Edition-Based Redefinition in 12c

COLLABORATE15

TECHNOLOGY AND APPLICATIONS FC
FOR THE ORACLE COMMUNITY

Presented by: John Jay King

Download this paper from:
http://www.kingtraining.com

COLLABORATE15

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Session Objectives

- Understand the implications of Edition Based Redefinition (EBR)
- Know the SQL necessary to implement EBR
- Become prepared to begin planning for EBR
- Learn how EBR may be used to make improve availability for your customers

Who Am I?

- John King – Partner, King Training Resources
- Oracle Ace Director 
- Member Oak Table Network 
- Providing training to Oracle and IT community for over 25 years – http://www.kingtraining.com
- "Techie" who knows Oracle, SQL, Java, ADF, and PL/SQL pretty well (along with other topics)
- Leader in Service Oriented Architecture (SOA)
- Member of AZORA, RMOUG, ODTUG, and IOUG
- Home is Scottsdale, Arizona

King Training Resources

Agenda

I. Introduction to EBR
II. EBR Administration
III. Editions
IV. Editioning Views
V. Cross-Edition Triggers
VI. Demo

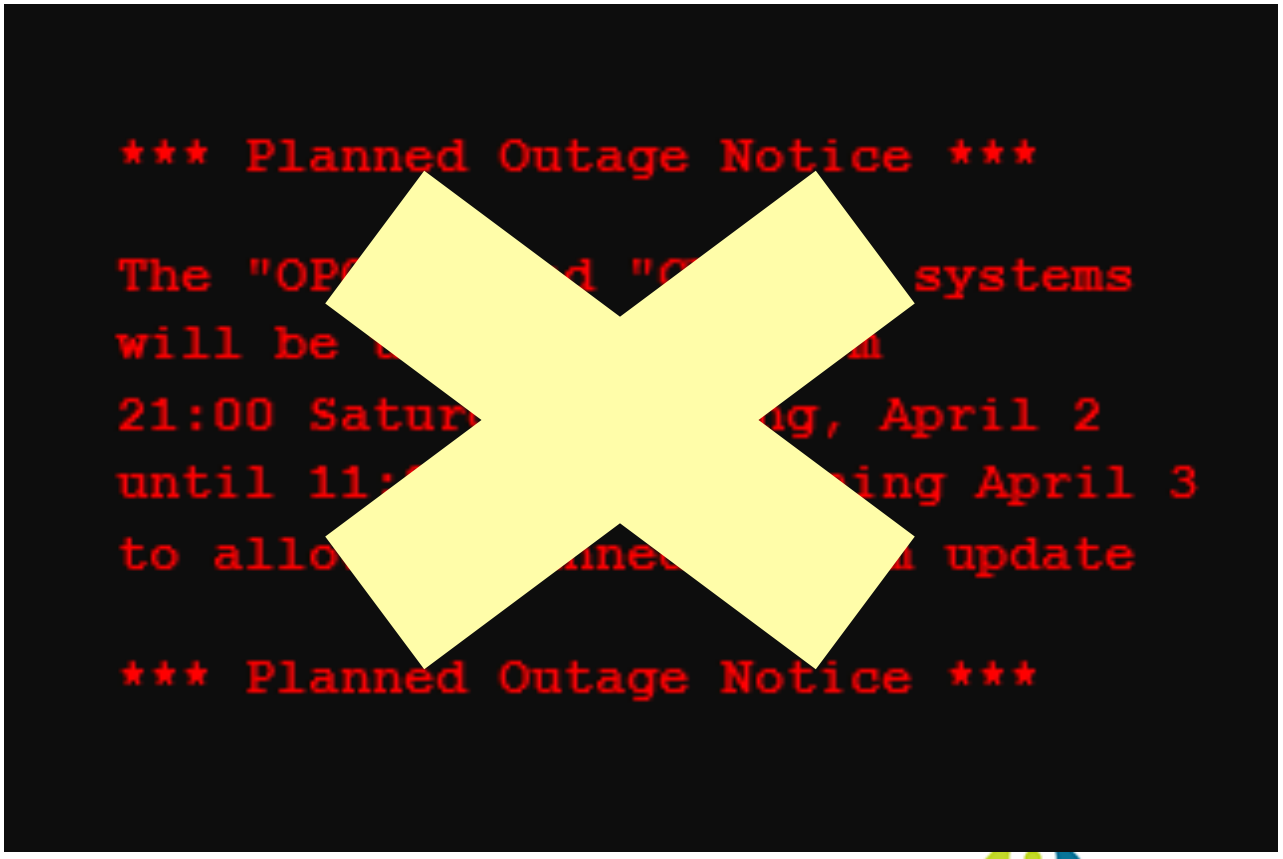
Planned Outages

"Can't live with them

- Can't live without them"

But…

- What if you could drastically reduce the downtime outages require?



