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. Dieser Artikel stellt als Alternative zum Repository Browser die sogenannten „Public Views" vor.

OWB-Repository – individuelle Reports

Ute Middendorf, metafinanz-Informationssysteme GmbH

Eine einfache Möglichkeit zum Zugriff auf die Warehouse-Builder-Repository-Informationen ist der Repository-Browser. Voraussetzung für dessen Nutzung ist jedoch, dass ein entsprechender Listener-Prozess gestartet ist. Darüber hinaus wird eine Freischaltung auf den Port des Repository-Browser-Listener benötigt. Ein Nachteil dabei ist, dass man auf die zahlreichen Informationen des OWB-Repository nur mit den vorgegebenen Reports zugreifen kann.

Grundlage des Repository-Browsers sind die OWB Public Views. Auf die Public Views für Design- und Runtime-Metadaten kann man mittels SQL zugreifen. Diese alternative Schnittstelle zu den Repository-Informationen ermöglicht die einfache Generierung individueller Reports.

Notwendige Berechtigung

Der Repository-Workspace-Owner besitzt alle notwendigen Berechtigungen, um mit SQL die Public Views abzufragen. In der Praxis ist es sicherlich nicht gewollt, dass alle Endanwender den Zugriff auf die Public Views durch den Logon als Workspace-Owner bekommen. Um einem normalen Datenbank-Benutzer die Abfrage der Public Views zu ermöglichen, muss er die Rolle „ACCESS_PUBLICVIEW_BROWSER" erhalten (siehe Abbildung 1). Ohne diese Rolle wird jede SQL-Abfrage der Public Views grundsätzlich mit „Es wurden keine Zeilen ausgewählt"/„0 rows returned" zurückgeben.

Die „ACCESS_PUBLIC_VIEW"-Rolle kann als Workspace-Owner im OWB Design Center zugewiesen werden. Dazu im Global Navigator erst den Sicherheits- und dann den Benutzer-Zweig erweitern; anschließend den Benutzer auswählen, der Zugriff auf die Repository-Browser-Public-Views bekommen soll. Mit einem Rechtsklick den Benutzer bearbeiten (Öffnen) und im Reiter „System Privilegien" das „ACCESS_PUBLICVIEW_BROWSER"-Recht aktivieren.

Für den Fall, dass es mehr als einen Workspace gibt und die Abfragen nicht in dem Default-Workspace lau-

