Edition Based Redefinition: Versionsverwaltung für Datenbankobjekte

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

Bei der Aktualisierung von Anwendungen ergibt sich für die Entwickler häufig die Anforderung, dass diese ohne große Wartungsfenster erfolgen sollen.
Des Weiteren 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). Hiermit lassen sich mehrere Versionen eines Datenbankobjektes im Schema installieren und sessionabhängig verwenden, wodurch ein Update auf eine neue Version Ihrer Datenbankobjekte oder ein Test der neuen Version ohne Beeinträchtigung des laufenden Betriebs erfolgen kann.
Die Anwender haben in ihrer Session die alte Version der Datenbankobjekte zur Verfügung, während die neue Version installiert und getestet werden kann. Mit einem einfachen „ALTER SESSION“-Befehl kann anschließend die neue Version den Anwendern zur Verfügung gestellt 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, so dass bislang aufwendige Lösungen benötigt wurden, um ein Anwendungs-Upgrade ohne Unterbrechung des laufenden Betriebes durchführen zu können.

Des Weiteren 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, mittels derer auf Basis von Produktivdaten der

Impact einer Änderung der Datenstruktur ermittelt werden kann. Hierbei entsteht allerdings durch das Vorhalten mehrerer Testumgebungen ein gewaltiger Aufwand, wodurch im Endeffekt oftmals veraltete Testdaten zur Verfügung stehen oder die Testschemas 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.
Das Grundprinzip des Online Application Upgrade 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 die neue Version freigegeben wird, werden die Benutzer quasi „auf Knopfdruck“ auf die neue Anwendungsversion umgestellt.

Die Oracle-Datenbank kann das: Edition Based Redefinition

Oracle bietet zur Durchführung dieser Online Application Upgrades mittels Edition Based Redefinition seit der Version 11g Release 2 ein umfangreiches Werkzeug an, welches mit dem Erscheinen von Version 12c erheblich erweitert und verbessert wurde.
Die Edition Based Redefinition lässt sich mit allen Editionen der Datenbank kostenfrei benutzen, so dass diese Funktionalität nicht nur den Anwendern der Enterprise Edition vorbehalten ist, sondern auch schon ab der Express Edition einsetzbar ist.

Versionierung der DB-Objekte als Lösungsweg

Zur Reduzierung des Aufwandes 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.
Hierbei werden mehrere Versionen eines Datenbankobjektes 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.
Hierdurch wird der Aufwand für die Pflege und Bereitstellung von Testumgebungen deutlich reduziert, da für die Durchführung von Tests mit Produktivdaten nicht mehr zwingend eine separate, auf dem aktuellsten Datenstand gehaltene, Testumgebung bereitgehalten werden muss.
Den größten Nutzen bietet die Editionierung der Datenbankobjekte aber im Bereich des Online Application Upgrades. Bei Aktualisierungen 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, so dass 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.

Welche Objekte werden durch EBR unterstützt?
Durch Edition Based Redefinition werden die folgenden Objekttypen sowohl in der Datenbank-Version 11g Release 2 als auch in der Version 12c Release 1 unterstützt:

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

In der Version 12c wurde die Edition Based Redefinition-Funktionalität stark erweitert, so dass in dieser Version die folgenden Objekt-Typen ebenfalls unterstützt werden:

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, welches Schemaübergreifend existiert, wodurch für die Datenbank nicht feststellbar ist, zu welcher Edition es zugeordnet werden soll.
Innerhalb eines editionierten Schemas dürfen Public Synonyme allerdings verwendet werden, 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 11g Release 2 können keine Materialized Views erstellt werden, die auf editionierten Objekten basieren, da für den Materialized View nicht ersichtlich ist, welche Edition des referenzierten Objektes verwendet werden soll.
  In Version 12c entfällt diese Einschränkung, da hier nun angegeben werden kann, welche Edition des referenzierten Objektes verwendet werden soll.
- Virtual Column (11g R2)
  Die Verwendung von virtuellen Spalten, die auf editionierte Objekte verweisen, ist in der Version 11g R2 aufgrund derselben Problematik wie bei Materialized Views nicht möglich.
  Mit der Version 12c hat Oracle 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 bzw. ohne manuellen Eingriff möglich ist. Zur Versionierung von Tabellen und den darin enthaltenen Daten existieren zwei unterschiedliche Konzepte, die im Abschnitt „Editionierung von Tabellen“ näher beschrieben werden.

Aktivierung von EBR

Zunächst wird mit dem folgenden Statement überprüft, ob EBR für das betreffende Schema bereits aktiviert wurde:

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

Für die Erzeugung einer neuen Edition wird das CREATE ANY EDITION-Recht benötigt, für das Löschen bestehender Editionen ist das DROP ANY EDITION-Recht erforderlich. Diese Rechte werden dem Schema mittels folgendem Statement zugewiesen:

```
GRANT CREATE ANY EDITION, DROP ANY EDITION TO meinschema;
```

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.

