Querying the OWB Repository – Individuelle Reports

Ute Middendorf
Metafinanz Informationssysteme GmbH
München

Schlüsselworte
Oracle Warehouse Builder Repository, Repository Browser, Public Views für Design und Runtime

Einleitung

Im Oracle Warehouse Builder Repository sind die Metadaten zu den im Design Center entwickelten ETL Strecken gespeichert. Darüber hinaus beinhaltet das OWB Repository Informationen zur Laufzeit von Mappings und Prozessflüssen, Audit Details, etc.

Auf diese Informationen lässt sich mit Hilfe des Repository Browsers zugreifen, doch dieser Zugriff bringt einige Nachteile mit sich. Zum Beispiel ist es nicht praxistauglich eine große Anzahl an ETL-Strecken lediglich mit Hilfe einer GUI zu überwachen.

Zum Glück gibt es eine Alternative!

Die Grundlagen des Repository Browsers bilden die OWB Public Views. Auf diese Public Views für Design und Runtime Metadaten kann man mittels SQL zugreifen und sich so einfach individuelle Reports generieren.

Somit ist es möglich projektspezifische Runtime Analysen durchzuführen, deren Ergebnisse dann auch als Status-EMail versendet werden können.

In diesem Vortragsmanuskript wird erklärt, wie man auf die Public Views des OWB Repository mit native SQL zugreifen kann. Es wird ein Überblick über die Public Views des Design- und Runtime Repositories gegeben, sowie auf einige praxisrelevante Views eingegangen.

Beispiel OWB Umgebung

Zur Verdeutlichung der Public Views wird die folgende kleine OWB Umgebung verwendet:

Workspace Owner:	doag_wsowner
Workspace:	doag
DB User:	doag
Mapping 1:	map_doag_1
	- Join der Customers und Countries Tabellen aus dem sh-Schema
	- Split nach Geschlecht in Female und Male Tabellen

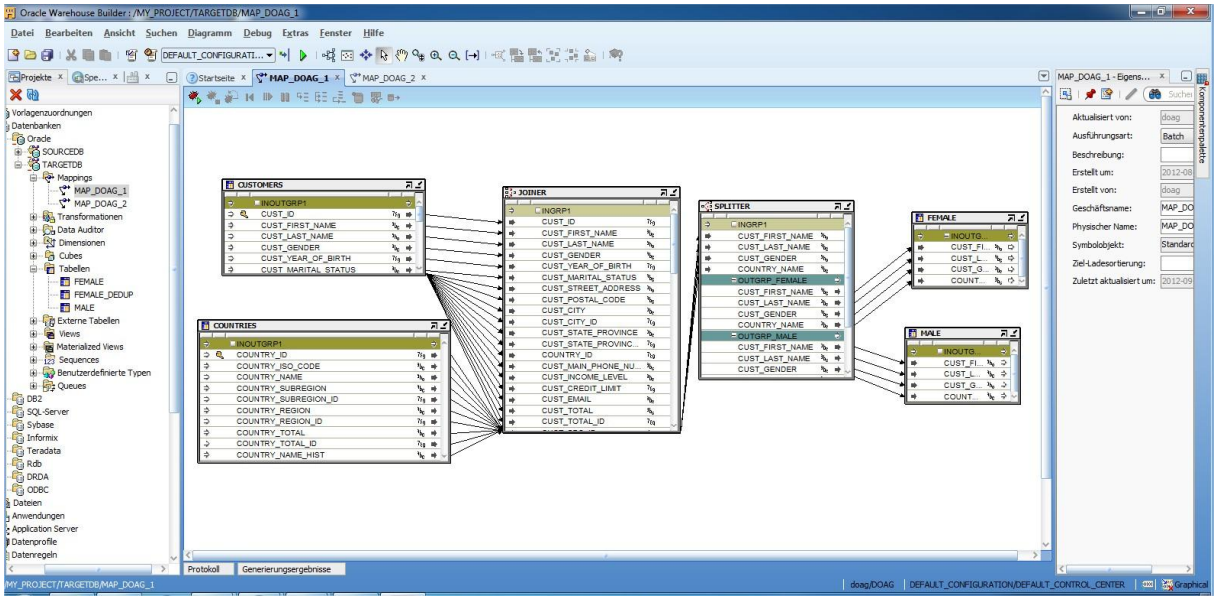

Abb 1: MAP_DOAG_1

Mapping 2: map_doag_2

- Deduplikator von der Female Tabelle zur Tabelle Female_Dedup
- Die Spalte Cust_First_Name der Female_Dedup Tabelle darf nicht NULL sein
- Die Spalten Cust_First_Name und Cust_Gender sind nicht in den Deduplikator hineingezogen

