5. Oracle Advanced Queuing

Contents:

Oracle AQ Concepts
Getting Started with Oracle AQ
Oracle AQ Nonprogram Elements
DBMS_AQ: Interfacing to Oracle AQ (Oracle8 only)
DBMS_AQADM: Performing AQ Administrative Tasks (Oracle8 only)
Oracle AQ Database Objects
Oracle AQ Examples

Oracle8 offers a facility (new to Oracle 8) called Oracle AQ (Oracle Advanced Queuing, referred to as AQ in this chapter) that will make it much easier for developers to build applications that require deferred execution of activity. Oracle is positioning Oracle AQ as an alternative to the queuing mechanisms of teleprocessing monitors and messaging interfaces. Oracle AQ will serve as a foundation technology for workflow management applications, both those delivered by Oracle Corporation itself and those implemented by third parties.

From a PL/SQL standpoint, Oracle AQ is made available through two packages: DBMS_AQADM and DBMS_AQ. The DBMS_AQADM package is the interface to the administrative tasks of Oracle AQ. These tasks include:

- Creating or dropping queue tables that contain one or more queues
- Creating, dropping, and altering queues, which are stored in a queue table
- Starting and stopping queues in accepting message creation or consumption

Most users of the Oracle AQ facility will not work with DBMS_AQADM. The DBA will most likely initialize all needed queue tables and queues. PL/SQL developers will instead work with the DBMS_AQ, whose tasks include:

- Creating a message to the specified queue
- Consuming a message from the specified queue

Most of the background information presented early in this chapter applies to both packages. Later, we'll provide individual discussions of the two packages and their programs.

Oracle AQ is full of features and offers tremendous flexibility in how you manipulate messages in queues. It also has only a handful of programs with which to administer and operate the queues. This relatively small number of procedures and functions can, nevertheless, be misleading. You will probably encounter lots of frustrating moments as

you come up to speed on Oracle AQ (I know I did).

Some of the frustration will arise from the way that Oracle allows you to modify the characteristics of queues, queue tables, and specific enqueue and dequeue operations by setting individual fields of PL/SQL records, which are then passed to the appropriate programs as arguments. If you are not comfortable with declaring and manipulating record structures in PL/SQL, you should read through *Chapter 9* of *Oracle PL/SQL Programming* (O'Reilly & Associates, 1997) before continuing with this chapter.

> **NOTE:** Oracle AQ first became available with Oracle 8.0.3. A number of Oracle AQ features described in this chapter were added in Oracle 8.0.4. These additions will be noted, but it is generally assumed in this chapter that you have installed and are using Oracle 8.0.4, which should be widely available by the time this book is published.

5.1 Oracle AQ Concepts

Oracle AQ provides message queuing as an integrated part of the Oracle server. It provides this functionality by integrating the queuing system with the database itself. Oracle Corporation is therefore now beginning to describe Oracle8 not only as an object-relational database, but also as a "message-enabled" database. Its intention is to free application developers from having to construct a messaging infrastructure (or rely on a third-party tool such as a TP monitor). Instead they can use AQ and devote their efforts to implementing their own specific business rules.

5.1.1 General Features

The following list summarizes Oracle's perspective on the features of the AQ facility that it offers to developers:

SQL-based access and management

> Messages are placed in normal rows in a database table. They can be queried using standard SQL. Thus, users can use SQL to access the message properties, the message history, and the payload. All available SQL technology, such as indexes, can be used to optimize the access to these messages.

Integrated database-level operational support

> All standard database features, such as recovery, restart, and enterprise manager, are supported. Oracle AQ queues are implemented in database tables; thus, all the operational benefits of high availability, scalability, and reliability are applicable to queue data. In addition, database development and management tools can be used with queues. For instance, queue tables can be imported and exported.

Structured payload

> Users can use object types to structure and manage the *payload* (i.e., content). RDBMSs in general have had a far richer typing system than messaging systems. Since Oracle8 is an object-relational DBMS, it supports traditional relational types and also user-defined types. Many powerful features are enabled as a result of having strongly typed content (i.e., content whose format is defined by an

external type system). These include:

Content-based routing

An external agent can examine the content and route the message to another queue based on the content.