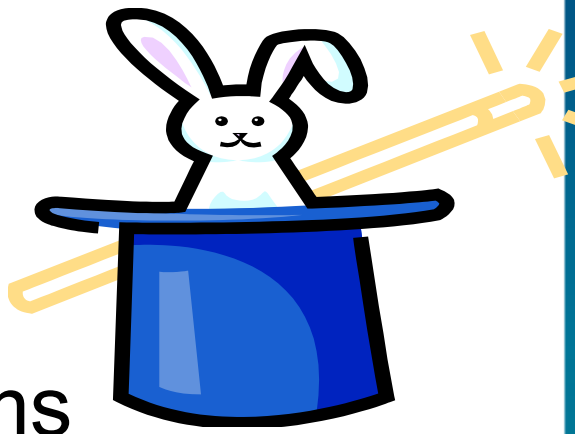
Online Application Upgrade

- The quest to eliminate downtime has led to a desire to provide "Online Application Upgrade"
  - An application need not be taken down when upgrades are applied
  - Users of the existing system continue uninterrupted
  - Users of the upgraded system use new code immediately

EBR to the Rescue!

- Edition-Based Redefinition (EBR) was described as the “killer feature” of 11gR2
- Enhanced in 12c
- EBR provides a revolutionary ability to manage change of stored PL/SQL
- Applications need not be taken down when upgrades/changes are applied greatly reducing downtime required to upgrade

Is EBR Magic?



- It just seems like it!
- Edition-Based Redefinition uses a non-schema "edition" of an application; including PL/SQL, views, and synonyms
  - A new edition may be created without impacting users of the current edition
  - New editions may be modified as desired then tested and deployed; again without impacting users of the original edition
  - When a new edition is ready for complete rollout it may be released to all users

What is Editionable?

- EBR allows the ability to manage versions (editions) of “editionable objects”
- Editionable object types:
  - PL/SQL objects of all kinds
  - Synonyms
  - Views
  - SQL translation profile (new in Oracle 12c)
- Beginning with Oracle 12c; Materialized Views and Virtual Column specifications may use “editioned” objects and PL/SQL objects may be “NONEDITIONABLE”

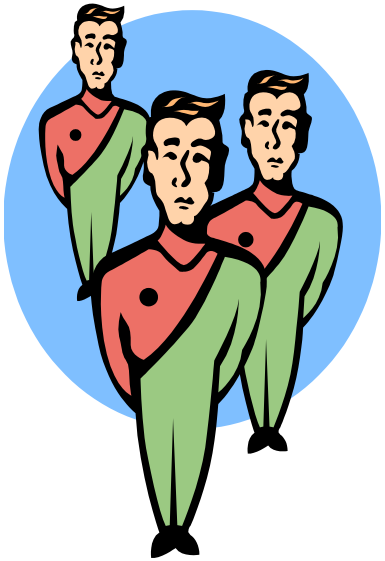
EBR Support

- EBR is safe, secure, and part of Oracle 11gR2 (both EE and SE) and 12c
- EBR is built-in and is without additional licensing cost
- EBR may require considerable design investment to work well

Oracle E-Business Suite 12.2 uses EBR to drastically reduce planned outages

How?

- Non-schema "edition" of a database's PL/SQL, views, and synonyms
  - New edition may be modified as desired then tested and deployed without impacting users of the original edition
  - Once the new edition is ready for complete rollout, it may be released



EBR Database Objects

- Editions
  - All pre-upgrade editionable objects exist in a parent edition
  - New editions inherit from parent
  - Post-upgrade objects exist in the child edition
- Editioning Views
  - Different projection of table for each edition
  - Each edition sees only its own columns
  - Changes are made safely writing only to new columns or new tables not seen in old edition
- Cross-Edition Triggers
  - Keep "old" and "new" editions synchronized

EBR Editions

- Non-schema objects that have no owner (says SYS in dba_objects but not “owned”)
- Editions are part of 11gR2 and 12c; whether you use them or not
- Name of default edition is ORA$BASE
- Database starts using single edition
- Each new edition must be the child of an existing edition
- Users (with permission) may use session-specific editions

