Oracle AQ – Example for queue-to-queue transmission in same database/another

Presenters: - Anantha Narayanan, Sakthivel
Visit our blog at: http://askanantha.blogspot.com

In this note we will se how to transmit messages from one queue to another queue. We will enqueue messages into one queue and dequeue the same message from another queue. The destination queue can be in the same database or another. Here we will see how to send messages between queues assuming that both the queues are in different database.

The technique for transmitting the messages from one queue to another is called **Propagation**. There are various usages for propagation. One such usage is illustrated below:

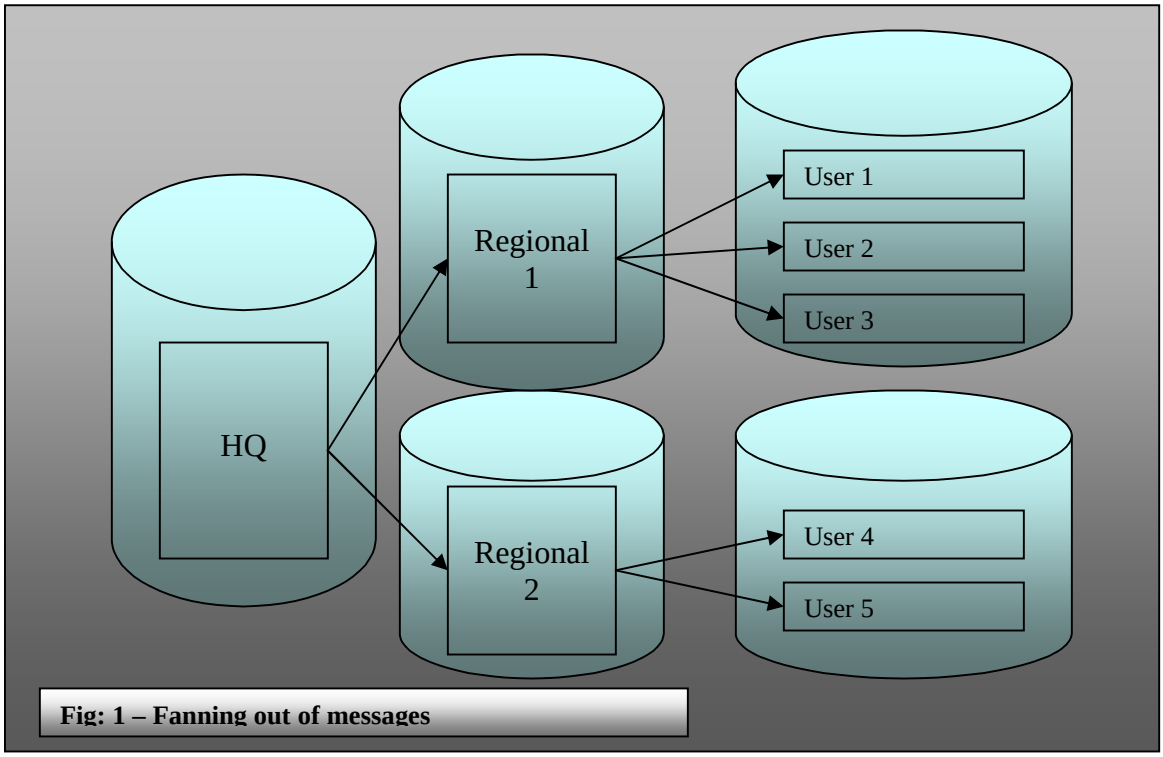

Fig: 1 – Fanning out of messages

In this above figure, the technique **fanning out of messages** is depicted. There are totally 8 queues managed in this scenario. The User 1, 2 and 3 are subscribers of Regional 1 queue. The User 4 and 5 are subscribers of Regional 2 queue. The queues Regional 1 and 2 are subscribers of HQ queue.

This technique is normally used in case of multi-database architecture where load balancing of message sending has to be achieved. Whenever message is enqueued in HQ queue, it is propagated to Regional 1 and 2 queues. The Regional 1 and 2 queues in turn will propagate the messages to their respective subscribers. There are rule based propagations that makes this happen.