Content-based subscription

A publish and subscribe system built on top of a messaging system can offer content based on subscription.

Querying

The ability to execute queries on the content of the message enables message warehousing.

Retention and message history

Users can specify that messages be retained after consumption. The system administrator can specify the duration for which messages will be retained. Oracle AQ stores information about the history of each message. The information contains the enqueue/dequeue time and the identification of the transaction that executed each request. This allows users to keep a history of relevant messages. The history can be used for tracking, data warehouse, and data mining operations.

Tracking and event journals

If messages are retained they can be related to each other. For example, if a message m2 is produced as a result of the consumption of message m1, m1 is related to m2. This facility allows users to track sequences of related messages. These sequences represent " event journals" that are often constructed by applications. Oracle AQ is designed to let applications create event journals automatically.

NOTE: Oracle AQ does not automatically produce messages as a result of the consumption of other messages. You will do this programmatically. On the other hand, Oracle AQ does automatically track that these messages were processed in the same transaction.

Integrated transactions

The integration of control information with content (data payload) simplifies application development and management.

5.1.2 Enqueue Features

Oracle AQ offers a wide sweep of options for the way you enqueue messages.

Correlation identifier

Users can assign an identifier to each message, thus providing a means of retrieving specific messages at a later time.

Subscription and recipient lists

Multiple consumers can consume a single message. A queue administrator can specify the list of subscribers who can retrieve messages from a queue. Different queues can have different subscribers, and a consumer program can be a subscriber to more than one queue. Further, specific messages in a queue can be directed toward specific recipients who may or may not be subscribers to the queue, thereby overriding the subscriber list.

Prioritization and ordering of messages

It is possible to specify the priority of the enqueued message. An enqueued message can also have its exact position in the queue specified. This means that users have three options to specify the order in which messages are consumed: (a) a sort order specifies which properties are used to order all messages in a queue; (b) a priority can be assigned to each message; (c) a sequence deviation allows you to position a message in relation to other messages. Furthermore, if several consumers act on the same queue, a consumer will get the first message that is available for immediate consumption. A message that is in the process of being consumed by another consumer will be skipped.

Propagation

With Oracle 8.0.4 and later versions, applications can use AQ to communicate with one another without having to be connected to the same database instance or to the same queue. Instead, messages can be propagated from one Oracle AQ environment to another. These AQ instances can be local or remote. AQ propagation relies on database links and Net8.

Message grouping

Messages belonging to one queue can be grouped to form a set that can only be consumed by one user at a time. This requires the queue be created in a queue table that is enabled for message grouping. Messages belonging to a group must be created in the same transaction, and all messages created in one transaction belong to the same group. This feature allows users to segment complex messages into simple messages. For example, messages directed to a queue containing invoices could be constructed as a group of messages starting with the header, followed by messages representing details, followed by the trailer message.

Time specification and scheduling

Delay interval and/or expiration intervals can be specified for an enqueued message, thereby providing windows of execution. A message can be marked as available for processing only after a specified time elapses (a delay time) and has to be consumed before a specified time limit expires. Messages can also be scheduled for propagation from a queue to a local or remote destination (Oracle 8.0.4 only). AQ administrators have the option to specify the start time, propagation window, and a function that determines the next propagation window for scheduling on a periodic basis.

5.1.3 Dequeue Features

As you would expect, Oracle AQ also allows you to dequeue messages in a variety of methods.

Multiple recipients

A message in a queue can be retrieved by multiple recipients without the need to store multiple copies of that same message.

Navigation of messages for dequeuing

Users have several options for selecting a message from a queue. They can select the first message, or, once they have selected a message and established a position, they can retrieve the next. The selection is influenced by the ordering or can be limited by specifying a correlation identifier. Users can also retrieve a specific message using the message identifier.

Multiple dequeue modes

A dequeue request can either browse or remove a message. If a message is browsed, it remains available for further processing; if a message is removed, it is no longer available for dequeue requests. Depending on the queue properties, a removed message may be retained in the queue table.

Message-waiting optimization

A dequeue could be issued against an empty queue. To avoid polling for the arrival of a new message, a user can specify if and for how long the request is allowed to wait for the arrival of a message.