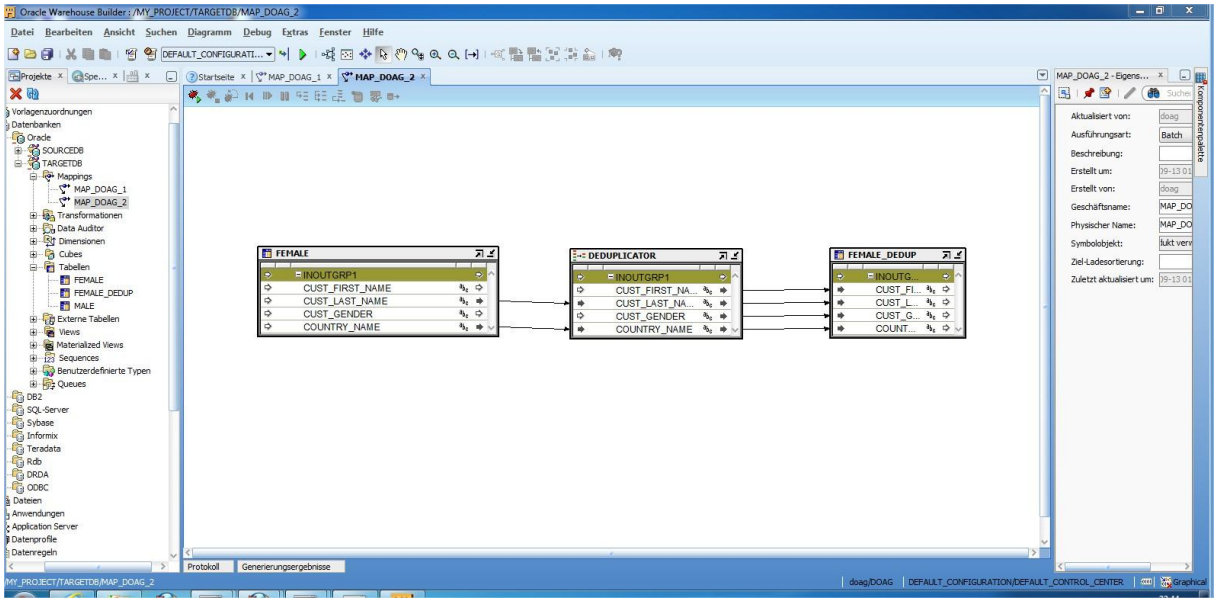

Abb 2: MAP_DOAG_2

Voraussetzung / Vorbereitung

Um als Workspace Owner auf die Public Views des Repository Browsers zugreifen zu können sind keine weiteren Voraussetzungen nötig. Aus Sicherheitsgründen sollte sich aber nicht jeder Anwender zur reinen Report Generierung als Workspace Owner an die Datenbank anmelden. Damit ein „normaler“ Datenbank-User die Public Views abfragen kann muss diesem User die Rolle ACCESS_PUBLICVIEW_BROWSER zugewiesen werden. Ohne diese Rolle wird jede SQL Abfrage der Public Views grundsätzlich „0 rows returned“ zurückgegeben.

Die ACCESS_PUBLICVIEW_BROWSER Rolle kann als Workspace Owner im OWB Design Center erteilt werden. Dazu im Global Explorer bzw Global Navigator erst den Sicherheit / Security Zweig und dann den Benutzer / User Zweig erweitern. Anschliessend den User auswählen, der Zugriff auf die Repository Browser Public Views bekommen soll. Mit einem Rechtsklick den Benutzer bearbeiten (Öffnen / Open) und im Reiter „System Privilegien“ das ACCESS_PUBLICVIEW_BROWSER Recht anhaken. Mit OK bestätigen und die Änderung mittels „alles speichern“ festschreiben.

