Edition Based Redefinition: Versionsverwaltung für Datenbankobjekte

Daniel Horwedel, merlin.zwo InfoDesign GmbH & Co. KG

Entwickler haben häufig die Anforderung, dass die Aktualisierung von Anwendungen ohne große Wartungsfenster erfolgen soll. Darüber hinaus kann es sinnvoll sein, neue Versionen von Datenbankobjekten innerhalb eines bestehenden Schemas zu testen und die Auswirkungen auf abhängige Objekte zu analysieren.

Oracle bietet zur Lösung dieser Problematik auf der Datenbank-Ebene ein einfach nutzbares Werkzeug an: Edition Based Redefinition (EBR). Damit lassen sich mehrere Versionen eines Datenbankobjekts im Schema installieren und Session-abhängig verwenden, wodurch ein Update auf eine neue Version der Datenbankobjekte oder ein Test der neuen Version ohne Beeinträchtigung des laufenden Betriebs erfolgen kann. Den Anwendern steht in ihrer Session die alte Version der Datenbankobjekte zur Verfügung, während die neue installiert und getestet werden kann. Mit einem einfachen „ALTER SESSION"-Befehl kann anschließend auf die neue Version umgestellt werden.

Szenario

Die Aktualisierung bestehender Datenbank-Anwendungen im laufenden Betrieb ist mit einigen Gefahren verbunden. Zunächst einmal muss eine Downtime eingeplant werden, um die aktualisierten Datenbank-Objekte zu installieren und zu kompilieren.

Je nach Art der Anwendung ist eine Unterbrechung der Verfügbarkeit nicht möglich, sodass bislang aufwändige Lösungen benötigt wurden, um ein Anwendungs-Upgrade ohne Unterbrechung des laufenden Betriebs durchführen zu können. Zudem kann auch nach intensivem Testen der Anwendung nicht in jedem Falle ausgeschlossen werden, dass auf dem Produktiv-System unvorhergesehene Probleme, Seiteneffekte oder datenbezogene Fehler auftreten.

Eine gängige Möglichkeit zum Testen umfangreicher Änderungen stellt die Verwendung einer Schema-Kopie dar, die auf Basis von Produktivdaten die Auswirkungen einer Änderung der Datenstruktur ermittelt. Dabei entsteht allerdings durch das Vorhalten mehrerer Testumgebungen ein gewaltiger Aufwand, wodurch im Endeffekt oftmals veraltete Testdaten zur Verfügung stehen oder die Testschemata unterschiedliche Versionsstände bei den enthaltenen Datenbankobjekten aufweisen, wodurch kein verlässlicher finaler Abnahmetest möglich ist.

Einen effizienten Lösungsweg für diese Problematik bietet das Online Application Upgrade, dessen Zielsetzung die Durchführung einer Aktualisierung von Datenbankobjekten möglichst ohne Downtime ist. Sein Grundprinzip zeichnet sich dadurch aus, dass der Benutzer zunächst auf der bestehenden Programmversion weiterarbeiten kann, bis die (parallel zu installierende) neue Version fertig installiert, getestet und freigegeben ist. Sobald dies geschehen ist, werden die Benutzer quasi per Knopfdruck auf die neue Anwendungsversion umgestellt.

Oracle bietet zur Durchführung dieser Online Application Upgrades seit der Version 11*g* R2 über Edition Based Redefinition ein umfangreiches Werkzeug an, das mit Erscheinen der Version 12*c* erheblich erweitert und verbessert wurde. Es lässt sich mit allen Editionen der Datenbank kostenfrei nutzen, sodass diese Funktionalität nicht nur den Anwendern der Enterprise Edition vorbehalten, sondern auch schon ab der Express Edition einsetzbar ist.

Versionierung der Datenbankobjekte als Lösungsweg

Zur Reduzierung des Aufwands beim Betrieb mehrerer Test- und Entwicklungsumgebungen sowie der Durchführung von Online Application Upgrades, also der Bereitstellung neuer Anwendungsversionen ohne Downtime, bietet sich eine „Versionierung" der Datenbankobjekte mittels Edition Based Redefinition an. Dabei werden mehrere Versionen eines Datenbankobjekts in Kombination mit der Information, zu welcher Edition das jeweilige Objekt gehört, innerhalb eines Schemas vorgehalten. Zwischen den einzelnen Editionen kann nun innerhalb des Session-Kontextes gewechselt werden. Dies reduziert den Aufwand für die Pflege und Bereitstellung von Testumgebungen deutlich, da für die Durchführung von Tests mit Produktivdaten nicht mehr zwingend eine separate, auf dem aktuellsten Datenstand gehaltene Testumgebung bereitgestellt werden muss.