Retries with delays

A message must be consumed exactly once. If an attempt to dequeue a message fails and the transaction is rolled back, the message will be made available for reprocessing after some user-specified delay elapses. Reprocessing will be attempted up to the user-specified limit.

Optional transaction protection

Enqueue/dequeue requests are normally part of a transaction that contains the requests, thereby providing the desired transactional behavior. Users can, however, specify that a specific request is a transaction by itself, making the result of that request immediately visible to other transactions. This means that messages can be made visible to the external world either as soon as the ENQUEUE or DEQUEUE statement is issued, or only after the transaction is committed.

Exception handling

A message may not be consumed within given constraints -- that is, within the window of execution or within the limits of the retries. If such a condition arises, the message will be moved to a user-specified exception queue.

5.1.4 Propagation Features

Oracle AQ with Oracle 8.0.4 supports propagation of messages, allowing automated coordination of enqueuing and dequeuing operations. The recipient of a message can be either in the same database as AQ (local) or in another database (remote). Since Oracle8 does not support distributed object types, it is not possible to rely on standard database links to perform remote AQ operations. Instead, Oracle AQ offers a special message propagation facility to allow an agent to enqueue to a remote queue.

You can configure AQ so that any messages enqueued in one (local) queue will be automatically propagated to another queue in either the local or remote database. AQ checks automatically that the type of the local queue to which the message is enqueued is the same as the type of the remote queue (same payload type). You can perform this same check by calling the DBMS_AQADM.VERIFY_QUEUE_TYPES procedure.

There are two useful ways to think about employing message propagation in Oracle AQ: *fanning out* and *funnelling in* messages.

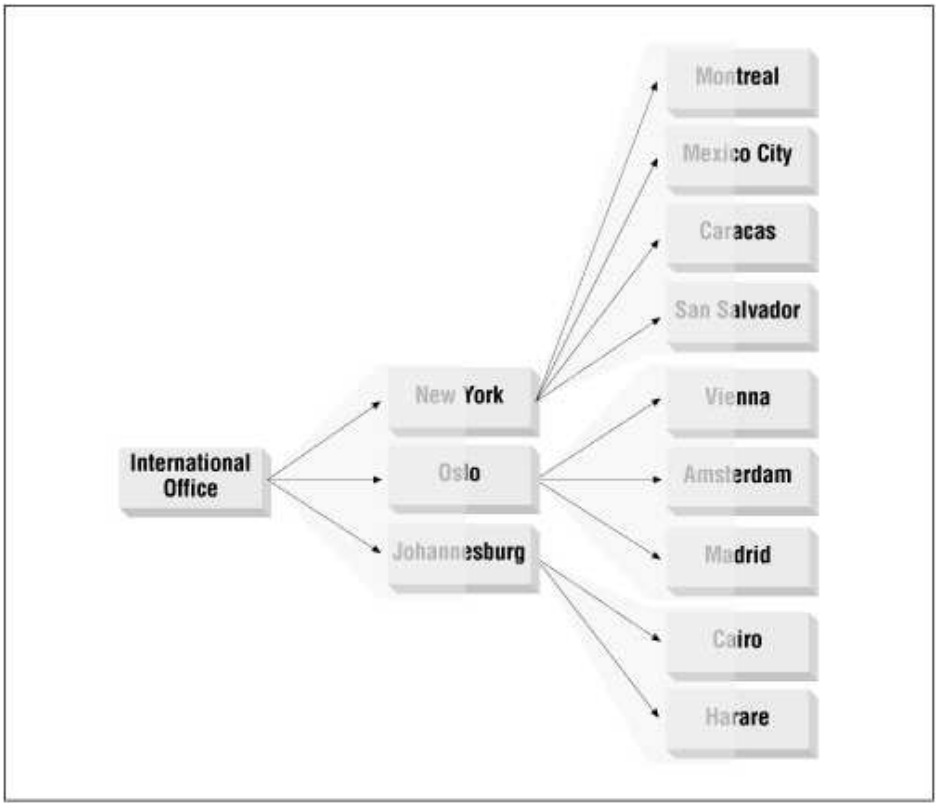
Fanning out messages

Suppose you want to distribute a message to a large number of recipients without forcing those recipients to perform dequeue operations from a single queue. This is most important when working in a distributed database environment. You can accomplish this by defining another queue as a recipient of a message. When a queue is a recipient for a message, the actual recipients are the set of agents defined in the subscription list for that queue.

