After a long silence, it's time to write another blog entry. I received a request to write about Oracle Reports. And I think that will be a new series (Even though I didn't finish the workflow series yet).

But the last few days, I've been working on cleaning up workflow tables. Most of these tables are very straightforward and you can find queries and descriptions in the workflow series. However, there are some more complex cases. There are the advanced queuing (aq) tables. And also some data hidden in wf_item_attribute_values for items started by the Business Event System (BES) (reminder to self: Write that article about BES too).

In the aq-tables, the payload of the message (i.e. the data transferred by the message) is stored in a custom type. The same goes for the event_data in wf_item_attribute_values. In this article we'll see how we can get the data from those 'strange' columns.

Let us start with wf_item_attribute_values. Processes that are started from a business event, store the data from the originating business event in the column 'EVENT_VALUE'. This has a type 'WF_EVENT_T'. When you query it in a sql*plus session, you will see a huge column filled with something like:

```
EVENT_VALUE(PRIORITY, SEND_DATE, RECEIVE_DATE, CORRELATION_ID, PARAMETER_LIST(NAME, VALUE), EVENT_NA
----------------------------------------------------------------------------------------------------
WF_EVENT_T(0, '17-SEP-09', NULL, NULL, WF_PARAMETER_LIST_T(WF_PARAMETER_T('TASK_ID', '16719879'), WF('ABORT_WORKFLOW', 'N'),
WF_PARAMETER_T('SUB_GUID', '73BAB9A51BAF5307E04400144F687CA0')), 'oracle.ap, NULL, NULL)
```

The problem that many ebs-dba's are facing is how to select the data inside this column. In this case, it would be the task_id that we are interested in. One way to do it is to use a clever substr/instr construction or a regular expression. But the efficient way to do it, is to tell Oracle which info you want.

Let's take a look at the custom type. As mentioned, it is WF_EVENT_T. We can find the description in the DBA_TYPES table.

```
select owner,type_name,typecode,attributes,methods
from dba_types
where type_name='WF_EVENT_T';
```

```
OWNER                          TYPE_NAME                      TYPECODE                       ATTRIBUTES    METHODS
------------------------------ ------------------------------ ------------------------------ ---------- ----------
APPS                           WF_EVENT_T                     OBJECT                                 13         31
```

This tells us that the WF_EVENT_T is an object type.
We can find its attributes in DBA_TYPE_ATTRS:

```
select owner,type_name,attr_name,attr_type_owner,attr_type_name,length
from dba_type_attrs
where type_name='WF_EVENT_T';
```

```
OWNER     TYPE_NAME          ATTR_NAME           ATTR_TYPE_OWNER     ATTR_TYPE_NAME
--------- ------------       ------------------- ----------------    --------------
APPS      WF_EVENT_T         CORRELATION_ID                          VARCHAR2
APPS      WF_EVENT_T         ERROR_MESSAGE                           VARCHAR2
APPS      WF_EVENT_T         ERROR_STACK                             VARCHAR2
APPS      WF_EVENT_T         ERROR_SUBSCRIPTION                      RAW
APPS      WF_EVENT_T         EVENT_DATA                              CLOB
```

```
APPS      WF_EVENT_T          EVENT_KEY                          VARCHAR2
APPS      WF_EVENT_T          EVENT_NAME                         VARCHAR2
APPS      WF_EVENT_T          FROM_AGENT          APPS           WF_AGENT_T
APPS      WF_EVENT_T          PARAMETER_LIST      APPS           WF_PARAMETER_LIST_T
APPS      WF_EVENT_T          PRIORITY                           NUMBER
APPS      WF_EVENT_T          RECEIVE_DATE                       DATE
APPS      WF_EVENT_T          SEND_DATE                          DATE
APPS      WF_EVENT_T          TO_AGENT            APPS           WF_AGENT_T
```

You see that the attributes are defined including their datatype, which can be a seeded datatype (VARCHAR2) or a custom one (WF_PARAMETER_LIST_T). Now that we now the attributes of the type, we can select them directly. To select the 'PRIORITY', just use an extra qualifier:

```
select v.event_value.priority
from   wf_item_attribute_values v
where  ROWNUM = 1;
```

```
EVENT_VALUE.PRIORITY
--------------------
                   0
```

But how about the 'PARAMETER_LIST'? That is where the task_id was stored. Let's check the WF_PARAMETER_LIST_T definition:

```
select owner,type_name,typecode,attributes,methods
from   dba_types
where  type_name ='WF_PARAMETER_LIST_T';
```

```
OWNER     TYPE_NAME           TYPECODE            ATTRIBUTES  METHODS
--------- ------------------- ------------------- ----------- -------
APPS      WF_PARAMETER_LIST_T COLLECTION                    0       0
```