Den größten Nutzen bietet die Editionierung der Datenbankobjekte im Bereich des Online Application Upgrades. Bei der Aktualisierung der Anwendung können die Benutzer weiterhin in ihrer bisherigen Edition weiterarbeiten, bis die aktualisierten Objekte in der neuen Edition fertig installiert und freigegeben sind. Anschließend wird die neue Edition als Default-Edition gesetzt, sodass die Nutzer ab der nächsten Anmeldung standardmäßig

mit den Objekten dieser Edition arbeiten. Alternativ kann auch innerhalb einer aktiven Session die zu verwendende Edition manuell gesetzt werden. Edition Based Redefinition unterstützt folgende Objekttypen sowohl in der Datenbank-Version 11*g* R2 als auch in der Version 12*c* R1:

- Views
- Functions/Procedures/Packages
- Trigger
- Types
- Libraries
- Private Synonyme

In der Version 12*c* ist die Edition-Based-Redefinition-Funktionalität stark erweitert.

Grenzen der Edition Based Redefinition

Leider wird der durch Edition Based Redefinition (EBR) gebotene Komfort durch einige Einschränkungen reduziert:

- *Public Synonyms*
  Im Gegensatz zu privaten Synonymen ist eine Editionierung von Public Synonyms nicht möglich, da nicht jede Edition in jedem Schema verwendet werden muss. Bei einem Public Synonym handelt es sich allerdings um ein Objekt, das Schema-übergreifend existiert, wodurch für die Datenbank nicht feststellbar ist, welcher Edition es zugeordnet werden soll. Innerhalb eines editionierten Schemas sind Public Synonyms allerdings möglich, solange diese nicht auf ein editioniertes Objekt verweisen.
- *Benutzerdefinierte Datentypen*
  Die Verwendung benutzerdefinierter Datentypen in Tabellen, durch die ein Verweis auf ein editioniertes Objekt erfolgt, ist ebenfalls nicht möglich, ebenso darf ein nicht-editioniertes Subprogram keine statische Referenz auf ein Subprogram haben, dessen Eigentümer-Schema editioniert ist.
- *Function Based Indizes*
  Eine Editionierung von Function Based Indizes, die auf einem editionierten Objekt basieren, ist nicht möglich.
- *Materialized Views (11g R2)*
  In der Version 11*g* R2 können keine Materialized Views erstellt werden, die auf editionierten Objekten basieren, da für die Materialized View nicht ersichtlich ist, welche Edition des referenzierten Objekts verwendet werden soll. In Version 12*c* entfällt diese Einschränkung, da hier nun angegeben werden kann, welche Edition des referenzierten Objekts verwendet werden soll.
- *Virtual Column (11g R2)*
  Die Verwendung von virtuellen Spalten, die auf editionierte Objekte verweisen, ist in der Version 11*g* R2 aufgrund derselben Problematik wie bei Materialized Views nicht möglich. Mit der Version 12*c* wurde hier ebenfalls eine Möglichkeit eingeführt, die zu verwendende Edition explizit anzugeben.
- *Statische Referenz auf editioniertes Subprogram*
  Eine statische Referenz auf ein editioniertes Subprogram durch ein nicht-editioniertes Subprogram ist ebenfalls nicht möglich, da nicht festgelegt werden kann, auf welche Edition die Referenz erfolgen soll.
- *Tabellen und darin enthaltene Daten*
  Ein großes Manko beim Einsatz von Edition Based Redefinition ist, dass die Editionierung von Tabellen nicht ohne Weiteres beziehungsweise ohne manuellen Eingriff möglich ist. Zur Versionierung von Tabellen und der darin enthaltenen Daten existieren zwei unterschiedliche Konzepte, die im Abschnitt „Editionierung von Tabellen" näher beschrieben sind.