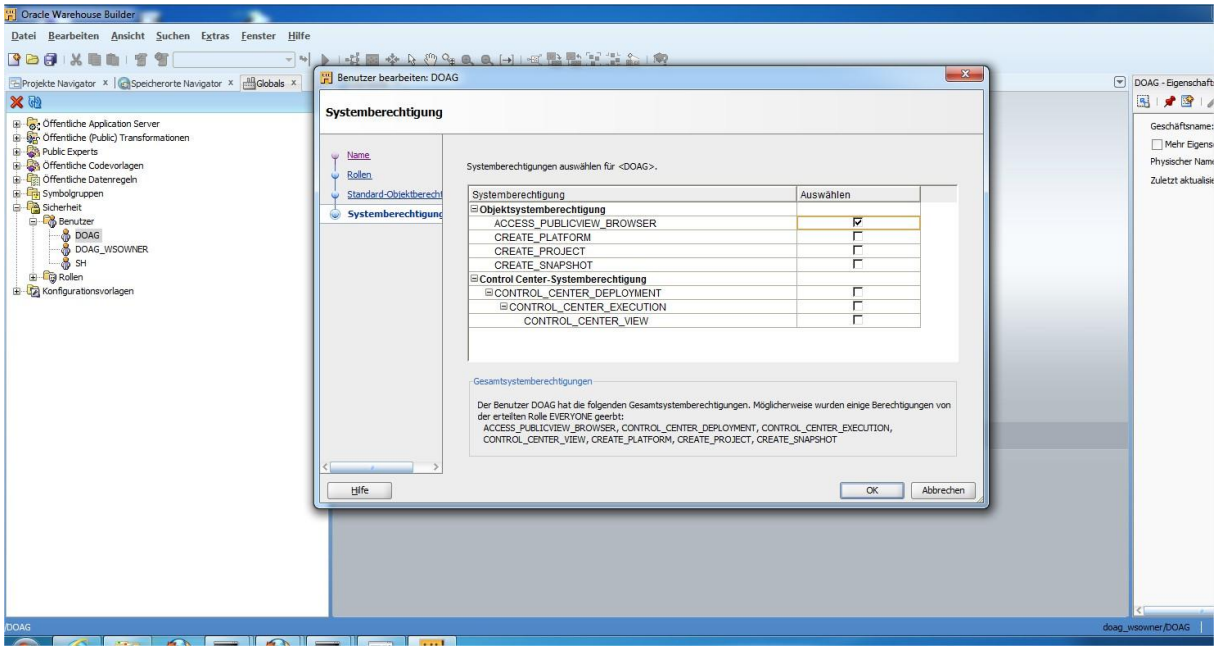

Abb 3: Grant ACCESS_PUBLICVIEW_BROWSER

Für den Fall, dass es mehr als einen Workspace gibt und die Abfragen nicht in dem Default-Workspace laufen sollen, so muss noch zu Beginn der SQL-Session der Workspace mit Hilfe der Prozedure WB_workspace_management.set_workspace(<Workspace Name>,<Workspace Owner>) gesetzt werden, z.B.:

```
SQL> connect doag/doag@ORCL

SQL> exec
owbsys.WB_workspace_management.set_workspace('doag','doag_wsowner');
```

Design Environment

Zum Design Environment zählen mehr als 200 Public Views. Diese lassen sich in die folgenden Bereiche unterteilen:

- General Model Views
- Data Model Views
- Flat Files Views
- Collections Views
- Function Model Views
- Configuration Model Views
- Deployment Model Views
- Mapping Model Views
- Process Flow Model Views
- Profiling Views
- Data Rules Views
- User Defined Object Views
- Expert Views
- Business Intelligence Views
- Real Time Views
- Scheduling Views
- Security Views
- Code Template Views
- Web Services Views
- Others

Ein guter Start zum Erforschen der Public Views des Design Environment bilden die Mapping Model Views. Um sich einen Überblick über die bestehenden Mappings eines Workspaces zu verschaffen kann die View ALL_IV_XFORM_MAPS abgefragt werden.

```
SQL> SELECT map_id, map_name, is_valid, updated_when, created_when,
updated_by, created_by

FROM all_iv_xform_maps;

MAP_ID   MAP_NAME   IS_VALID  UPDATED_ CREATED_ UPDATED_BY CREATED_BY
-------- ---------- --------- -------- -------- ---------- ----------------
79529    MAP_DOAG_1 Y         13.09.12 26.08.12 doag       doag
80400    MAP_DOAG_2 Y         13.09.12 13.09.12 doag       doag
```

Meist ist man aber nicht nur an einer Übersicht der einzelnen Mappings interessiert, sondern möchte auch noch die einzelnen Bestandteile der Mappings wissen. Für diesen Bericht kann die ALL_IV_XFORM_MAPS View mit der View ALL_IV_XFORM_MAP_COMPONENTS verknüpft werden.

```
SQL> SELECT maps.map_name, comp.map_component_id, comp.map_component_name,
comp.operator_type

FROM all_iv_xform_maps maps

JOIN all_iv_xform_map_components comp
```

```
ON maps.map_id = comp.map_id;


MAP_NAME    MAP_COMPONENT_ID  MAP_COMPONENT_NAME  OPERATOR_TYPE
---------- ------------------ ------------------- ------------------------
MAP_DOAG_1  79616              COUNTRIES           Table
MAP_DOAG_1  79662              JOINER              Join
MAP_DOAG_1  79541              CUSTOMERS           Table
MAP_DOAG_1  79940              SPLITTER            Splitter
MAP_DOAG_1  80023              FEMALE              Table
MAP_DOAG_1  80049              MALE                Table
MAP_DOAG_2  80412              FEMALE              Table
MAP_DOAG_2  80439              DEDUPLICATOR        Distinct
MAP_DOAG_2  80463              FEMALE_DEDUP        Table
```