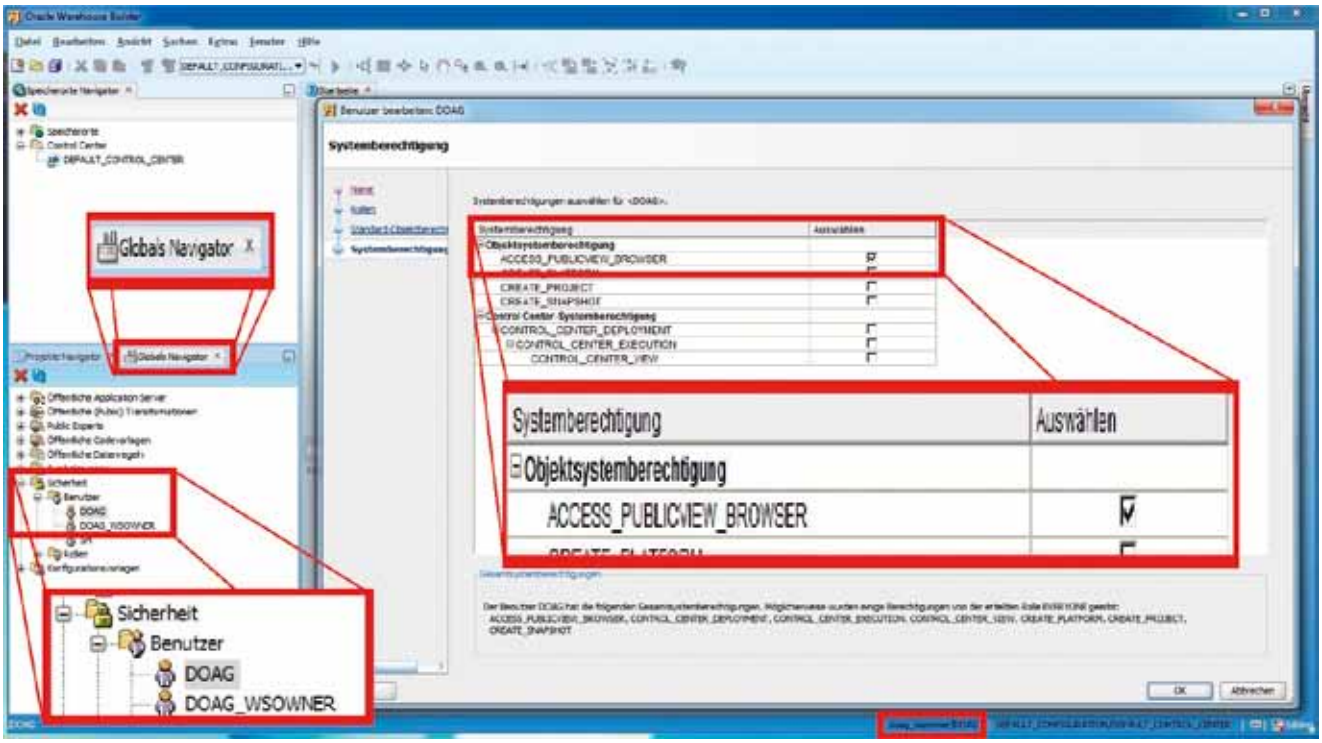

Abbildung 1: „ACCESS_PUBLICVIEW_BROWSER"-Berechtigung vergeben

fen sollen, muss noch zu Beginn der SQL-Session der Workspace mithilfe der Prozedur „WB_workspace_management.set_workspace (<Workspace Name>,<Workspace Owner>)“ gesetzt werden (siehe Listing 1).

Beispiel-Szenario

In den folgenden Abschnitten werden einige der Public Views vorgestellt. Dies geschieht anhand der zwei Mappings „MAP_DOAG_1“ und „MAP_DOAG_2“ (siehe Abbildung 2 und 3). Im ersten Mapping werden die „CUSTOMERS“- und die „COUNTRIES“-Tabellen des SH-Schemas miteinander gejoint. Anschließend findet ein Split nach Geschlechtern in zwei Tabellen „FEMALE“ und „MALE“ statt.

Das zweite Mapping besteht aus einem Deduplikator der Tabelle „FEMALE“ zur „FEMALE_DEDUP“-Tabelle. Die Besonderheit in diesem Mapping ist, dass die „FEMALE“-Input-Tabelle im Mapping „FEMALE_IN“ heißt. Zudem wurde die „FEMALE_DEDUP“-Tabelle so angelegt, dass die Spalte „CUST_FIRST_NAME“ nicht „NULL“ sein darf. Die Spalten „CUST_FIRST_NAME“ und „CUST_GENDER“ wurden absichtlich nicht in den Deduplikator hineingezogen. Das Mapping endet mit einem Fehler.

Zum Design Environment zählen mehr als 200 Public Views, die sich in folgende Bereiche unterteilen lassen:

- General Model Views
- Data Model Views
- Flat Files Views
- Collection 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

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

Listing 1

```
SELECT map_id, map_name, is_valid as V, updated_when as U_WHEN,
created_when as C_WHEN, updated_by
FROM all_iv_xform_maps;
```

MAP_ID	MAP_NAME	V	U_WHEN	C_WHEN	UPDATED_BY
79529	MAP_DOAG_1	Y	13.09.12	26.08.12	doag
80400	MAP_DOAG_2	Y	13.09.12	13.09.12	doag