This time, the type is a collection. We can find more info about a collection with:

```
select type_name,coll_type,elem_type_owner,elem_type_name
from   dba_coll_types
where  type_name='WF_PARAMETER_LIST_T';
```

```
TYPE_NAME                      COLL_TYPE                      ELEM_TYPE_OWNER                ELEM_TYPE_NAME
------------------------------ ------------------------------ ------------------------------ --------------
WF_PARAMETER_LIST_T            VARYING ARRAY                  APPS                           WF_PARAMETER_T
```

So the WF_PARAMETER_LIST_T is a Varray of WF_PARAMETER_T. Before we look at selecting from Varrays, we first check what WF_PARAMETER_T looks like:

```
select owner,type_name,typecode,attributes,methods
from   dba_types
where  type_name ='WF_PARAMETER_T';
```

```
OWNER     TYPE_NAME           TYPECODE  ATTRIBUTES  METHODS
--------- ------------------- --------- ---------- ----------
APPS      WF_PARAMETER_T      OBJECT             2          4
```

That is an object type again. So we select:

```
select owner,type_name,attr_name,attr_type_owner,attr_type_name
from   dba_type_attrs
where  type_name='WF_PARAMETER_T';
```

```
OWNER     TYPE_NAME           ATTR_NAME ATTR_TYPE_OWNER     ATTR_TYPE_NAME
--------- ------------------- --------- ------------------- --------------
APPS      WF_PARAMETER_T      NAME                          VARCHAR2
APPS      WF_PARAMETER_T      VALUE                         VARCHAR2
```

Ok. We now know the whole structure of the parameter list. Back to the Varray. A Varray (Virtual Array) is of course an array structure. Since this is similar to a table structure, you can cast the Varray into a table. Then you use the casted table to select your data. Let's do that to get the names from the parameter list.

```
select t.name
from   wf_item_attribute_values v
,      table(v.event_value.parameter_list) t
Where  v.event_value IS NOT NULL
And    ROWNUM = 1;
```

```
NAME
------------------------------
TASK_ID
```

Now that's a neat trick. We can join our table to its own column!
In our case, we only have a task_id parameter. We could do the same again, to get the value of the parameter from the value column.

But to join wf_item_attribute_values to itself is a very expensive operation. Take a look at the explain plan:

```
PLAN_TABLE_OUTPUT
```

```
-----------------------------------------------------------------------------------------------------------
| Id  | Operation                          |  Name                    | Rows  | Bytes | Cost  | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                          |     1 |    62 |   432M|       |       |
|*  1 |  COUNT STOPKEY                     |                          |       |       |       |       |       |
|   2 |   NESTED LOOPS                     |                          |   207G|    11T|   432M|       |       |
|   3 |    PARTITION RANGE ALL             |                          |       |       |       |     1 |    77 |
|   4 |     PARTITION HASH ALL             |                          |       |       |       |     1 |     8 |
|*  5 |      TABLE ACCESS FULL             | WF_ITEM_ATTRIBUTE_VALUES |    25M|  1453M|   948K|     1 |   616 |
|   6 |    COLLECTION ITERATOR PICKLER FETCH|                         |       |       |       |       |       |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(ROWNUM=1)
   5 - filter(SYS_OP_NOEXPAND("V"."EVENT_VALUE") IS NOT NULL)

Note: cpu costing is off
```

That can kind of hurt if it,were it not for the NL with a COUNT STOPKEY (Because of the rownum=1). The reason for the expensive plan, is because for every row from wf_item_attribute_values, Oracle needs to get the data from event_value. It basically does a carthesian join with itself.

So here is another way to get your data:

```
select v.item_key
,      (select value
        from table(v.event_value.parameter_list) t
        where t.name='TASK_ID' ) task_id
from   wf_item_attribute_values v
where  v.event_value is not null
and    rownum=1;
```

```
ITEM_KEY                                           TASK_ID
-------------------------------------------------- --------
oracle.apps.jtf.cac.task.createTask-155563676     16719879
```

Now we use a scalar subquery, where Oracle will access only the event_value for the rows that will be returned to the user. You can see this in the explain plan by the collection iterator picklefetch (= the operation that collects the data from a collection type) being pushed up to just before the select.

```
PLAN_TABLE_OUTPUT

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                          |  Name                    | Rows  | Bytes | Cost  | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                          |     1 |    60 |   948K|       |       |
|*  1 |  COLLECTION ITERATOR PICKLER FETCH|                          |       |       |       |       |       |
|*  2 |  COUNT STOPKEY                     |                          |       |       |       |       |       |
|   3 |   PARTITION RANGE ALL              |                          |       |       |       |     1 |    77 |
|   4 |    PARTITION HASH ALL              |                          |       |       |       |     1 |     8 |
|*  5 |     TABLE ACCESS FULL              | WF_ITEM_ATTRIBUTE_VALUES |    25M|  1453M|   948K|     1 |   616 |
-----------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SYS_OP_ATG(VALUE(KOKBF$),1,2,2)='TASK_ID')
   2 - filter(ROWNUM=1)
   5 - filter(SYS_OP_NOEXPAND("SYS_ALIAS_1"."EVENT_VALUE") IS NOT NULL)

Note: cpu costing is off
```

Still expensive. But much better.

