

Edition-based Redefinition (EBR) Frequently Asked Questions

May, 2024, Version 4.0
Copyright © 2024, Oracle and/or its affiliates
Public

Purpose Statement

This document provides guidance regarding common questions and answers regarding using the Edition-based Redefinition (EBR) feature within the Oracle Database. EBR enables seamless near-zero or zero downtime application upgrades as part of an application lifecycle by allowing database objects to be updated in a new edition while maintaining the current edition. Please find more details regarding EBR, specifically in the Edition-based Redefinition Technical Deep Dive, [here](#).

Disclaimer

This document, in any form, software, or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without the prior written consent of Oracle. This document is not part of your license agreement, nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remain at the sole discretion of Oracle.

Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Contents

Purpose Statement	2
Disclaimer	2
Frequently Asked Questions (FAQ)	4
What is actualization?	4
Does the Oracle Database have a maximum limit regarding the number of editions that can be created?	4
What happens to inherited objects when I drop an older edition?	4
Should grants cause invalidations?	5
What does drop edition cascade do in 12.2?	5
Can the timeframe for when dropped editions are cleaned be controlled to fit a specific schedule?	6
How should an application that utilizes Materialized Views & Virtual Columns that reference objects that would be editionable utilize EBR?	6
How many editions should a user maintain?	7
Should a user drop ORA\$BASE?	7
What is the best practice for dropping cross-edition triggers?	7
Are database links editionable?	7
Where can I get more information about EBR?	7

Frequently Asked Questions (FAQ)

What is actualization?

To save time and space when creating a new edition, all of the editioned objects in the new edition will be inherited from the parent edition. In other words, the new edition will use the actual definition of the object from the prior edition rather than creating a new copy. When objects are redefined as their definitions are updated, a new copy is created in the new edition so that the copy in the prior edition is not disturbed. The process of creating a new copy of an object in a new edition is called "actualization."

Actualization is most often used to update the definition of an object when a patch is installed that needs to change the object. Occasionally, a user might want to sever the inheritance of the old copy and create an independent copy of an object in the latest edition, even without altering the definition. This creation of a new copy with the exact definition is also called "actualization."

Does the Oracle Database have a maximum limit regarding the number of editions that can be created?

Yes, but it is very high at 32,767 which should satisfy the vast majority of requirements regarding the number of editions. It should be noted that the maximum limit was 2,000 in Oracle Database 11g R2 but was increased to the current limit as of Oracle Database 12.1.

What happens to inherited objects when I drop an older edition?

An inherited object is a database object that has not been actualized in the current edition. It is only a stub.

- How can one tell that an object is inherited? What changes to the base object would be observable in the later edition?

Looking at the ALL_OBJECTS table, an inherited object will have an EDITION_NAME value different from the current edition. Propagation of changes to the base object, such as dropping it, should not be relied upon. Changes to any object should be repeated in all editions where the change is expected to be observable.

- Can one cause an object to become actualized? If so, is there an observable difference between inherited and actualized objects?

DDL statements (e. g. CREATE, ALTER, DROP) will actualize inherited objects, even if the statement doesn't functionally change the object.

Example: an editioned package can be recompiled with the REUSE SETTINGS clause; this will actualize the package in the current edition without changing it respective to its parent. Dependent objects on this package will be lazily actualized as they're needed. We can force dependent packages to be actualized via `dbms_utility.validate()`.

An actualized object can be differentiated from an inherited object by comparing the values of the EDITION_NAME column in ALL_OBJECTS. If the value matches the name of the current edition, it is actual.

- Inheritance is at the heart of the shine-through model. What is the shine-through model?

The shine-through model refers to how changes to an object in a given edition might be reflected in dependent objects that belong to descendent editions. This behavior should not be relied upon. Any changes that are expected to be seen in descendent editions should be repeated on a per-edition basis.

- What is a covered object?

A covered object is an editioned object no longer inherited in any usable descendent edition. An automated scheduled maintenance process drops each covered object in any unusable edition.

Should grants cause invalidations?

Technically, a DDL can only be performed on an actual object (as opposed to an inherited object). For this reason, any time a DDL is executed on an object inherited in the current edition of the session executing the DDL, the object must first be actualized before the DDL can proceed. This step does apply to grants.

One immediate effect of actualizing an object is that all of its dependents recursively must be invalidated so that all dependents can be recompiled to refer to the new copy of the object rather than the old copy. If this behavior is undesirable, there are two ways around it.

One way is to plan ahead for grants during a patch installation. This way has two parts. As much as possible, grants should be executed during the installation of a patch before it becomes available for execution by user sessions. While it is impossible to anticipate precisely which users or roles would receive the grants, it is possible to predict which objects the grants would be executed on; those objects can be preemptively actualized, even if the actual grant is not executed until later. This would ensure that all actualizations occur before users begin using the new edition so users are not disturbed by invalidations.