Listing 2

```
SELECT maps.map_name, maps.is_valid as V, comp.map_component_id as 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
ORDER BY 1,2;
```

MAP_NAME	V	ID	MAP_COMPONENT_NAME	OPERATOR_TYPE
MAP_DOAG_1	Y	79541	CUSTOMERS	Table
MAP_DOAG_1	Y	79616	COUNTRIES	Table
MAP_DOAG_1	Y	79662	JOINER	Join
MAP_DOAG_1	Y	79940	SPLITTER	Splitter
MAP_DOAG_1	Y	80023	FEMALE	Table
MAP_DOAG_1	Y	80049	MALE	Table
MAP_DOAG_2	Y	80412	FEMALE_IN	Table
MAP_DOAG_2	Y	80439	DEDUPLICATOR	Distinct
MAP_DOAG_2	Y	80463	FEMALE_DEDUP	Table

Listing 3

```
SELECT map_name, map_component_name, data_entity_name
FROM all_iv_xform_map_components
WHERE operator_type = ,Table'
AND data_entity_name = ,FEMALE';
```

MAP_NAME	MAP_COMPONENT_NAME	DATA_ENTITY_NAME
MAP_DOAG_1	FEMALE	FEMALE
MAP_DOAG_2	FEMALE_IN	FEMALE

Listing 4

Ein guter Start zum Erforschen der Public Views bilden die „Mapping Model Views“. Um sich einen Überblick über die bestehenden Mappings eines Workspace zu verschaffen, wird die View „ALL_IV_XFORM_MAPS“ abgefragt (siehe Listing 2).

Meist ist man jedoch nicht nur an einer Übersicht über die einzelnen Mappings interessiert, sondern möch-

```
SELECT deployment_audit_name AS name, repository_user AS user, gene-
ration_time AS gen_time, deployment_audit_status AS status
FROM all_rt_audit_deployments
WHERE deployment_audit_name LIKE ‚%DOAG_1‘;
```

NAME	USER	GEN_TIME	STATUS
MAP_DOAG_1	doag	26.08.12	COMPLETED
MAP_DOAG_1	doag	26.08.12	COMPLETED
MAP_DOAG_1	doag_wsowner	10.09.12	COMPLETED

Listing 5

```
SELECT map_name AS name,
       start_time AS start,
       elapse_time AS e,
       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	E	NE	STATUS	REC_ SEL	REC_ INS	REC_ UPD
MAP_DOAG_2	12.09.12	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

Listing 6