Parent-Child Editions

- Parent edition

  Represents objects prior to changes (pre-upgrade)

- Child edition

  Represents objects after changes (post-upgrade)

Object Identification

- Oracle 11g R2 (and beyond) database objects are identified internally as:

edition_name.schema_name.object_name

(SQL and PL/SQL **do not** reference objects directly in this fashion; the current session's edition is used; SQL and PL/SQL reference objects as always)

schema_name.object_name

Oracle Catalog Support

- v$session (session_edition_id column)
- dba_editions
- dba_edition_comments
- dba_editioning_views
- dba_editioning_views_ae (ae = all editions)
- dba_editioning_view_cols
- dba_editioning_view_cols_ae
- dba_triggers
- dba_objects (edition_name column)
- more...

Edition Management

- Editions impact an entire database; only selected users should have the ability to manage editions; probably a subset of the DBA staff
- To create a new edition and make it available to a user

```
create edition yyy as child of xxx
-- Requires CREATE ANY EDITION privilege
```

EBR Access

- Access to editions is user based
- Users must be granted use of an edition
  - Schema/user owning editioned objects
  - User(s) who will be executing editioned code

```
grant use on edition yyy to someuser
```

Editioning Rules

- A schema object of an editionable type is editioned if its owner is editions-enabled; otherwise, it is potentially editioned (12c allows “NONEDITIONABLE” objects)
- A schema object of a noneditionable type is always noneditioned, even if its owner is editions-enabled
- Non-editioned 11g objects may not depend upon editioned objects (some ok 12c)
- Editioning Views must have same owner as base table

Three Ways to Use EBR

- Edition-Based Redefinition (EBR) use has three-levels of complexity
  1. EBR used to install new versions of PL/SQL, Views, or Synonyms
  2. EBR used to change user view when column definitions must be added, modified, or removed; Same as above, plus, uses Editioning Views
  3. EBR used when multiple synchronized editions of tables need to be “live” simultaneously; same as above; but adds Cross-Edition Triggers

PL/SQL, Views, Synonyms Only

- If only PL/SQL, Views, and Synonyms are changing from edition to edition little or no readying is required
  - Simplest way to use EBR, almost automatic
  - Editioning Views are not required
  - Cross-Edition Triggers are not required
- Create a new edition; post changes needed to PL/SQL, Views, and Synonyms; test with selected users; make available to all users; retire old edition

EBR Schema Changes

- A schema (user) must be “editions enabled” to have objects maintained in separate editions - to enable editions:

```
alter user someschema enable editions
```

(dba_users.editions_enabled=Y or N)

IRREVERSIBLE; cannot “undo”

- To modify an edition’s objects; a schema/ user must be granted use of the edition

```
grant use on edition yyy to someuser
```

Edition Enable APP Schema

```
alter user app_schema enable editions;
alter user app_schema enable editions
   for etype1,etype2; -- new in 12c
```

- Normally, an existing application schema is enabled for editions
- A second (non-edition-enabled) schema might be required for editionable objects that should not be editioned
  (e.g. Oracle E-Business Suite 12.2 uses two schemas: APPS and APPS_NE)

Edition Use is Session-based

- Sessions access one edition at a time via:
  - Database default edition (set by DBA)
  - Host-system environment variable of session
  - Setting edition in TNS service definition
  - Specification of edition at user login
  - Use of ALTER SESSION

Switching Editions – Admin

- Set database default edition

```
alter database
  default edition = edition_name;
```

  - Grants access to PUBLIC in database

- Edition set from environment variable

```
ORA_EDITION=edition_name
```

- **New idea from earlier session today - change JDBC connection pool settings**

Switching Editions – Service

- A service definition may override the database default edition
- DBMS_SERVICE's CREATE_SERVICE & MODIFY_SERVICE now have EDITION parameter (overrides default; NULL reverts)

```
BEGIN DBMS_SERVICE.modify_service(
   service_name => 'myservice',
   edition => 'myedition'
   modify_edition => TRUE);
END;
```

Switching Editions - Session

- Set session upon SQL*Plus login

```
connect ebr_user/ebr_user
  edition=myedition1;
```

- Set session edition

```
alter session
  set edition=edition_name;
```

User Grants

```
grant use on edition xxx
    to user
```

```
grant use on edition xxx
    to public
```

- (automatically performed by ALTER DATABASE SET DEFAULT EDITION)
- Allows selected users to log in to desired editions or switch to desired edition