The other technique of propagating messages is the reverse of fanning. Here the Users will be propagating messages to Regional and it in turn will propagate to HQ queue. This method is called **Funneling in messages**. It is illustrated below:

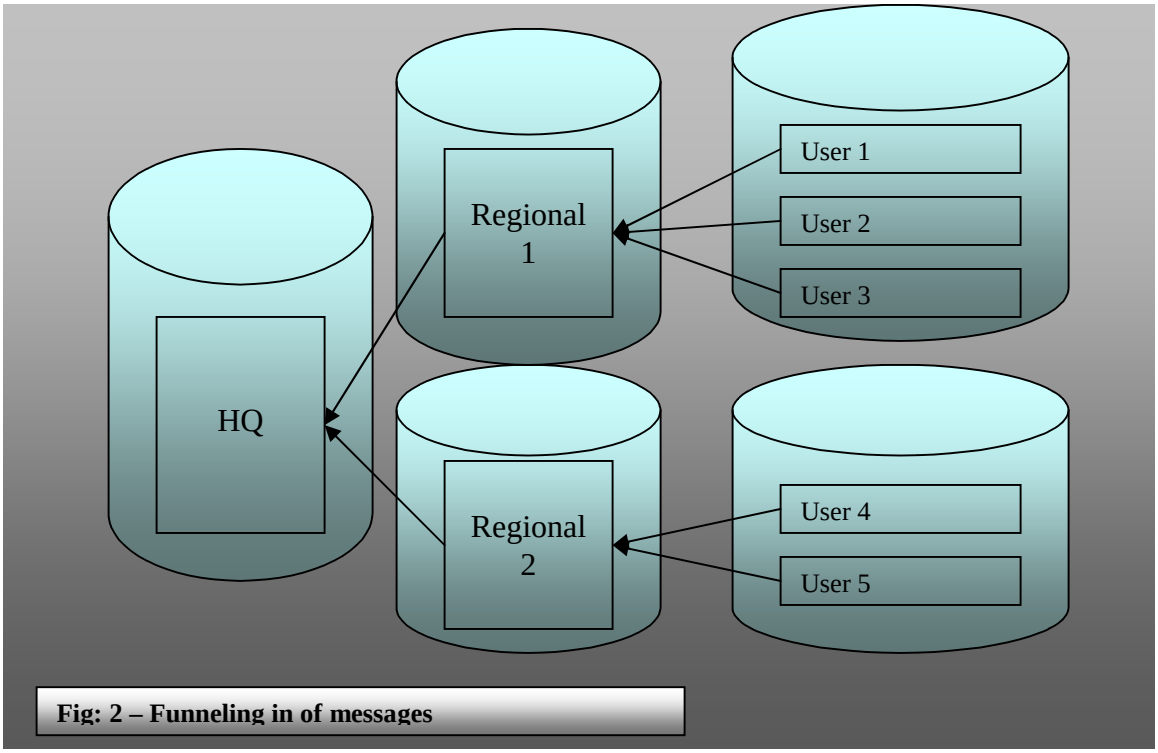

Fig: 2 – Funneling in of messages

In the following diagram the propagation of messages from one queue to another is depicted:

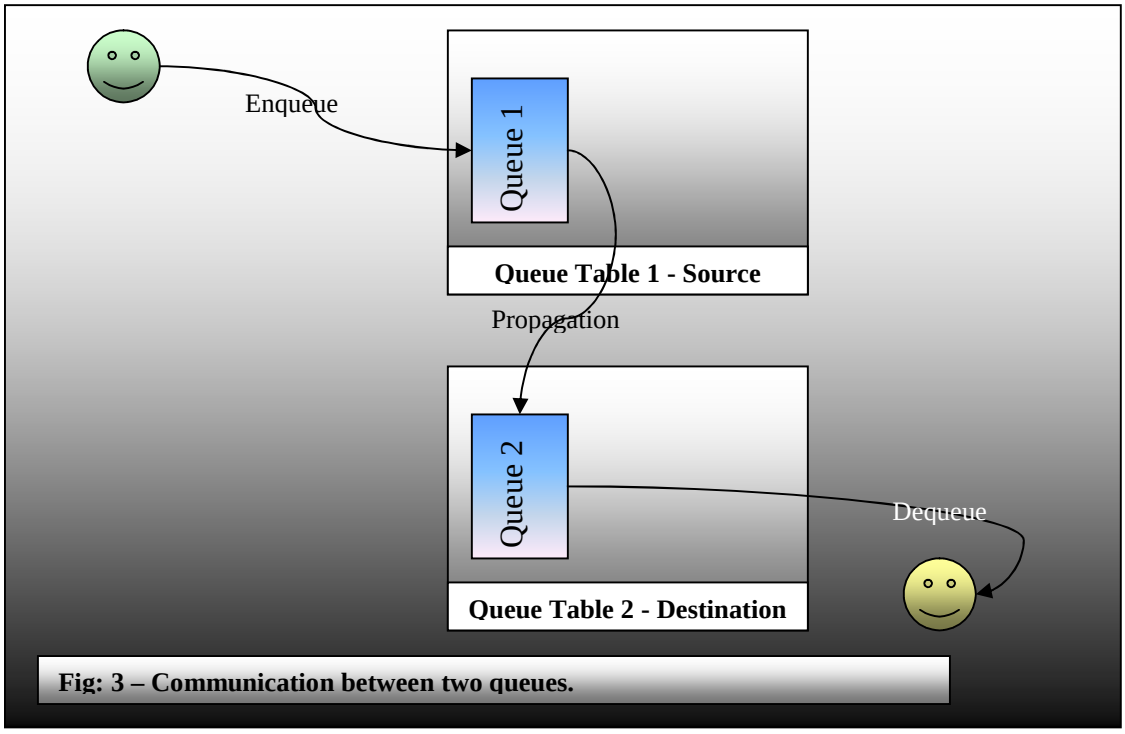

Fig: 3 – Communication between two queues.

In this note we will see how to propagate messages from one queue to another. Here in this example it is a "blind" propagation is what we are focusing to make the idea clear. All messages enqueued in the source queue will be propagated to destination queue.

Here is our agenda for the example:
Step 1: Create object type for Payload and create dblink
Step 2: Create Destination queue table and queue, and add subscriber
Step 3: Create Source queue table and queue, and add subscriber
Step 4: Schedule propagation from source queue
Step 5: Enqueue message in source queue
Step 6: Dequeue message from destination queue

Let's get started:

1. Create object type for Payload and create dblink

First we will define the payload for the message to be created. This payload has to be created in both ***source and destination database***.

```
create type mytype as object (n number);
/

PL/SQL procedure successfully completed
```

We will create database link from source to destination database. (This has to be created in source database).

```
create database link symadm_to_slsp connect to symadm identified by symadm using 'slsp';

Database link created.
```

Note: The **using clause** in create database link must be present in TNSNAMES.ORA in creating database.

2. Create Destination queue table, queue and add subscriber

Here we will create queue_table in destination schema first. Refer the following code:

```
Begin
  dbms_aqadm.create_queue_table('queue_table','mytype',multiple_consumers => true);
End;
/

PL/SQL procedure successfully completed
```

Note: For messages to go from one queue to another, the queue table must be created as a Multiple-consumer (Publish-Subscriber mode). This is required because the destination queue has to be a subscriber for the source queue.

Now we will proceed to create the queue:

```
begin
  dbms_aqadm.create_queue('queue2','queue_table');
end;
/

PL/SQL procedure successfully completed
```

Now we will start the queue:

```
begin
  dbms_aqadm.start_queue('queue2');
end;
/

PL/SQL procedure successfully completed
```

Now we will add a subscriber who will dequeue messages from destination queue:

```
begin
  dbms_aqadm.add_subscriber(queue_name => 'queue2',
                            subscriber => sys.aq$_agent('SUBS2',null,null));
end;
/

PL/SQL procedure successfully completed
```

This concludes the setup in destination user schema.