Consider the following scenario: an international human rights organization based in London has received news of the torture and imprisonment of a dissident. It needs to notify its branch organizations throughout the world as quickly as possible -- using Oracle AQ, of course. The subscribers to its *urgent_alert* queue consist of each of its regional offices: *urgent_alert@new_york* , *urgent_alert@johannesburg*, and so on. Each of the regional urgent alert queues has as its subscribers the country offices. So the Johannesburg queue has the following subscribers: *urgent_alert@harare* , *urgent_alert@cairo*, and so on.

When the headquarters office enqueues the alert message to its *urgent_alert* queue, the message will be propagated out to each regional office queue. That message will then in turn be propagated to each country office. Figure 5.1 illustrates the fanning out technique.

Figure 5.1: Fanning out messages

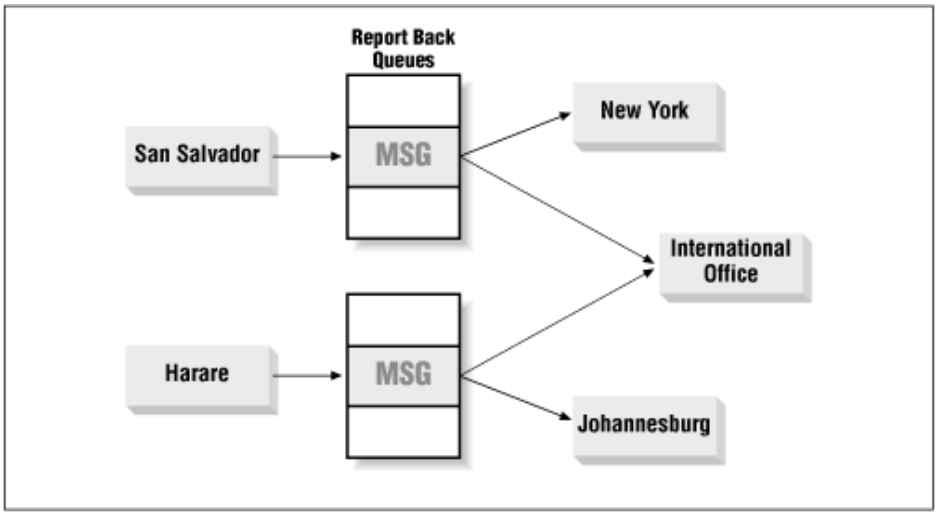


Funnelling in messages

It is also very useful to concentrate messages from different queues back to a single queue. Examples of this technique, also called " compositing," include roll-up of monthly accounting figures from regional offices and confirmation messages in response to a broadcast.

If we use the same scenario as with the fanning out approach, each country office employs a *report_back* queue to confirm that urgent alerts have been received and responded to. The subscriber list for each office's *re port_back* queue include the regional office as well as the international office. As a result, the London-based *report_back@london_hq* will automatically receive a copy of all confirmation reports without having to wait for those messages to be transferred back through the individual regional offices. Figure 5.2 illustrates the funneling in technique.

Figure 5.2: Funneling in messages



5.1.5 A Glossary of Terms

Before diving into the sometimes overwhelming details of configuring queue tables and queues and enqueuing and dequeuing messages, let's review some of the terms used throughout this chapter. Then we'll take a step back and look at the process flow for the queuing facility. At that point, you'll have a much easier job understanding and then deploying Oracle AQ.

The basic elements of Oracle AQ follow:

Message

The smallest unit of work in the queue, consisting of information about how the message is to be treated (metadata) and the payload (the data supplied by the user). The metadata or control information is used by AQ to manage the messages. The payload information is stored in the queue and is transparent to AQ (that is, AQ does not try to *interpret* that information in any way). A message resides in only one queue. It is created by a call to the DBMS_AQ.ENQUEUE procedure and is dequeued by a call to the DBMS_AQ.DEQUEUE procedure.