Need for Editioning Views

- If only PL/SQL is changing from edition to edition; Editioning Views are not required
- If Tables might have column definitions added, removed, or altered between editions; then, Editioning Views are necessary to make sure each Edition's users see only relevant data
- If Editioning Views will be created and it is desirable to execute multiple editions in production; Cross-Edition Triggers may be needed to keep data synchronized

EBR Editioning Views

- Represent projected data from an underlying table (unaltered and unfiltered)
- May not use: Joins, Functions, Operators, Group By, Order By, or Distinct (or anything causing view to misrepresent the data)
- Act like tables and may have triggers and other table-like features
- Are referenced by ALL application code rather than the base tables; applications "think" the Editioning View is the base table

Editioning View Performance

- Since Editioning Views merely PROJECT column data; the optimizer converts all activity to use the underlying table
  - SQL referencing Editioning Views will get EXACTLY the same execution plan as SQL using the base table
  - There is no additional performance cost (other than statement parsing) involved with Editioning Views
  - Forward and reverse Cross-Edition triggers (if used) will have performance impact like any other trigger

EBR Cross-Edition Trigger

- Propagate changes between editions
  - Changes in Parent propagated to Child (Forward)
  - Changes in Child propagated to Parent (Reverse)
- Cross-Edition Triggers synchronize changes made between Parent and Child editions allowing multiple editions to be “live” at the same time without causing an outage
- Cross-Edition Triggers are temporary

What About New Data?

- If the base table has new/altered columns; data must be moved into them
- Oracle has improved things in two ways:
  - Alter Table DDL has been improved to make most column additions non-blocking and dependency-tracking was improved to allow "fine-grain dependency" in support of EBR
  - dbms_parallel_execute package may be used to make changes in smaller chunks limiting locking issues when executing Cross-Edition triggers

EBR in Use

- Current edition's (version1 – pre-Upgrade) views represent table data
- A new edition (version2 – post-Upgrade) is created perhaps building new Editioning Views in the schema; Editioning Views reference the new/changed columns using the old column names
- Cross-edition triggers fire in one edition (or the other, but only one edition); typically changes to the current edition's data will be copied/forwarded to the new version

EBR and Table Modifications

- Post-Upgrade edition processing must not interfere with any Pre-Upgrade edition processing (don't break existing code!)
- Column datatype change requires:
  - Creating a new column in the base table so that both sets of code are still valid
  - Creating Cross-Edition triggers to keep values "in-sync" if pre-upgrade and post-upgrade editions will be in use at the same time
  - Populating new/altered column values

Managing Transition

- Changes may be installed, verified, compiled, and validated without impacting the current system
- When ready; administrator may "cut over" to the new version/release by simply setting the edition for users of the database

EBR Planning - Simplest

- If EBR will only be used for PL/SQL, Views, and Synonyms:
  - Create naming convention for editions
  - Create “editions enabled” users (Oracle E-Business Suite Version 12.2 has two admins; one edition enabled, and one not)
  - Determine update and rollout strategy
- In Oracle 12c (and later) determine if any Materialized Views or virtual column definitions might change to include edition information (probable outage)

EBR Planning and Readying

- When Editioning Views will be used a "readying" process is followed:
  - Tables renamed and then replaced using Editioning Views with the original table name
  - Drop triggers from base tables
  - Create triggers on editioning views
  - Recompile PL/SQL and Views using tables
  - then test, test, and test some more