```
SELECT runs.map_name as name, runs.start_time as start, runs.number_
errors AS ne, executions.return_code AS rc, executions.return_result
AS r_result,          executions.number_task_errors AS no_error,
 executions.number_task_warnings AS no_warn
FROM all_rt_audit_map_runs runs
LEFT OUTER JOIN all_rt_audit_executions executions
ON runs.execution_audit_id = executions.execution_audit_id
ORDER BY map_name DESC, start_time DESC;
```

NAME	START	NE	RC	R_RESULT	NO_ERROR	NO_ WARN
MAP_DOAG_2	13.09.12	51	1	FAILURE	0	51
MAP_DOAG_2	12.09.12	0	0	OK	0	0
MAP_DOAG_1	26.08.12	0	0	OK	0	0

Listing 7

te auch noch deren einzelne Bestandteile kennen. Für diesen Bericht kann die „ALL_IV_XFORM_MAPS“-View mit der View „ALL_IV_XFORM_MAP_COMPONENTS“ verknüpft werden (siehe Listing 3).

Diese Abfrage listet die Komponenten der Mappings genauso, wie sie im OWB zu sehen sind. Für die Komponente „80412“ wird der Name „FEMALE_IN“ angezeigt. Der tatsächliche Name der Tabelle innerhalb der Datenbank steht in der Spalte „DATA_ENTITY_NAME“ der „ALL_IV_XFORM_MAP_COMPONENTS“-View. Mithilfe dieser Spalte lässt sich eine Übersicht über alle Mappings erstellen, die von der Änderung der Struktur einer Tabelle, zum Beispiel „FEMALE“, betroffen sind (siehe Listing 4).

Runtime Environment

Im Gegensatz zum Design Environment ist die Anzahl der Public Views über das Runtime Environment überschaubar. Es gibt je 14 Views zu den Themengebieten „Deployment“ und „Execution“:

Deployment Audit Views
- 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

Execution Audit 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

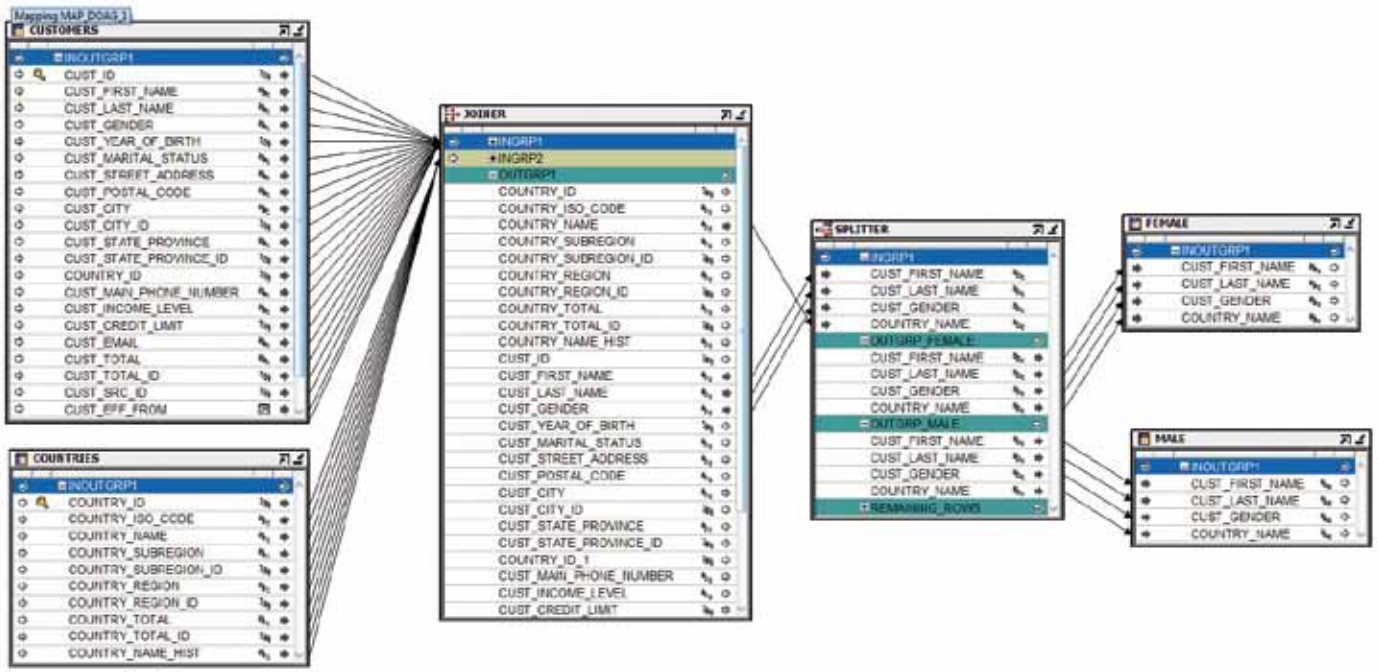

Abbildung 2: Mapping map_doag_1

Im Zusammenhang mit Auditing stellt sich oft die Frage: „Wann wurde durch wen deployt?" Eine Historie von Deployments kann durch eine Abfrage erstellt werden (siehe Listing 5).

Die Public Views, die zum 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 einen individuellen Report aufgenommen werden (siehe Listing 6).

Bei diesem Report ist zu beachten, dass trotz der 51 Fehler während der Ausführung des zweiten Mappings der Status „COMPLETE" angezeigt wird. Der Run-Status gibt also keine Auskunft über den Erfolg oder Misserfolg der Ausführung, sondern lediglich darüber, ob die Ausführung beendet ist oder nicht. Je nach Art der betrachteten Mappings kann man noch die Anzahl der gelöschten (number_records_deleted), der gemischten (number_records_merged) oder der bei einer SQL-Loader-Ladung abgewiesenen (number_records_discarded) Datensätze mit in die Abfrage aufnehmen. Return-Code-Informationen lassen sich mithilfe der View „ALL_RT_AUDIT_EXECUTIONS"