The other method for dealing with grants is to explicitly seek out the actual object that is the target of the grant. The (User/All/DBA)_Objects view has a column that specifies the edition in which every editioned object is defined. Using either DBMS_SQL or alter session, the grant can be executed in the edition where the target of the grant is defined. This way, executing a grant on an actual object rather than on an inherited copy will avoid any need for actualization and avoid invalidations.

The first method of preemptively actualizing objects should be preferred whenever possible rather than executing grants in old editions. Generally, it should always be considered best practice to execute DDLs in the newest edition, not in any edition with a child.

What does drop edition cascade do in 12.2?

Drop edition cascade of the root edition will determine if the edition itself is not empty. It will mark the edition unusable and then clean it in the background once the edition is empty.

Before 12.2, a drop edition cascade would error out if the edition contained objects inherited by later editions.

For more control over when the cleanup happens, calling DBMS_Editions_Utilities.Clean_Unusable_Editions will immediately clean up all objects not currently inherited in a usable edition.

Can the timeframe for when dropped editions are cleaned be controlled to fit a specific schedule?

It's best practice to let the periodically scheduled cleanup process handle timeframes.

- What is the relationship between the cleanup and drop edition?

Dropping an edition that is the newest edition will cause all editioned objects, along with the edition itself, to be dropped before the drop edition statement returns.

In the case of the oldest edition, the edition will be marked as unusable, and the drop edition statement will return immediately. However, a cleanup process will periodically clean all covered objects. When all objects in the edition are cleaned, the edition itself will be dropped.

- What exactly gets cleaned?

The periodically scheduled cleanup process will clean covered objects in unusable editions. Unusable editions are dropped automatically once all covered objects in them are cleaned.

- If cleanup and drop edition are coupled, does it make sense to do the drop edition at an opportune time instead?

It's unclear if this approach would yield any benefits, and the scheduling of 'drop edition' statements could add an unnecessary layer to code. It's best to let the cleanup process handle dropped editions unless it results in concrete problems.

How should an application that utilizes Materialized Views & Virtual Columns that reference objects that would be editionable utilize EBR?

Using the syntax "evaluate using current edition" will allow the materialized view or virtual column to reference editioned objects in the same edition that the session that creates the materialized view or virtual column is currently using (the one returned by `sys_context('USERENV','CURRENT_EDITION_NAME')`). The materialized view or virtual column will continue to use this specific edition. It will always use this particular edition, even if the materialized view or virtual column is queried (or refreshed in the case of a materialized view) from a session using a different edition.

To avoid unexpected behavior due to stale evaluate relationships, materialized view or virtual column evaluate clauses should be rolled forward to the latest edition whenever possible via `ALTER MATERIALIZED VIEW` or `ALTER TABLE` commands.

Typically, if a particular patch or upgrade to the application is being installed that will change the results of the defining expression in some way, a new materialized view would be created to store the different results without disturbing the existing materialized view used by the running application. In anticipation of the occasional need to create replacement materialized views, in general, materialized views should be covered by editioning views, traditional views, or synonyms so that any application code that refers to the materialized view by name would reference an editioned object that can be updated to point to a new materialized view when a replacement is created.

How many editions should a user maintain?

Users can use the drop edition feature to drop old editions once they are no longer used. This way, the database would have the one edition currently being used by the running application and perhaps one additional edition if a patch or update is being rolled out. All other editions in the database would have been marked unusable by a prior "drop edition" command.

Should a user drop ORA\$BASE?

It is necessary to drop the ORA\$Base edition before other old, retired editions can be dropped. The ORA\$Base edition is not unique in any way other than the name given to the one edition in the database when the database is first created. All our existing customers that have happily been using edition-based redefinition for many years all dropped the ORA\$Base edition many years ago and have had no problems.

What is the best practice for dropping cross-edition triggers?

A cross-edition trigger is a trigger that has cross-edition visibility. Its primary purpose is keeping data synchronized across the editions.

As editions are dropped, it makes sense that cross-edition triggers should also be dropped.

- Can a cross-edition trigger have visibility over more than two editions?

No, a cross-edition trigger can only "link" two editions and only belongs to the edition in which it is actual.

- If a cross-edition trigger deals with editions that have been dropped, is the trigger also dropped automatically? What about a reverse-cross edition trigger? Is it dropped as soon as the older edition is dropped?

Forward triggers are dropped automatically.

Reverse triggers should be disabled first as part of a teardown process before dropping the edition.

Are database links editionable?

Currently (21.3) no.

Where can I get more information about EBR?

<https://blogs.oracle.com/maa/>

<https://www.oracle.com/corporate/events/maa-webcast-series.html>

Connect with us

Call **+1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

 blogs.oracle.com

 facebook.com/oracle

 twitter.com/oracle

Copyright © 2024, Oracle and/or its affiliates. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.