3. Create Source queue table, queue and add subscriber

Here we will create queue_table in source schema. Refer the following code:

```
Begin
dbms_aqadm.create_queue_table('queue_table_source','mytype',multiple_consumers => true);
End;
/

PL/SQL procedure successfully completed
```

Now we will proceed to create the queue:

```
begin
  dbms_aqadm.create_queue('queue1','queue_table_source');
end;
/

PL/SQL procedure successfully completed
```

Now we will start the queue:

```
begin
  dbms_aqadm.start_queue('queue1');
end;
/

PL/SQL procedure successfully completed
```

Now we will add a subscriber to source queue (The agent properties has to include the destination queue name in the address parameter):

```
begin
  dbms_aqadm.add_subscriber(queue_name => 'queue1',
                            subscriber =>
sys.aq$_agent('SUBS1','symadm.queue2@symadm_to_slsp',null), queue_to_queue => true);
end;
/

PL/SQL procedure successfully completed
```

Note: While propagation both the queues must be of same datatype. (However there is a possibility to use other datatypes, but they must be converted to same format before processing. This we will cover in transformation of messages). In this case as we are creating both queues based on same type, we will not encounter any errors. But if two types are used, DBMS_AQADM consists of one procedure using which you can check whether both are same or not.

The procedure is DBMS_AQADM.VERIFY_QUEUE_TYPES. It takes both queue names as parameters. See the below example:

```
declare
  rc binary_integer;
begin
  dbms_aqadm.verify_queue_types(src_queue_name => 'queue1',
                                dest_queue_name => 'symadm.queue2',
```

```
                                destination=>'symadm_to_slsp',
                                rc => rc);
  dbms_output.put_line(rc);
end;
/

VQT: new style queue
1

PL/SQL procedure successfully completed
```

The status returned by DBMS_AQADM.VERIFY_QUEUE_TYPES is 1. It confirms that both queues are same. If queues are not same then it will return 0.

4. Schedule propagation from source queue

Now we will schedule propagation of message from queue1 to queue2.

```
begin
  dbms_aqadm.schedule_propagation(queue_name => 'queue1',
                                  destination => 'symadm_to_slsp'
                                 ,start_time => sysdate
                                 ,duration => 10
                                 ,next_time => 'sysdate'
                                 ,latency => 25
                                 ,destination_queue => 'symadm.queue2');
end;
/

PL/SQL procedure successfully completed
```

In the destination parameter we have to pass the database link name. In the destination_queue parameter, we have to pass the <schema_name>.<destination_queue_name>. The next_time parameter can be avoided which will make the propagation run once. The duration parameter specifies the turn around time to propagate messages. The latency parameter specifies the turn around time when there are no messages to propagate.

5. Enqueue message in source queue

Now we will enqueue message in source queue:

```
declare
  rc binary_integer;
  nq_opt dbms_aq.enqueue_options_t;
  nq_pro dbms_aq.message_properties_t;
  datas  mytype;
  msgid  raw(16);
begin
    nq_opt.visibility := dbms_aq.immediate;
    nq_pro.expiration := dbms_aq.never;
    datas := mytype(200);

    dbms_aq.enqueue('queue1',nq_opt,nq_pro,datas,msgid);
end;
/

PL/SQL procedure successfully completed
```

```
Select q_name, user_data from queue_table_source;

Q_NAME          USER_DATA.N
--------        -----------
QUEUE1          100
```

7. Dequeue message from destination queue

Now assuming that the job for propagation is run, we will dequeue the message from the destination queue:

```
declare
   dq_opt   dbms_aq.dequeue_options_t;
   dq_prop   dbms_aq.message_properties_t;
   datas     mytype;
   msg_id raw(16);
begin
  dq_opt.consumer_name := 'SUBS2';
  dq_opt.dequeue_mode := dbms_aq.browse;
  dq_opt.navigation := dbms_aq.first_message;
  dq_opt.wait := dbms_aq.no_wait;

  dbms_aq.dequeue(queue_name => 'QUEUE2'
                 ,dequeue_options => dq_opt
                 ,message_properties => dq_prop
                 ,payload => datas
                 ,msgid => msg_id);

  dbms_output.put_line('Message dequeued='||datas.n);
end;
/

Message dequeued=100

PL/SQL procedure successfully completed
```

Well that concludes our session on queue-to-queue transmission in different database.

Material prepared for training on Oracle AQ by http://askanantha.blogspot.com