Queue

A queue is a storage space for messages. You can create two different types of queues: *user queues* , also known as normal queues, and *exception queues* . The

user queue is employed for standard message processing. The exception queue is used to hold messages if attempts to retrieve the message through a dequeue operation fail (this is done automatically by AQ) or if the message is not dequeued before its expiration time. You will use the AQ administrative interface, the DBMS_AQADM package, to create, start, stop, and drop queues.

Queue table

A queue table is a database table that holds one or more queues; this queue table is created when you create the queue table. Each queue table also contains a default exception queue. You will use the AQ administrative interface (the DBMS_AQADM package) to create and drop queue tables.

Agent

An agent is a user of a queue. There are two types of agents: *producers* , who place messages in a queue (enqueuing); and *consumers* , who retrieve messages (dequeuing). Any number of producers and consumers may be accessing the queue at a given time. An agent's name, address, and protocol identify that agent. The address field is a character field of up to 1024 bytes that is interpreted according to the protocol value (of which the only supported value is currently 0). A protocol of 0 indicates that the address is to be interpreted as a database link. The address will therefore have this form,

```
queue_name@dblink
```

where queue_name has the form [schema.]queue and dblink is either a fully-qualified database link name or a database link name that does not incorporate the domain name.

Agents insert messages into a queue and retrieve messages from the queue by using the Oracle AQ operational interfaces offered in the DBMS_AQ package.

Recipient list

A list of one or more agents that you can construct to receive a message (through the dequeue operation). With Oracle 8.0.4., a recipient can be either an agent or a queue.

Producer

An agent that places messages in a queue with the enqueue operation.

Consumer

An agent that retrieves messages from a queue with the dequeue operation. Note that you can modify the characteristics of your dequeue operation so that the agent does not actually consume (i.e., read and then remove from the queue) its message.

Message ID

The unique handle for a message. This value is generated by AQ. You can use it to retrieve a specific message from a queue (bypassing the default order of dequeuing associated with the queue and/or queue table). You can also use this message ID to find out about the status of a message from the underlying data dictionary views. These message IDs are long, complex values like

"105E7A2EBFF11348E03400400B40F149."

Message group

One or more messages can be joined together logically as a *group* . You do this by specifying a queue table as supporting message grouping. All messages queued in a single transaction must then be dequeued as a group for that set of messages to be considered completely dequeued.

Queue Monitor

The Queue Monitor is an optional background process that monitors the status of messages in your queues. Use the Queue Monitor when you want to set expiration and delay intervals for messages.

Lots of concepts, lots of terminology. Let's see if we can obtain some additional clarity from the following figures.

Figure 5.3 illustrates how you create one or more queues within a queue table. Each queue can have one or more messages. Messages in different queues do not have any relation to each other. Each queue table has a single default exception queue, which contains messages that have expired or have otherwise failed to dequeue successfully

Figure 5.4 shows that producer agents place messages in a queue and consumer agents remove messages from a queue. The same Oracle process can be both a producer and a consumer; more commonly, you will use Oracle AQ to allow multiple Oracle connections to enqueue and dequeue messages. Some points to keep in mind:

- Messages do not have to be dequeued in the same order in which they were enqueued.
- Messages can be enqueued but never dequeued.