Runtime Environment

Bei den Views der Runtime Environment wird zwischen den Deployment Auditing Views und den Execution Auditing Views unterschieden.

Mit Hilfe der Deployment Auditing Views lässt sich zum Beispiel Nachweisen, wann und durch welchen User Änderungen in die Produktions-Umgebung eingespielt wurden.

```
SQL> SELECT deployment_audit_name AS name, repository_user AS user,
generation_time AS gen_time, deployment_audit_status AS status
FROM all_rt_audit_deployments
WHERE deployment_audit_name LIKE '%DOAG_1';

NAME           REP_USER       GEN_TIME       STATUS
-------------- -------------- -------------- ------------------------------
MAP_DOAG_1     doag           26.08.12       COMPLETE
MAP_DOAG_1     doag           26.08.12       COMPLETE
MAP_DOAG_1     doag_wsowner   10.09.12       COMPLETE
```

Die Public Views, die zu dem Bereich der Execution Auditing Views zählen, unterstützen bei der Prüfung der Ausführung von Mappings. Neben Ausführungszeiten können auch Informationen über die Anzahl von verarbeiteten Datensätzen in einem individuellen Report aufgenommen werden.

```
SQL> SELECT map_name AS name, start_time, elapse_time AS elapse,
number_errors AS NE, run_status AS status, number_records_selected AS
rec_sel, number_records_inserted AS rec_ins, number_records_updated AS
rec_upd

FROM all_rt_audit_map_runs;

NAME        START_TI ELAPSE  NE STATUS    REC_SEL  REC_INS  REC_UPD
----------- -------- ------- -- --------- -------- -------- ----------
MAP_DOAG_2  13.09.12    0     0  COMPLETE    7333     7333       0
MAP_DOAG_2  13.09.12    5    51  COMPLETE    1000        0       0
MAP_DOAG_1  26.08.12    7     0  COMPLETE   55500    55500       0
```

Je nach Art der betrachteten Mappings kann man noch die Anzahl der gelöschten (number_records_deleted), der gemergeten (number_records_merged) oder bei einer SQL-Loader Ladung abgewiesenen (number_records_discarded) Datensätze mit in die Abfrage aufnehmen.

Über diese rein informativen Abfragen hinaus lassen sich Execution Auditing Views auch zur Fehleranalyse heranziehen. Die View ALL_RT_AUDIT_MAP_RUN_ERRORS beinhaltet Fehlermeldungen zu einem Mapping. Mit dem folgendem Statement lässt sich ein Bericht über die gelaufenen Mappings eines Tages inklusive etwaiger Fehlermeldungen generieren. Gegebenenfalls mehrfach auftretende Fehler werden nur einmal gelistet. Die Spalte err.run_error_number gibt Auskunft über die Häufigkeit der einzelnen Fehler während die Spalte run_error_message die ORA-Fehlermeldung enthält.

```
SQL> SELECT distinct runs.map_run_id AS id, runs.map_name, runs.start_time,
runs.end_time, runs.number_errors AS ne, runs.run_status AS status,
err.run_error_number AS err_no, err.run_error_message, err.target_name

FROM all_rt_audit_map_runs runs
LEFT OUTER JOIN all_rt_audit_map_run_errors
ON err.map_run_id = runs.map_run_id
WHERE trunc(runs.start_time) = trunc(sysdate)
ORDER BY start_time DESC, end_time DESC;

ID MAP_NAME    START_TI END_TIME  NE STATUS    ERR_NO       ERROR_MESSAGE
-- --------    -------- --------  -- ------    ------       ---------------
--------------------------------------------------------------------------------
TARGET_NAME
-----------
22 MAP_DOAG_2  13.09.12 13.09.12  51 COMPLETE  -1400        ORA-01400:
Einfügen von NULL in ("SH"."FEMALE_DEDUP"."CUST_FIRST_NAME") nicht möglich
FEMALE_DEDUP

21 MAP_DOAG_2  13.09.12 13.09.12   0 COMPLETE
```