Mittels folgendem, als SYS-Benutzer auszuführendem, SELECT-Statement kann (für die DB-Version 11g R2) überprüft werden, ob solche Objekte im Schema vorhanden sind:

```
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)
            )
        )
;
```

Sollte dieser SELECT keine Daten zurückliefern, stellt die Aktivierung von EBR kein Problem dar. In der Version 11g stellen oftmals Materialized Views, die auf editionierbare Views zugreifen, ein Problem dar, da Materialized Views in dieser Datenbankversion nicht editioniert werden können. In der Version 12c muss hierbei lediglich angegeben werden, welche Edition der zugrundeliegenden Objekte verwendet werden soll.
Anschließend wird EBR mittels einem ALTER USER-Statement für das angegebene Schema aktiviert:

```
ALTER USER meinschema ENABLE EDITIONS;
```

Erzeugen und Löschen von Editionen

Nach der Aktivierung der Editionierung ist standardmäßig die so genannte Root-Edition „ora$base“ vorhanden.
Auf dieser Edition basieren alle im Folgenden angelegten Editions. Diese Edition kann daher nicht gelöscht werden. Sobald eine neue Edition erstellt wird, wird diese durch die Datenbank als Child-Editions unterhalb der Root-Edition angelegt.

Eine neue Edition wird immer als schemaunabhängiges Objekt angelegt, zur Erstellung dient der folgende 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 mittels dieses Befehls nicht gelöscht.

Wechsel zwischen den Editionen

Bei einem Wechsel der Edition innerhalb der Session wird zunächst die aktuell in der Session aktivierte Edition mittels der Funktion SYS_CONTEXT ermittelt:

```
SELECT sys_context('userenv', 'session_edition_name') FROM dual;
```

Analog dazu ist die aktuell gültige bzw. neueste Edition zu ermitteln, 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 erfolgen:

```
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 ermöglicht werden, so wird durch das Grant-Statement dieses Recht einfach dem Schema PUBLIC zugewiesen.
Die standardmäßig zu verwendende Edition wird mittels