```
SELECT distinct runs.map_name as name, runs.start_time as start,
runs.number_errors AS ne, err.target_name as tname, err.run_error_
number AS err_no, err.run_error_message as err_messsage
FROM all_rt_audit_map_runs runs
LEFT OUTER JOIN all_rt_audit_map_run_errors err
ON err.map_run_id = runs.map_run_id
WHERE trunc(runs.start_time) >
  to_date('11.09.2012','dd.mm.yyyy')
ORDER BY start_time ASC;
```

NAME	START	NE	TNAME	ERR_NO	ERR_MESSAGE
MAP_DOAG_2	12.09.12	0			
MAP_DOAG_2	13.09.12	51	FE-MALE_ DEDUP	-1400	ORA-01400: Einfügen von NULL in („SH"."FEMALE_ DEDUP"."CUST_ FIRST_NAME") nicht möglich

Listing 8

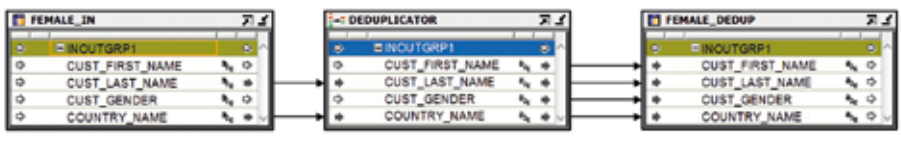

Abbildung 3: Mapping map_doag_2

Vorschau auf die nächste Ausgabe

Das Schwerpunktthema der Ausgabe 03/2013 lautet
Oracle Best Practices auf Infrastrukturen und Plattformen
Sie erscheint am 7. Juni 2013

zusammenstellen (siehe Listing 7). Über diese rein informativen Abfragen hinaus lassen sich Execution Auditing Views auch zur Fehler-Analyse heranziehen. Die View „ALL_RT_AUDIT_MAP_RUN_ERRORS“ beinhaltet Fehlermeldungen zu einem Mapping. Mit dem Statement in Listing 8 lässt sich ein Bericht über die gelaufenen Mappings seit einem bestimmten Tag (alternativ auch „sysdate“) 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.

Fazit

Es gibt zahlreiche Anwendungsbeispiele, wie die individuellen SQL-Reports der Public Views genutzt werden können. Hierzu zählen zum Beispiel die Produktionsüberwachung inklusive Einbindung von Monitoring, Fehler-Analyse, technischer Freigabe, Impact-Analyse, Deployment-Historie und Überprüfung von Namenskonventionen.

Der Repository-Browser ist komplett auf die Public Views aufgebaut. Jede im Repository-Browser angezeigte Information lässt sich somit auch mit SQL-Abfragen generieren. Die Public Views ermöglichen es, Berichte über das Design oder Runtime-Analysen ganz nach den eigenen Bedürfnissen zusammenzustellen. Durch die Möglichkeit, nur relevante Daten herauszufiltern, erhält man gerade auch für große ETL-Umgebungen anwendbare Reports. Diese individuellen Reports lassen sich problemlos automatisieren und ins Scheduling einbinden.

Ute Middendorf
ute.middendorf@metafinanz.de