the queues in the customer service database are set up to retain messages even after they have propagated. This enables rejected policies and replies to be matched up with the original messages. In the absence of this feature the billing application would have to attach the original message to the replies. This increases the size of messages going back and forth, adversely affecting performance.
- **Correlation Identifier:** The correlation identifier is used to associate rejected policies and replies to with the original messages. The message id of the original message is stored in the correlation identifier field of the reply and is used to dequeue messages selectively**.**

The customer was able to transition from a cumbersome and error prone file transfer based batch processing system, to an simple, transactionally protected, online system. The messaging based system eliminated the inherent latency of batch processing, and also was to eliminate the need for special remote access and reporting tools.

Conclusion

For mission critical applications, messaging systems based on database technology are superior to those based on file systems. However, database technology needs to be modified to deal with the performance, scalability and functionality challenges of messaging.

Advanced queuing is the full-featured message queuing functionality of the Oracle database. The unique management functionality offered by AQ cannot be accomplished by merely combining a message-oriented middleware and a database. It has unique functionality such as Internet accesses tremendously simplifies e-business interaction.