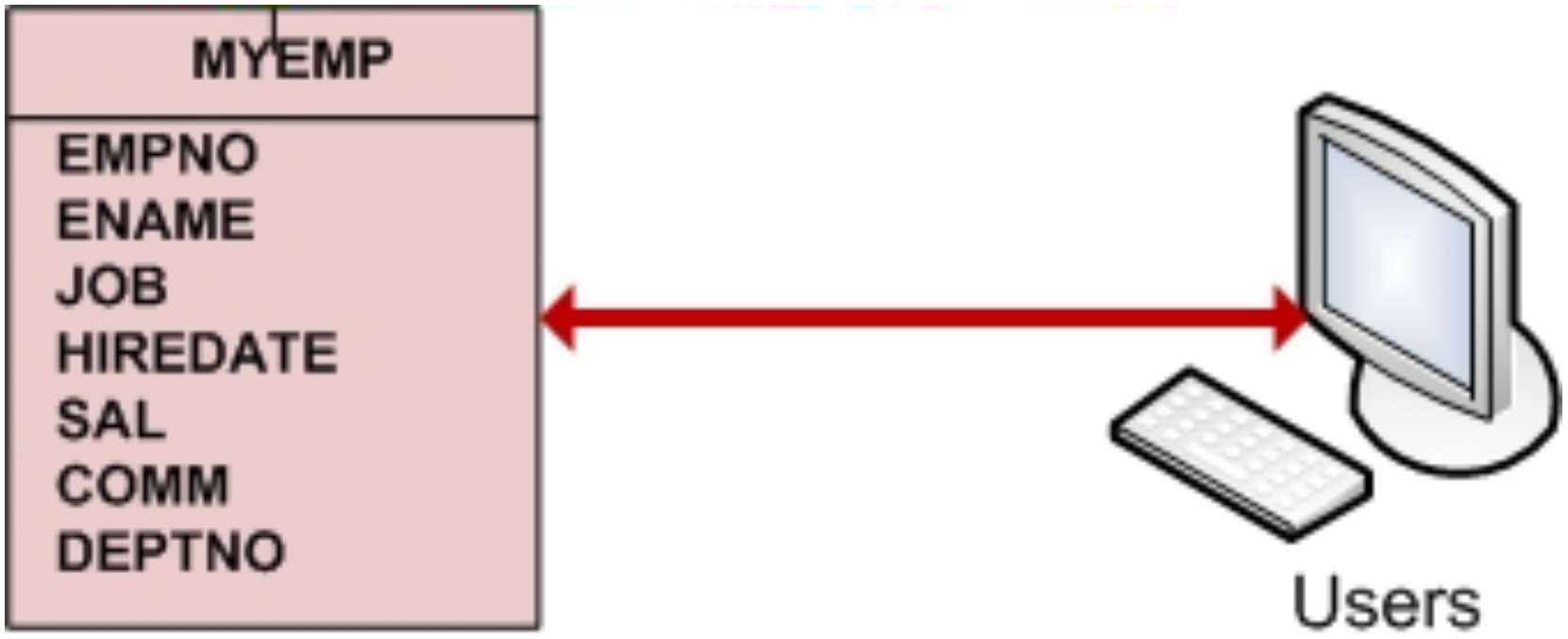
Non-Editioned Item Issues, 1

- 11gR2 Public Synonyms cannot refer to editioned objects; in 12c they may
- Function-based Indexes cannot depend upon editioned functions
- 11g Materialized Views cannot refer to editioned objects; 12c allows some use
- 11g Virtual Column expressions may not refer to editioned objects; 12c allows some use
- Oracle 12c allows PL/SQL objects to be marked as “NONEDITIONABLE”

Non-Editioned Item Issues, 2

- Tables cannot have columns based upon user-defined types (collection/ADT) whose owner is editions-enabled in 11g
  - Editioned ADTs may not be evolved
  - 12c allows selected Types to be NONEDITIONABLE
- Non-editioned subprograms cannot reference an editioned subprogram
- Editioning Views may not be part of Foreign Key definitions

EBR – Not Used Yet



Build First Edition

```
create edition myedition1
  as child of ora$base;
-- as child of xxx  is optional
-- requires "create any edition"

alter session set edition =
  myedition1;

select sys_context('Userenv',
                   'Current_Edition_name')
                   CurrentEdition
     from dual;
```

Changes to PL/SQL Only

- If changes involve only PL/SQL, Synonyms, and/or Views
  - Create new edition
  - Modify PL/SQL, Synoyms, Views
  - Make available to users

Yes, it really is that easy…

Testing New Edition PL/SQL

```
connect app_schema/app_schema
alter session set edition = myedition2;
create or replace function ebr_test …
-- test using new edition
connect ebr_user/ebr_user
  edition=myedition2;
select app_schema.ebr_test from dual;
-- test using current default edition
connect ebr_user/ebr_user;
select app_schema.ebr_test from dual;
```

Beyond PL/SQL Only

- Using EBR when table columns may change adds the need for Editioning Views
- "Ready" database objects as follows:
  - Rename table
  - Project table columns using Editioning View (Editioning View uses original table name; if columns change use new name in base table and old name in Editioning View)
  - Drop triggers from table
  - Recreate triggers on Editioning View
  - Recompile PL/SQL and Views using tables

"Readying" Oracle Objects

```
alter table myemp
  rename to myemp_table;
create or replace
  editioning view myemp
   as select
      empno,ename,job,mgr,hiredate,sal,
         comm,deptno from myemp_table;
-- drop trigger(s) on myemp_table
-- create trigger(s) on myemp
-- recompile views and pl/sql
```

