Oracle Advance Queuing: An Overview

Hamid R. Minoui

Fritz, a UPS Company

Messaging & Queuing

- An essential part of Distributed Application Development (DAD)
- A key piece of a middle-ware known as Message-Oriented Middleware (MOM)
- Motto: Every DAD needs a MOM

What does MOM provide?

- Time-independent responses in a client/server environment
- Help in passing information between servers and clients
- A facility that allows general-purpose messages to be exchanged in a client/server system using message queues

Communication with Messages

- Applications communicate over networks by putting messages in queues (ENQUEUE) and by getting messages from queues (DEQUEUE) using simple high-level API

Store-and-forward Communication

- Characteristics and benefits
  - Less vulnerable to network, machine, and application failures.
  - No need for a real-time logical or dedicated connection
  - Connections can be established later with no loss
  - Applications can run independently at different speeds

Application Communication models

Two Application Models

- Synchronous (online, connected)
  - Program sends a request to another program and waits for the reply
- Asynchronous (deferred,disconnected)
  - Producer programs place messages in a ***queue*** and continue
  - Consumer programs retrieve messages (requests) from the queue and act on them

A sample synchronous model

- A credit needs to be approved before a loan application can be processed
- The load processing module awaits the completion of the credit approval model before it can do any work

A sample Asynchronous model

- An order entry system places orders in appropriate order queues to be processed later. A complete order.
- The order queues are read and handled one-at-a-time, in specific sequence , by the order processing systems

Oracle AQ

- Oracle Advance Queuing is Oracle's message queuing facility
- Available as of Oracle 8.0.3
- First database-integrated messaging system in the industry

Overview of AQ Features

- Querying
- Retention
- Propagation
- Exception Handling
- Subscription List
- Multiple Recipients
- Time Specification
- Statistics
- Optional Transaction Protection
- Priority & Ordering

SQL-based access

- SQL-based access
  - Messages are placed in normal rows in a database table.
  - They can be queried using standard SQL.
  - SQL can be used to access
    - Message properties
    - Message history
    - Payload
  - Indexes can be used to optimize access

Database-level Operational support

- Standard database facilities: backup, recovery,export, import, OEM
- AQ queues also benefit from high availability, scalability and reliability features of the Oracle server

Structured Payload

- Object types can be used to structure and manage message content (or payload)
- Support of strongly type content enables these features:
  - Content-based routing
  - Content-based subscription
  - Querying which enables message warehousing

Retention & History

- Retention
  - AQ users can specify messages to be retained after consumption
  - Administrators can specify the retention period
- Message History
  - History information like enqueue/dequeue time and the identification of transactions that executed each request is stored by AQ

Tracking & Journaling

- Retained messages can be related to each other
- Users can track sequences of related messages and produce event journals automatically

Other Features

- Integrated Transactions
  - Management and development is simplified by the integration of control information and data payload
- Queue Level Access Control (oracle 8i)
  - Queue level privileges can be granted or revoked by queue owners
  - DBAs can grant or revoke AQ system level privileges to database users
  - AQ administration can also be granted or revoked

AQ Administrator

- AQ Administrator created by DBA:
  - Create the user (aqadm)
  - Grant AQ_ADMINISTRATOR_ROLE
  - Grant connect, resource to aqadm;
- Additional grants:
  - Execute on dbms_aqadm
  - Execute on dbms_aq

AQ User

- AQ_USER_ROLE granted by AQ Administrator
- Has execute privilege on DBMS_AQ

Enqueue Options

- Correlation Identifier
- Subscription & Recipient Lists
- Message Ordering & Prioritization
- Propagation
- Sender Identification
- Message Grouping
- Time specification & Scheduling
- Rule-based Subscribers
- Asynchronous Notification

Correlation Identifier

- Users can assign an identifier to each message.
- A specific enqueued message may be retrieved later by its identifier

Publish and Subscribe

- Allows the establishment of a publish and subscribe mechanisms that include:
  - Rule-based subscription
  - Message propagation
  - Listening for incoming messages
  - Notification capabilities

Subscription & Recipient Lists

- Multiple consumers can consume a single message from a queue.
- A queue administrator can specify the list of subscribers who can retrieve messages from a queue
- Different queues can have different subscribers

Subscribers & Recipients (more)

- A consumer program can subscribe to more than one queue
- Specific messages in a queue can be directed towards specific recipients who may not be in the subscribers list for the queue

Prioritization & Ordering

- Enqueued messages can be assigned priority
- The queue position of enqueued messages can be specified

Prioritizing & Ordering Options

- Three Consuming Options

1. A sort order specifies properties used for ordering messages in a queue
2. A priority can be assigned to each message
3. A sequence deviation by allowing to position a message in relation to the others

Propagation Features

- Allows coordination of ENQUEUE and DEQUEUE operations
- Allows for location independence between the senders and recipients
- Propagation agents automatically ENQUEUE from a local queue to another local or remote queue using database links