Die Aktivierung

Zunächst wird überprüft, ob EBR für das betreffende Schema bereits aktiviert wurde *(siehe Listing 1)*. Für die Erzeugung einer neuen Edition ist das „CREATE ANY EDITION"-Recht erforderlich, für das Löschen bestehender Editionen das „DROP ANY EDITION"-Recht. Diese Rechte werden dem Schema durch „GRANT CREATE ANY EDITION, DROP ANY EDITION TO meinschema;" zugewiesen.

```
SELECT editions_enabled
  FROM dba_users
 WHERE username = 'MEINSCHEMA';
```

Listing 1

```
SELECT u.name Schema,
       o1.name Objektname
  FROM obj$ o1,
       obj$ o2,
       dependency$ dep,
       user$ u
 WHERE o1.obj# = dep.d_obj#
   AND o2.obj# = dep.p_obj#
   AND o1.remoteowner is null
   AND o2.owner# = (   SELECT user_id
                        FROM sys.dba_users
                       WHERE username = 'MEINSCHEMA'
                   )
   AND o1.owner# = u.user#
   AND o2.type# in (4,5,7,8,9,10,11,12,13,14,22,87)
   AND (
          (      u.type# <> 2
             AND bitand(u.spare1, 16) = 0
             AND u.user# <> o2.owner#
          )
        OR
          (
             O1.type# NOT IN (4,5,7,8,9,10,11,12,13,14,22,87)
          )
       )
;
```

Listing 2

Bevor nun für das jeweilige Schema die Editionierung aktiviert wird, sollte zunächst überprüft werden, ob in diesem Schema Objekte vorhanden sind, die selbst nicht editionierbar sind, aber gleichzeitig von editionierbaren Objekten abhängen, wodurch eine Aktivierung der Editionierung zu Problemen führen kann. Mithilfe des als SYS-Benutzer auszuführenden SELECT-Statements kann (für die Version 11*g* R2) überprüft werden, ob solche Objekte im Schema vorhanden sind *(siehe Listing 2)*.

Sollte dieser „SELECT" keine Daten zurückliefern, stellt die Aktivierung von EBR kein Problem dar. In der Version 11*g* sind oftmals Materialized Views, die auf editionierbare Views zugreifen, ein Problem, da Materialized Views in dieser Datenbankversion nicht editioniert werden können. In der Version 12*c* muss hierbei lediglich angegeben werden, welche Edition der zugrunde liegenden Objekte verwendet werden soll. Anschließend wird EBR mittels „ALTER USER meinschema ENABLE EDITIONS;" für das angegebene Schema aktiviert.

Erzeugen und Löschen von Editionen

Nach der Aktivierung der Editionierung ist standardmäßig die sogenannte Root-Edition „ora$base" vorhanden. Auf dieser basieren alle nachfolgend angelegten Editionen. Sie kann daher nicht gelöscht werden. Sobald eine neue Edition erstellt wird, wird diese durch die Datenbank als Child-Edition unterhalb der Root-Edition angelegt.

Eine neue Edition wird immer als Schema-unabhängiges Objekt angelegt, zur Erstellung dient der SQL-Befehl „CREATE EDITION edition1 [AS CHILD OF ora$base];". Eine neue Edition erbt zum Zeitpunkt ihrer Erstellung immer die editionierten Objekte ihrer Parent-Edition – diese werden dazu in die neue Edition hineinkopiert.

Zum Löschen einer Edition – mitsamt ihrer editionierten Objekte – genügt ein simpler DROP-Befehl: „DROP EDITION edition1;". Objekte, die nicht editioniert sind, werden mit diesem Befehl nicht gelöscht.

Wechsel zwischen den Editionen

Bei einem Wechsel der Edition innerhalb der Session wird zunächst die aktuell in der Session aktivierte Edition durch die Funktion „SYS_CONTEXT" ermittelt: „SELECT sys_context('userenv', 'session_edition_name') FROM dual;". Analog dazu ist die aktuell gültige bzw. neueste Edition festzustellen, dies geschieht über den Systemkontext „current_edition_name".

Die Auswahl der zu nutzenden Edition erfolgt auf Session-Ebene und kann dementsprechend komfortabel per „ALTER SESSION"-Statement stattfinden: „ALTER SESSION SET EDITION = edition1;". Das Recht zum Zugriff auf die einzelnen Editionen muss dem Nutzer allerdings im Voraus mittels „GRANT USE ON EDITION edition1 TO meinschema;" zugewiesen werden. Soll die Verwendung einer Edition für alle Nutzer möglich sein, wird durch das Grant-Statement dieses Recht einfach dem Schema PUBLIC zugeordnet. Die standardmäßig zu verwendende Edition wird durch „ALTER DATABASE DEFAULT