So that will let us peek into the event_value column on wf_item_attribute values. Be aware that different processes will enter different parameters into the event_value column. But with the information above you should be able to understand how to select the data and use it in your own queries.

Now let’s take a look at the AQ-tables?

As I mentioned, they carry their payloads in custom types too. One that most ebs-dba’s will have seen is WF_NOTIFICATION_OUT. When the workflow engine generates a notification, it will be stored in

'WF_NOTIFICATIONS'. However, a message will be posted on 'WF_NOTIFICATION_OUT' too. This message will be read by the workflow notification mailer, which will use it to get the relevant data from WF_NOTIFICATIONS.

If you haven't run the notification mailer for a long time, you might want to clean up WF_NOTIFICATION_OUT a bit. There is a script available from Metalink to do the job. (There is also a quicker but unsupported way). But you might want to see the records in WF_NOTIFICATION_OUT related to WF_NOTIFICATIONS to decide if a cleanup is appropriate.

Let's look at WF_NOTIFICATION_OUT:

```
Name                  Null?    Type
----------------------- -------- ----------------
Q_NAME                          VARCHAR2(30)
MSGID                 NOT NULL RAW(16)
CORRID                          VARCHAR2(128)
PRIORITY                        NUMBER
STATE                           NUMBER
DELAY                           DATE
EXPIRATION                      NUMBER
TIME_MANAGER_INFO               DATE
LOCAL_ORDER_NO                  NUMBER
CHAIN_NO                        NUMBER
CSCN                            NUMBER
DSCN                            NUMBER
ENQ_TIME                        DATE
ENQ_UID                         NUMBER
ENQ_TID                         VARCHAR2(30)
DEQ_TIME                        DATE
DEQ_UID                         NUMBER
DEQ_TID                         VARCHAR2(30)
RETRY_COUNT                     NUMBER
EXCEPTION_QSCHEMA               VARCHAR2(30)
EXCEPTION_QUEUE                 VARCHAR2(30)
STEP_NO                         NUMBER
RECIPIENT_KEY                   NUMBER
DEQUEUE_MSGID                   RAW(16)
SENDER_NAME                     VARCHAR2(30)
SENDER_ADDRESS                  VARCHAR2(1024)
SENDER_PROTOCOL                 NUMBER
USER_DATA                       SYS.AQ$_JMS_TEXT _MESSAGE
```

The regular AQ-information is there. And our payload in USER_DATA. This time it's an AQ-type. Let's follow the same procedure:

```
select type_name,typecode,attributes,methods
from   dba_types
where  type_name='AQ$_JMS_TEXT_MESSAGE';
```

```
TYPE_NAME                     TYPECODE                       ATTRIBUTES    METHODS
------------------------------ ------------------------------ ---------- ----------
AQ$_JMS_TEXT_MESSAGE          OBJECT                                  4         34
```

An object type. So let's see its attributes:

```
select type_name,attr_name,attr_type_owner,attr_type_name
from   dba_type_attrs
where  type_name='AQ$_JMS_TEXT_MESSAGE';
```

```
TYPE_NAME                     ATTR_NAME                      ATTR_TYPE_OWNER                ATTR_TYPE_NAME
------------------------------ ------------------------------ ------------------------------ --------------
```

```
AQ$_JMS_TEXT_MESSAGE          HEADER                              SYS                            AQ$_JMS_HEADER
AQ$_JMS_TEXT_MESSAGE          TEXT_LEN                                                           INTEGER
AQ$_JMS_TEXT_MESSAGE          TEXT_LOB                                                           CLOB
AQ$_JMS_TEXT_MESSAGE          TEXT_VC                                                            VARCHAR2
```

We can already read the text_len, text_lob and text_vc. The last 2 contain an XML with a reference to the notification. But the information that I want to show you is in the header. This is a type 'AQ$_JMS_HEADER'. When we check this one, we see that it again is an object with these attributes:

```
select type_name,typecode,attributes,methods
from   dba_types
where  type_name='AQ$_JMS_HEADER';
```

```
TYPE_NAME                     TYPECODE                      ATTRIBUTES    METHODS
----------------------------- ----------------------------- ---------- ----------
AQ$_JMS_HEADER                OBJECT                                 7         31
```

```
select type_name,attr_name,attr_type_owner,attr_type_name
from   dba_type_attrs
where  type_name='AQ$_JMS_HEADER';
```

```
TYPE_NAME                     ATTR_NAME                     ATTR_TYPE_OWNER                ATTR_TYPE_NAME
----------------------------- ----------------------------- ------------------------------ -------
AQ$_JMS_HEADER                APPID                                                        VARCHAR2
AQ$_JMS_HEADER                GROUPID                                                      VARCHAR2
AQ$_JMS_HEADER                GROUPSEQ                                                     INTEGER
AQ$_JMS_HEADER                PROPERTIES                    SYS                            AQ$_JMS_USERPROPARRAY
AQ$_JMS_HEADER                REPLYTO                       SYS                            AQ$_AGENT
AQ$_JMS_HEADER                TYPE                                                         VARCHAR2
AQ$_JMS_HEADER                USERID                                                       VARCHAR2
```