Figure 5.3: Multiple queues in a single queue table

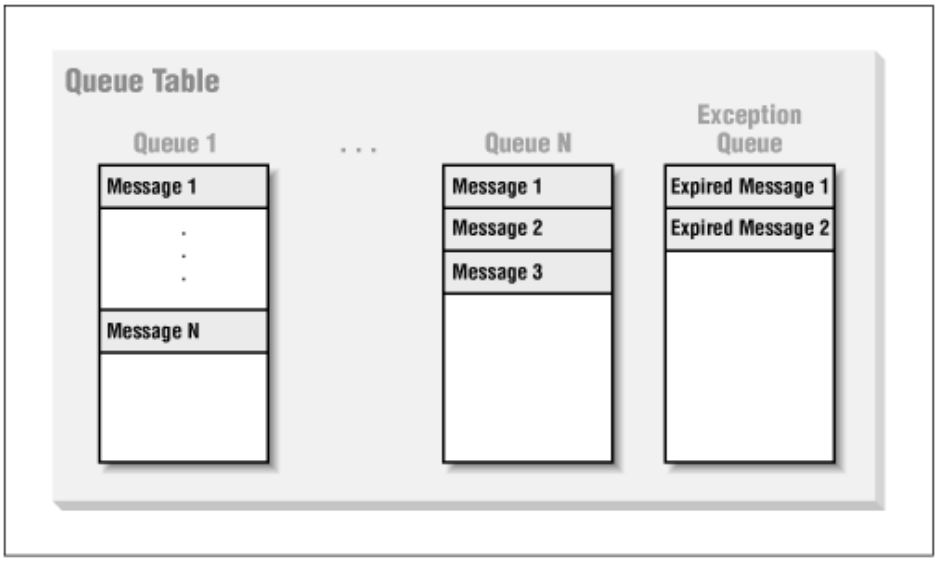


Figure 5.4: Producers enqueue, consumers dequeue

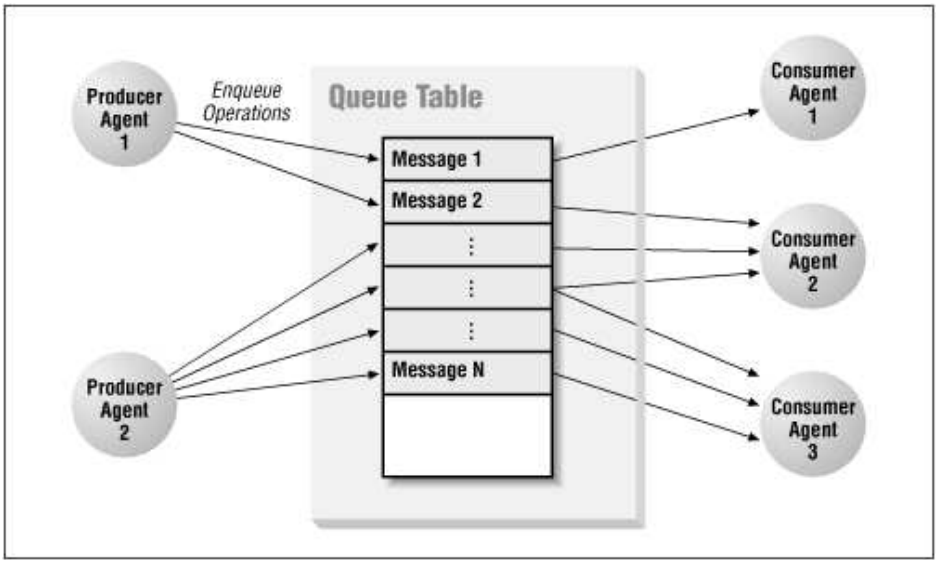
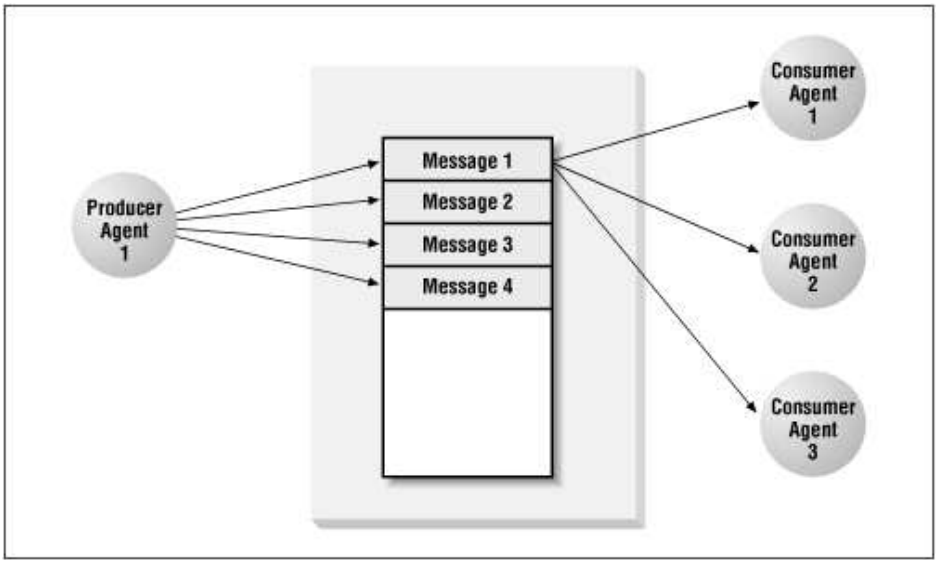


Figure 5.5 illustrates the " broadcast" feature of Oracle AQ. You can define a queue table to support " multiple consumers." In this configuration, a single message can be consumed by more than one agent, either through the default subscription list or with an override recipient list. Under this scenario, a message remains in the queue until it is consumed by all of its intended consumer agents.