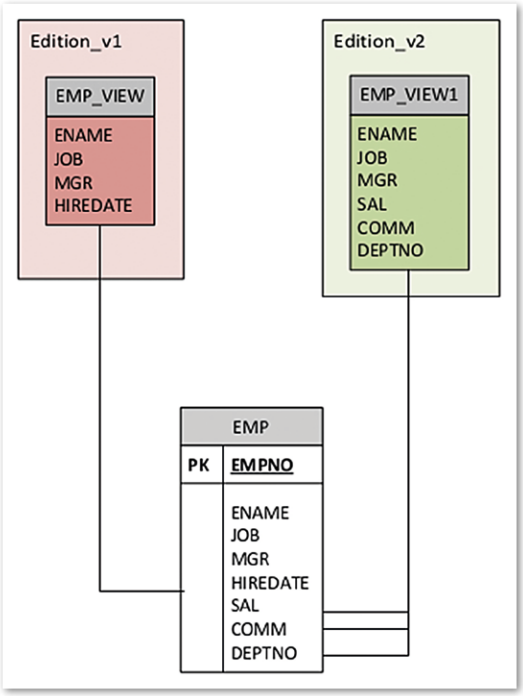

Abbildung 1: Tabellen-Redefinition mittels „Editioning Views"

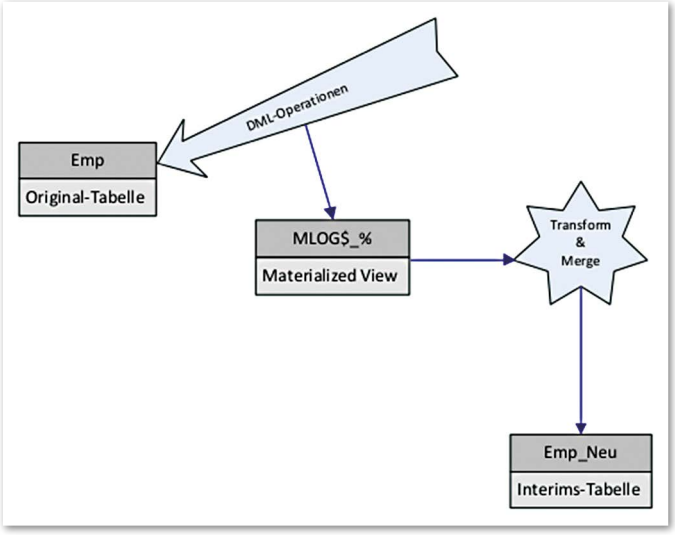

Abbildung 2: Tabellen-Migration mit „DBMS_REDEFINITION"

EDITION = edition1" datenbankweit festgelegt.

Das Erzeugen, Bearbeiten und Löschen von Datenbank-Objekten erfolgt wie gewohnt, es ist lediglich zu beachten, dass in der Datenbanksession die jeweils gewünschte Edition aktiviert ist – es werden ausschließlich die in der für die jeweilige Session aktiven Edition verfügbaren Objekte bearbeitet, erzeugt oder gelöscht.

Die Editionierung erfolgt vollständig transparent, sodass zum Zeitpunkt der Installation der Datenbankobjekte (bis auf die Auswahl der Edition innerhalb der Session) keinerlei Besonderheiten beachtet werden müssen.

Editionierung von Tabellen

Mittels EBR lassen sich Tabellen leider nicht in Editionen verwalten, da eine automatisierte Veränderung und Anpassung der Daten konzeptionell nicht gewünscht ist. Zur Lösung dieser Problematik stehen zwei verschiedene Lösungswege zur Verfügung. Mithilfe von „Editioning Views" kann eine Redefinition der Tabellen in allen Editionen der Oracle-Datenbank durchgeführt werden, der Aufwand hierfür ist unter Umständen aber etwas höher als bei der Synchronisierung mittels „DBMS_REDEFINITION", die leider nur in der Enterprise Edition zur Verfügung steht.

Bei der Tabellen-Redefinition per „Editioning Views" wird der Zugriff auf die Tabellen mit einem View-Layer abstrahiert, durch den die jeweilige „Edition" der Tabelle abgebildet wird *(siehe Abbildung 1)*. Alle Editionen der View greifen auf dieselbe Tabelle zu, stellen der Anwendung aber nur die für die jeweilige Edition benötigten Daten zur Verfügung. Wird nun der Tabelle eine neue Spalte hinzugefügt, so wird diese an die zugrunde liegende Tabelle angehängt und in der View der Edition, mit der die Spalte zur Verfügung stehen soll, ebenfalls hinzugefügt. Beim Entfernen von Spalten werden diese nicht aus der Tabelle entfernt, sondern lediglich in der View der entsprechenden Edition nicht mehr mit ausgegeben.