As you can imagine, we need to drill down into 'PROPERTIES'. The other attributes might or might not contain any data, depending on the notification. properties is a Varray of 'AQ$_JMS_USERPROPERTY':

```
select type_name,typecode,attributes,methods
from   dba_types
where  type_name='AQ$_JMS_USERPROPARRAY';
```

```
TYPE_NAME                     TYPECODE                      ATTRIBUTES    METHODS
----------------------------- ----------------------------- ---------- ----------
AQ$_JMS_USERPROPARRAY         COLLECTION                             0          0
```

```
select type_name,coll_type,elem_type_owner,elem_type_name
from   dba_coll_types
where  type_name='AQ$_JMS_USERPROPARRAY';
```

```
TYPE_NAME                     COLL_TYPE                     ELEM_TYPE_OWNER                ELEM_TYPE_NAME
----------------------------- ----------------------------- ------------------------------ --------------------
AQ$_JMS_USERPROPARRAY         VARYING ARRAY                 SYS                            AQ$_JMS_USERPROPERTY
```

One more level to check:

```
select type_name,typecode,attributes,methods
from   dba_types
where  type_name='AQ$_JMS_USERPROPERTY';
```

```
TYPE_NAME                     TYPECODE                      ATTRIBUTES    METHODS
----------------------------- ----------------------------- ---------- ----------
```

```
AQ$_JMS_USERPROPERTY          OBJECT                              5         0
```

```
select type_name,attr_name,attr_type_owner,attr_type_name
from   dba_type_attrs
where  type_name='AQ$_JMS_USERPROPERTY';
```

```
TYPE_NAME                     ATTR_NAME                     ATTR_TYPE_OWNER               ATTR_TYPE_NAME
----------------------------- ----------------------------- ------------------------------ -------
AQ$_JMS_USERPROPERTY          JAVA_TYPE                                                    INTEGER
AQ$_JMS_USERPROPERTY          NAME                                                         VARCHAR2
AQ$_JMS_USERPROPERTY          NUM_VALUE                                                    NUMBER
AQ$_JMS_USERPROPERTY          STR_VALUE                                                    VARCHAR2
AQ$_JMS_USERPROPERTY          TYPE                                                         INTEGER
```

So let's see what properties we have. I just query the whole contents of properties for the first row in wf_notification_out (This particular system doesn't have a WF-mailer running. If it is running, chances are that you won't have any records in WF_NOTIFICATION_OUT).

```
select p.*
from   (select *
        from wf_notification_out
        where rownum=1) n
,      table(n.user_data.header.properties) p;
```

```
NAME                  TYPE  STR_VALUE                                    NUM_VALUE JAVA_TYPE
--------------------- ----- -------------------------------------------- --------- ---------
BES_EVENT_NAME          100 oracle.apps.wf.notification.send                      27
BES_EVENT_KEY           100                                               39388680        27
BES_PRIORITY            200                                                     50        23
BES_SEND_DATE           100 2009/03/06 01:12:23                                   27
BES_RECEIVE_DATE        100 2009/03/06 01:12:34                                   27
BES_FROM_AGENT          100 WF_NOTIFICATION_OUT@TESTDB.STIJF.COM                  27
BES_ERROR_SUBSCRIPTION  100 C10E7C2EF71253C1E0340800208D03E1                      27
NOTIFICATION_ID         100 39388680                                              27
ROLE                    100 FND_RESP535:21704                                     27
GROUP_ID                100 39388680                                              27
Q_CORRELATION_ID        100 XDPWFSTD                                              27
```

There you go. Among others, the 'NOTIFICATION_ID' is there. It maps to the notification_id on 'WF_NOTIFICATIONS'. And by now, we have seen enough to select it directly in our queries:

```
select n.msgid,(select str_value
                from   table(n.user_data.header.properties)
                       where  name='NOTIFICATION_ID') notification_id
from   wf_notification_out n
where  rownum=1;
```

```
MSGID                           NOTIFICATION_ID
------------------------------  ---------------
64672D4985E57075E04400144F687CA0        39388680
```

That concludes this article. Based on the above, you will be able to select the data you want.