Starting Propagation Process

- Set JOB_QUEUE_PROCESSES according to the number of queues involved in propagation
- Set COMPATIBLE to 8.0.4 or higher
- Schedule propagation
  - DBMS_AQADM.SCHEDULE_PROPAGATION
- Remove propagation schedules
  - DBMS_AQADM.UNSCHEDULE_PROPAGATION

Message Grouping

- Messages for a queue can be grouped in the queue to form a set to be consumed by one user at a time
- All messages in a group are created in one transaction
- Queue table of the queue must be enabled for message grouping

Sender Identification

- Applications can mark the messages they send
- Oracle identifies the queue from which a message was dequeued
- Applications can track the pathway of a propagated message

Time Specification & Scheduling

- A enqueued message allows execution window by specifying:
  - Delay interval
  - Expiration time
- A message is available after the delay and before the expiration

AQ_TM_PROCESSES

- An INIT.ORA parameter for managing the time:
  - In which messages are ***available*** for dequeuing
  - After which messages are ***expired***
- Values between 1 and 10 creates that many Queue Monitor background processes to monitor messages in queues
  - Processes are named ora_aqtm<oracle_sid>

Queue Monitor

- To start or stop the Queue Monitor
  - DBMS_AQADM.START_TIME_MANAGER
  - DBMS_AQADM.STOP_TIME_MANAGER

Rule-based subscribers

- Users can subscribe to receive only messages of specified properties or contents
- They define a rule-based subscription for a given queue to receive only messages of interest

Dequeue Options

- Multiple Recipients
- Local & Remote Recipients
- Navigation of Messages in Dequeing
- Dequeuing Modes
- Dequeue Message header/ No Payload
- Optimization of Waiting for the Arrival of Messages
- Retries with Delays
- Transaction Protection
- Exception Handling
- Wait on Multiple Queues

Multiple Recipients & Navigation

- Multiple recipients can DEQUEUE the same message
- Users can DEQUEUE in several ways:
  - Retrieve the first message
  - Establish a position & retrieve relative to that position
  - Select based on an order
  - Use the message identifier

DEQUEUE modes

- Browse
  - Will stay in the queue for more processing
- Remove
  - No longer available for other dequeue requests
- Locked
  - Messages are locked for other dequeuing request for the duration of the transaction

Fanning out Messages

- Distribute messages to a large number of recipients.
  - Other queues would be recipients
  - Agents are defined as subscribers

Funneling In Messages

- Concentrate messages for many queue to a single queue, also called composing
- For example to get confirmation from a broadcast message

Basic AQ elements

- Message
- Queue
- Queue Table
- Agent
- Queue Monitor
- Recipient List
- Producer
- Consumer
- Message ID
- Message group

Basis Steps

- Create a queue table
- Create one or more queues in the queue table
- Start the queue manager
- Enqueue/dequeue to/from queue

Programming Interfaces

- Users Programming Interface
  - DBMS_AQ
    - enqueue/dequeue
- AQ Administrator Interface
  - DBMS_AQADM
    - Create/drop queue, queue table
    - Start/Stop queue manager

Message

- The smallest unit of work in the queue
- Contains
  - Metadata (control information)
  - Payload (supplied data)
- Created by DBMS_AQ.ENQUEUE

Queue

- Data structure for messages
- Two types of queues can be created
  1. Users Queues (normal queues)
     - Used by standard message processing
  2. Exception Queues
     - Used by AQ for messages indicative of failed DEQUEUE attempts, or expired messages
- Managed by DBMS_AQADM

Queue Table

- A database table that holds one or more queues
- A queue table also contains a default exception queue

Simple AQ Examples

- Define a message object type

```
Create TYPE message_type AS OBJECT
        (title   VARCHAR2(30),
         text    VARCHAR2(2000));
```

Create A Queue Table

- AQADM creates a queue table called msg:

```
EXEC DBMS_AQADM.CREATE_QUEUE_TABLE
   (queue_table => ‘msg’,
    queue_payload_type => ‘message_type’);
```

Create & start a queue

- AQADM creates a queue named msgqueue in msg and start it:

```
EXEC DBMS_AQADM.CREATE_QUEUE
   (queue_name => ‘msgqueue’,
    queue_table => ‘msg’;

EXEC DBMS_AQADM.START_QUEUE
   (queue_name => ‘msgqueue’);
```

Message Properties

- Default message property record

```
TYPE DBMS_AQ.MESSAGE_PROPERTIES_T IS RECORD
   (priority            BINARY_INTEGER DEFAULT 1,
    delay               BINARY_INTEGER DEFAULT DBMS_AQ.NODELAY,
    expiration          BINARY_INTEGER DEFAULT DBMS_AQ.NEVER,
    correlation         VARCHAR2(128) DEFAULT NULL,
    attempts            BINARY_INTEGER,
    recipient_list      DBMS_AQ.AQ$_RECIPIENT_LIST_T,
    exception_queue VARCHAR2(51) DEFAULT NULL,
    enqueue_time  DATE,
    state                BINARY_INTEGER);
```