Durch diese Lösung lassen sich Änderungen am Datenmodell relativ leicht abbilden, ohne die zugrunde liegenden Daten ändern zu müssen, allerdings lässt sich nicht jeder Fall einwandfrei abbilden.

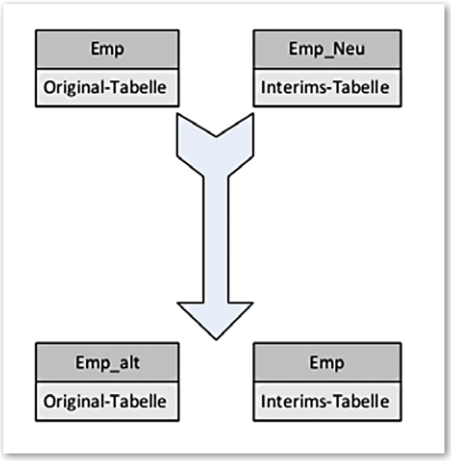

Abbildung 3: Ersetzen der Ursprungstabelle durch die neue Tabellenversion

Probleme können zum Beispiel beim Ändern von Datentypen auftreten, außerdem können die zugrunde liegenden Tabellen schnell sehr viele nicht mehr benötigte Spalten enthalten.

Bei der Tabellen-Redefinition mit „DBMS_REDEFINITION" werden nicht mehrere Editionen parallel vorgehalten, sondern nur die Migration von Tabellenobjekten in eine neue Version unterstützt. Vereinfacht formuliert handelt es sich hierbei um die Erzeugung eines Klons der zu verändernden Tabellen, der nun bearbeitet werden kann, bei Abschluss der Redefinition die Daten der ursprünglichen Tabelle enthält und diese Tabelle ersetzt *(siehe Abbildung 2)*.

Vor der Tabellen-Redefinition muss zunächst überprüft werden, ob eine Redefinition grundsätzlich möglich ist. Hierzu stellt Oracle eine Überprüfungsfunktion zur Verfügung: „dbms_redefinition.can_redef_table('meinschema', 'meine_original_tabelle')". Anschließend wird eine Tabelle mit der neuen Struktur erzeugt, die später die ursprüngliche Tabelle ersetzen wird. Hierfür wird ein normales „CREATE TABLE"-Statement ohne jegliche Besonderheiten verwendet. Es ist nur zu beachten, dass genügend Speicherplatz zur Erzeugung einer Kopie der Ursprungstabelle zur Verfügung steht.

Mittels „dbms_redefinition" werden nun die Redefinition gestartet und die Inhalte der Ursprungstabelle in die neue Tabelle kopiert: „dbms_redefinition.start_redef_table('meinschema', 'meine_original_tabelle', 'meine_neue_tabelle')". Seit der Datenbank-Version 10*g* werden die den Tabellen zugehörigen Objekte wie Constraints, Trigger sowie Indizes automatisch mithilfe der Prozedur „COPY_TABLE_DEPENDENTS" mitkopiert und bei Bedarf auch kompiliert bzw. aktiviert.

Aus Performance-Gründen sollte vor dem Abschluss der Redefinition noch eine Synchronisation der Tabellen durch die Prozedur „dbms_redefinition.SYNC_INTERIM_TABLE" erfolgen. Mittels „FINISH_REDEF_TABLE" wird anschließend die Redefinition abgeschlossen und die neue Tabelle ersetzt die Ursprungstabelle *(siehe Abbildung 3)*.

Fazit

Mit Edition Based Redefinition stellt Oracle eine komfortable Möglichkeit zur transparenten Versionierung von Datenbankobjekten zur Verfügung. Allerdings lässt sich diese Technik nicht durchgängig für alle Objekttypen anwenden, wodurch sich der Einsatzbereich im Alltag in der Regel auf die Editionierung von Stored Procedures und Views beschränken wird. Mit einigen Abstrichen ist auch die Migration von Tabellen auf eine neue DDL-Version möglich – allerdings kann hierbei nicht ohne Weiteres innerhalb der Session zwischen verschiedenen Versionen gewechselt werden.

Insbesondere für den Test neuer Versionen von Stored Procedures und deren Auswirkungen auf andere Datenbankobjekte sowie das Online Application Upgrade von Stored Procedures stellt die Verwendung von Edition Based Redefinition ein praktisches Feature dar, das sich insbesondere in der Datenbank-Version 12*c* für die regelmäßige Verwendung eignet.

Daniel Horwedel
daniel.horwedel@merlin-zwo.de