```
ALTER DATABASE DEFAULT EDITION = edition1
```

datenbankweit festgelegt.

Erzeugen, Bearbeiten und Löschen von DB-Objekten

Das Erzeugen, Bearbeiten und Löschen von Datenbank-Objekten erfolgt wie gewohnt, hierbei 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, so dass zum Zeitpunkt der Installation der Datenbankobjekte – bis auf die Auswahl der Edition innerhalb der Session – keinerlei Besonderheiten beachtet werden müssen.

Editionierung von Tabellen

Mittels Edition Based Redefinition 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. Mittels 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.
Im Folgenden werden diese beiden Varianten näher beschrieben sowie deren Vor- und Nachteile erläutert.

Tabellen-Redefinition mittels Editioning Views

Bei der Tabellen-Redefinition mittels Editioning Views wird der Zugriff auf die Tabellen mittels eines View-Layers 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 eine neue Spalte zur Tabelle hinzugefügt, so wird diese Spalte an die zugrundeliegende 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 zugrundeliegenden Daten ändern zu müssen, allerdings lässt sich nicht jeder Fall einwandfrei abbilden. Probleme können zum Beispiel beim Ändern von Datentypen auftreten, außerdem können die zugrundeliegenden Tabellen schnell sehr viele nicht mehr benötigte Spalten enthalten.

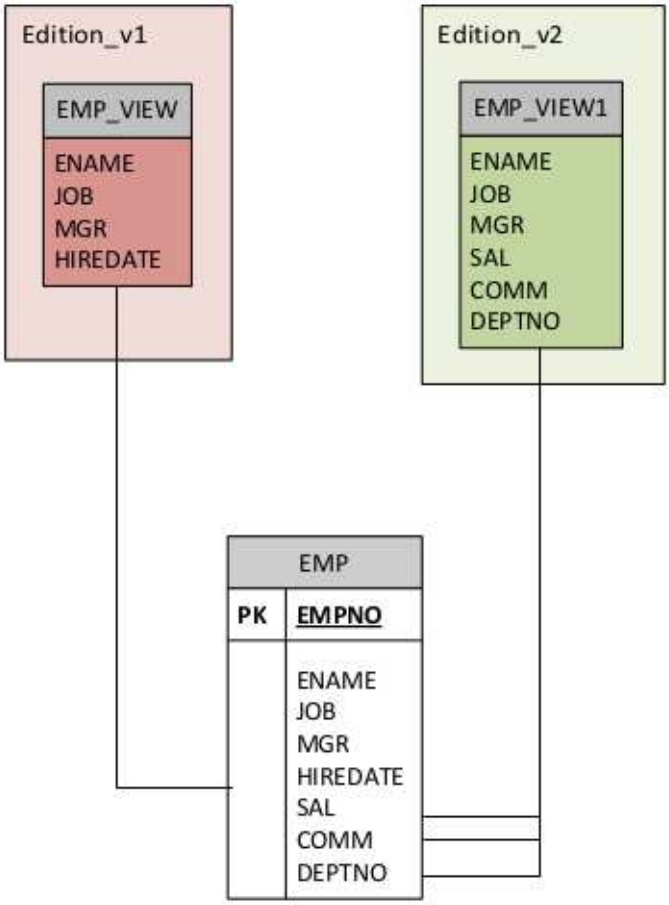

Abbildung 1: Tabellen-Redefinition mittels Editioning Views

Tabellen-Redefinition mittels DBMS_REDEFINITION

Hierbei werden nicht mehrere Editionen parallel vorgehalten, sondern lediglich 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 und bei Abschluss der Redefinition die Daten der ursprünglichen Tabelle enthält und diese Tabelle ersetzt.

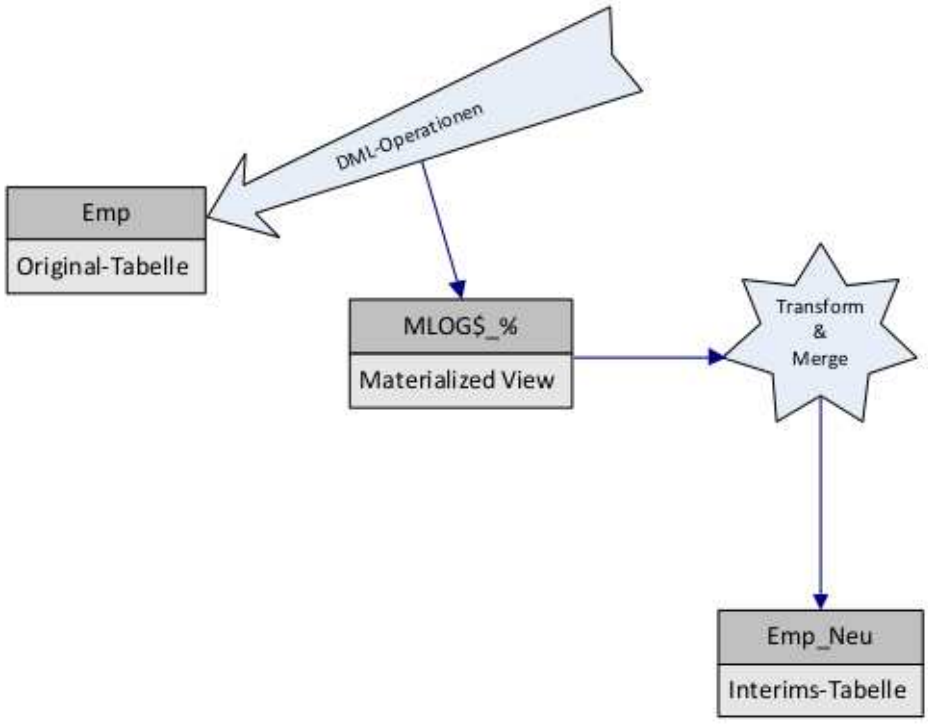

Abbildung 2: Tabellen-Migration mit DBMS_REDEFINITION

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 lediglich zu beachten, dass genügend Speicherplatz zur Erzeugung einer Kopie der Ursprungstabelle zur Verfügung steht.

Mittels dbms_redefinition wird 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 10g 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 mittels der 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).

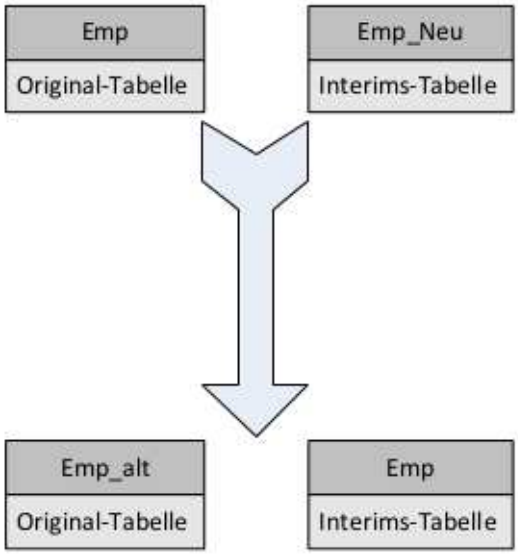

Abbildung 3: Ersetzen der Ursprungstabelle durch die neue Tabellenversion

Fazit

Mittels der Funktionalität “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 Einschränkungen 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 12c für die regelmäßige Verwendung eignet.

Kontakt:

Daniel Horwedel

merlin.zwo InfoDesign GmbH & Co. KG

daniel.horwedel@merlin-zwo.de