When you set up a subscriber list for a queue, you are establishing that list of agents as the default set of agents to be able to dequeue messages from the queue. You can change this list at any time, but the change will affect only those messages enqueued after the change is made. The subscription list is, in other words, associated with a message at the time of the enqueue operation, not with the dequeue operation.

You will use the r ecipient list to override the subscription list for dequeuing. Under this scenario, at the time the message is enqueued, you specify a list of agents to which the message may be dequeued. The default list of subscribers for the queue is then ignored for dequeue operations on this message.

Figure 5.5: Multiple consumers of the same message



5.1.6 Components of Oracle AQ

Oracle AQ is composed of a number of different elements, not just a single built-in package. These elements follow:

The DBMS_AQADM package

An interface to the administrative tasks of Oracle AQ, such as creating and dropping queues and queue tables. This package and the programs it supports, are described in later sections.

The DBMS_AQ package

The package that offers access to the enqueue and dequeue operations (the "operational tasks") of Oracle AQ. This package, and the programs it supports are described in later sections.

The Queue Monitor

A background process that can be used to delay and expire messages for dequeuing (described in the Section 5.1.7, "Queue Monitor" " section following).

Data dictionary views

A set of views against the underlying AQ tables that allows users of AQ to view the results of administrative and operational tasks (described in the Section 5.1.8, "Data Dictionary Views" " section following).

5.1.7 Queue Monitor

The Queue Monitor is an Oracle process that runs in the background and monitors the status of messages in your queues. It is optional and is present only when you set the appropriate database initialization parameter, or you call the DBMS_AQADM.START_TIME_MANAGER procedure (see Section 5.2.2, "Database Initialization" ," later in this chapter). You will need the Queue Monitor if you want to set expiration and delay intervals for messages.

5.1.8 Data Dictionary Views

Oracle AQ offers a set of data dictionary views that allows you to monitor the status of queuing operations. This section lists the different views. At the end of the chapter, you will find a section titled Section 5.6, "Oracle AQ Database Objects" ," which offers more details on these views and other database objects created and used by Oracle AQ.

Queue table database table

The queue table in which message data is stored. This table is created automatically when you create a queue table. The name of this table is <queue_table>, where <queue_table> is the name of the queue table you specified.

Queue table view

A view of the queue table in which message data is stored. This view is created automatically when you create a queue table. The name of this view is aq$<queue_table>, where <queue_table> is the name of the queue table you specified.

DBA_QUEUE_TABLES

This view describes the names and types of all queue tables created in the database. To see this view, you must have the DBA authority or the SELECT

ANY TABLE privilege.

USER_QUEUE_TABLES

This view describes the names and types of all queue tables created in your schema. It has the same structure as DBA_QUEUE_TABLES, except that it does not have an OWNER column.

DBA_QUEUES

This view displays all operational characteristics for every queue in a database. Operational characteristics include whether the queue is enabled for queuing, the number of retries allowed for a dequeue operation, and so on. To see this view, you must have DBA authority or the SELECT ANY TABLE privilege.

USER_QUEUES

This view displays all operational characteristics for every queue in your schema. It has the same structure as DBA_QUEUES, except that it does not have an OWNER column.

PREVIOUS 4.2 DBMS_TRANSACTION: Interfacing to SQL Transaction Statements

HOME BOOK INDEX

NEXT 5.2 Getting Started with Oracle AQ

Library Home | Oracle PL/SQL Programming, Second Edition | Oracle PL/SQL Programming: Guide to Oracle8i Features | Oracle Built-in Packages | Advanced PL/SQL Programming with Packages | Oracle Web Applications: PL/SQL Developer's Introduction | Oracle PL/SQL Language Pocket Reference | Oracle PL/SQL Built-ins Pocket Reference