Set message properties

- Example code:

```
DECLARE
    msg_prop     DBMS_AQ.MESSAGE_PROPERIES_T;
BEGIN
    msg_prop.priority := -100       /*high*/
    msg_prop.delay:= 60*60*24    /* 1 day */
    msg_prop.expiration:= 60*60 /* 1 hour after delay */
     …..
```

Enqueue Options Record Type

- Specify options associated with the message when you enqueue it

```
TYPE DBMS_AQ.ENQUEUE_OPTIONS_T IS RECORD
  (visibility                BINARY_INTEGER
                                 DEFAULT DBMS_AQ.ON_COMMIT,
   relative_msgid            RAW(16)
                                 DEFAULT NULL,
   sequence_deviation        BINARY_INTEGER
                                 DEFAULT NULL);
```

Set enqueue options

- Code example:
  - Have the message placed at the top of the queue and immediately visible.

```
DECLARE
    queue_opts    DBMS_AQ.ENQUEUE_OPTIONS_T;
BEGIN
    queue_opts.visibility                := DBMS_AQ.IMMEDIATE;
    queue_opts.sequence_deviation := DBMS_AQ.TOP
```

ENQUEUE the message

```
DECLARE
    queue_opts      DBMS_AQ.ENQUEUE_OPTIONS_T;
    msg_props       DBMS_AQ.MESSAGE_PROPERTIES_T;
    msg_id          aq.msgid_type;   /* set in a user package */
    the_message     message_type;
BEGIN
    the_message := message_type ('First msg', 'more to come..');
    DBMS_AQ.ENQUEUE ('msgqueue', queue_opts, msg_props,
                                    the_message, msg_id);
END;
/
```

Change Message Properties & Options

- Delay first message by 2 days

```
msg_props.delay := 2 * 60 * 60 * 24
```

- Request one message to be dequeued before another

```
DECLARE msg_id1, msg_id2 := aq.msgid_type;
….
queue_opts.sequence_deviation := DBMS_AQ.BEFORE;
queue_opts.relative_msgid := msg_id1;
DBMS_AQ.ENQUEUE (‘msg_queue’, queue_opts, msg_props,
      another_message, msg_id2;
```

DEQUEUE Specification

```
PROCEDURE DBMS_AQ.DEQUEUE
    (queue_name           IN VARCHAR2,
     dequeue_options      IN DBMS_AQ.DEQUEUE_OPTIONS_T,
     message_properties   OUT DBMS_AQ.MESSAGE_PROPERTIES_T,
     payload              OUT <type_name>,
     msgid                OUT RAW);
```

DEQUEUE Options Record Type

```
TYPE DBMS_AQ.DEQUEUE_OPTIONS_T IS RECORD
    (consumer_name      VARCHAR2(30)
                        DEFAULT NULL,
     dequeue_mode       BINARY_INTEGER
                        DEFAULT DBMS_AQ.REMOVE,
     navigation         BINARY_INTEGER DEFAULT
                        DBMS_AQ.NEXT_MESSAGE,
     visibility         BINARY_INTEGER
                        DEFAULT DBMS_AQ.ON_COMMIT,
     wait               BINARY_INTEGER
                        DEFAULT DBMS_AQ.FOREVER,
     msgid              RAW(16)          DEFAULT NULL,
     correlation        VARCHAR2 (128) DEFAULT NULL);
```

DEQUEUE the message

- Code example:

```
DECLARE
    queue_opts      DBMS_AQ.ENQUEUE_OPTIONS_T;
    msg_props       DBMS_AQ.MESSAGE_PROPERTIES_T;
    msg_id          aq.msgid_type;   /* set in a user package */
    the_message     message_type;
BEGIN
    DBMS_AQ.DEQUEUE ('msgqueue', queue_opts, msg_props,
                                    the_message, msg_id);
    DBMS_OUPUT.PUT_LINE ('Dequeued text: ' ||
                                        the_message.text);
END;
/
```

Stopping & Dropping Operations

- On QUEUES
  - DBMS_AQADM.STOP_QUEUE
  - DBMS_AQADM.DROP_QUEUE
- On QUEUE TABLES
  - After all queues in the table are dropped
  - DBMS_AQADM.DROP_QUEUE_TABLE

Data Dictionary Views

- DBA_QUEUE_TABLES
- USER_QUEUE_TABLES
- DBA_QUEUES
- USER_QUEUES
- AQ$<queue_table>

AQ Statistics Views

- Basic queue statistics are available via the V$AQ view
- This view can be queried to see the number of messages in waiting, ready or expired state for each queue
- Total and average wait time for all ‘READY’ messages in the queue is also recorded

The End of The Queue

- This concludes the AQ overview presentation
- Thank you for your attention

ANY QUESTIONS ????