Mit Hilfe der Public View ALL_RT_AUDIT_EXECUTIONS lassen sich die Laufzeiten einzelner Ausführungen im Vergleich betrachten. Diese Information kann man dafür nutzen mögliche Kandidaten für ein Tuning zu identifizieren.

```
SQL> SELECT object name AS name, trunc(min(elapse_time)) AS min,
trunc(max(elapse_time)) AS max, trunc(sum(elapse_time)) AS sum,
trunc(avg(elapse_time)) AS avg, count(elapse_time) AS count

FROM all_rt_audit_executions
WHERE 1=1
AND task_type = 'PLSQL'
AND created_on >= to_date('01.08.2012','DD.DD.YYYY')
GROUP BY object_name
ORDER BY avg DESC;


NAME              MIN        MAX        SUM     AVG         COUNT
----------------- ---------- ---------- ---------- ---------- ----------
MAP_DOAG_2         7          11         18         9           2
MAP_DOAG_1         3          14         17         8           2
```

Bei den bisher erwähnten Public Views der Runtime Environment handelt es sich nur um einen Bruchteil der zur Verfügung stehenden Views. Die Views der Runtime Environment im Einzelnen lauten:

1. Deployment Auditing View

   - ALL_RT_AUDIT_LOCATIONS
   - ALL_RT_AUDIT_LOCATION_MESSAGES
   - ALL_RT_AUDIT_LOCATION_FILES
   - ALL_RT_AUDIT_OBJECTS
   - ALL_RT_AUDIT_SCRIPT_MESSAGES
   - ALL_RT_AUDIT_SCRIPT_RUNS
   - ALL_RT_AUDIT_SCRIPT_FILES
   - ALL_RT_AUDIT_DEPLOYMENTS
   - ALL_RT_INSTALLATIONS
   - ALL_RT_LOCATIONS
   - ALL_RT_LOCATION_PARAMETERS
   - ALL_RT_OBJECTS
   - ALL_RT_TASKS
   - ALL_RT_TASK_PARAMETERS

2. Execution Auditing Views

   - ALL_RT_AUDIT_EXECUTIONS
   - ALL_RT_AUDIT_EXECUTION_PARAMS
   - ALL_RT_AUDIT_EXEC_MESSAGES
   - ALL_RT_AUDIT_EXEC_FILES
   - ALL_RT_AUDIT_MAP_RUNS
   - ALL_RT_AUDIT_MAP_RUN_SOURCES
   - ALL_RT_AUDIT_MAP_RUN_TARGETS
   - ALL_RT_AUDIT_STEP_RUNS
   - ALL_RT_AUDIT_STEP_RUN_SOURCES
   - ALL_RT_AUDIT_STEP_RUN_TARGETS
   - ALL_RT_AUDIT_MAP_RUN_ERRORS
   - ALL_RT_AUDIT_MAP_RUN_TRACE
   - ALL_RT_AUDIT_PROC_RUN_ERRORS
   - ALL_RT_AUDIT_STEP_RUN_STRUCTS

Fazit

Der Repository Browser ist komplett auf die Public Views aufgebaut. Jede im Repository Browsers angezeigt Information lässt sich somit auch mit native SQL generieren. Die Public Views ermöglichen es einem Berichte über das Design oder Runtime-Analysen ganz nach den eigenen Bedürfnissen zusammen zu stellen. Dadurch, dass man nur die Daten rausfiltern kann, die eine wirklich interessieren, erhält man gerade auch für große ETL-Umgebungen verwendbare Reports. Für die

Generierung individueller Reports mit Hilfe der Public Views ist es nicht notwendig, dass man sich erst durch die GUI klicken muss. Daher lassen sich diese individuellen Reports problemlos automatisieren. Einer Einbindung ins Scheduling sowie einem automatisierten EMail-Versand der Abfrage-Ergebnisse steht nichts mehr im Wege.

Es lohnt sich, sich mit den Public Views zu beschäftigen!

Quellen / Weiterführende Literatur

- Oracle® Warehouse Builder API and Scripting Reference 11g Release 2 (11.2)
- https://blogs.oracle.com/warehousebuilder/entry/owb_public_views_in_11g_release_2

Kontaktadresse:
Ute Middendorf
Metafinanz Informationssysteme GmbH
Leopoldstrasse 140
D-80404 München

Telefon:	+49 (0) 89-360 531 5167
Fax:	+49 (0) 89-360 531 15
E-Mail	ute.middendorf@metafinanz.de
Internet:	www.metafinanz.de