EBR – Edition 1

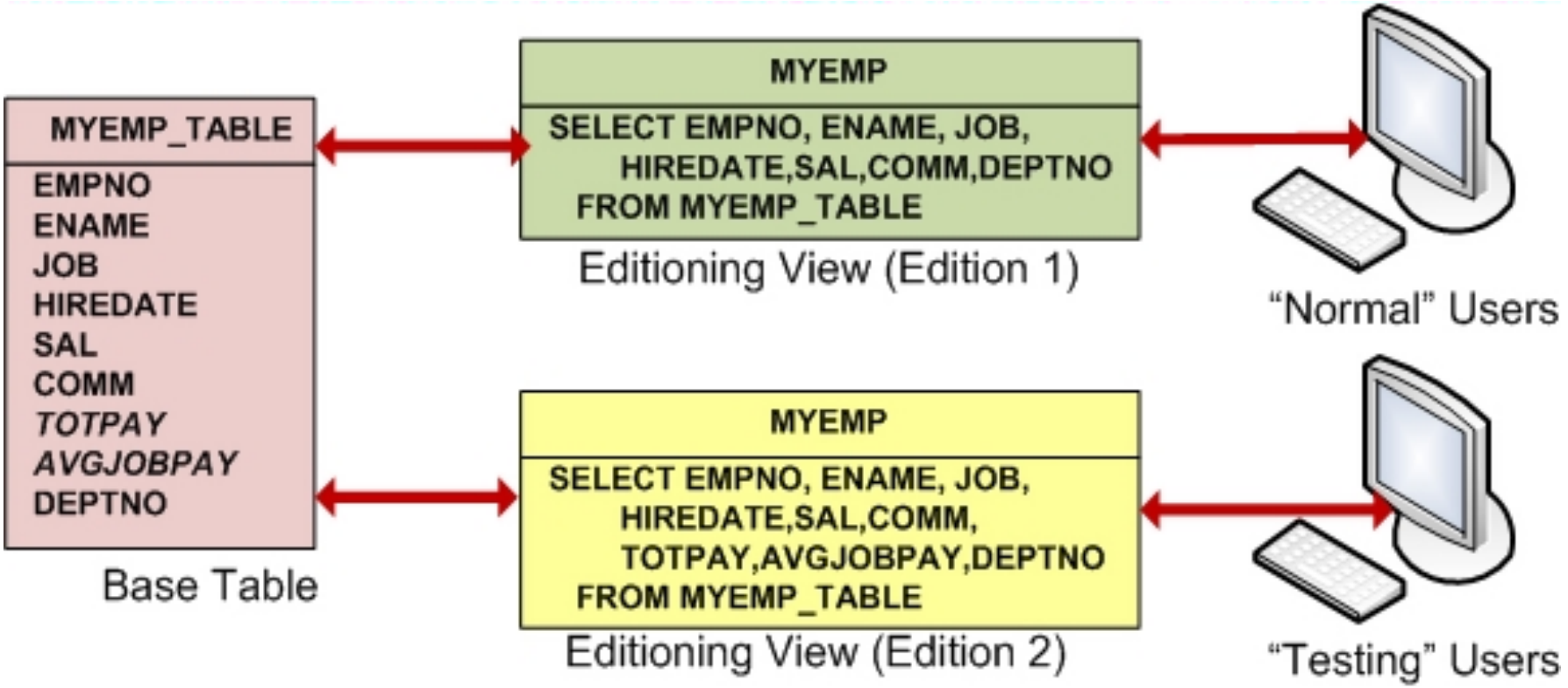
Build Next Edition

```
create edition myedition2
     as child of myedition1;

alter session
  set edition = myedition2;

select sys_context('Userenv',
                   'Current_Edition_name')
                  CurrentEdition
     from dual;
```

Alter Oracle Objects

```
alter table myemp_table
  add (totpay number(9,2)
    as (nvl(sal,0)+nvl(comm,0)));
alter table myemp_table
  add (avgjobpay number(9,2));
create or replace editioning view
  myemp as
  select empno,ename,job,mgr,hiredate,
    sal,comm,totpay,avgjobpay,deptno
    from myemp_table;
```

EBR – Edition 2



Add Cross-Edition Trigger

```
create or replace trigger
  myemp_table_avgpay
   before insert or update of sal
      on myemp_table
        for each row
        forward crossedition
        disable
   begin
      :new.avgjobpay
            := getavgjobpay(:new.job);
  end;
```

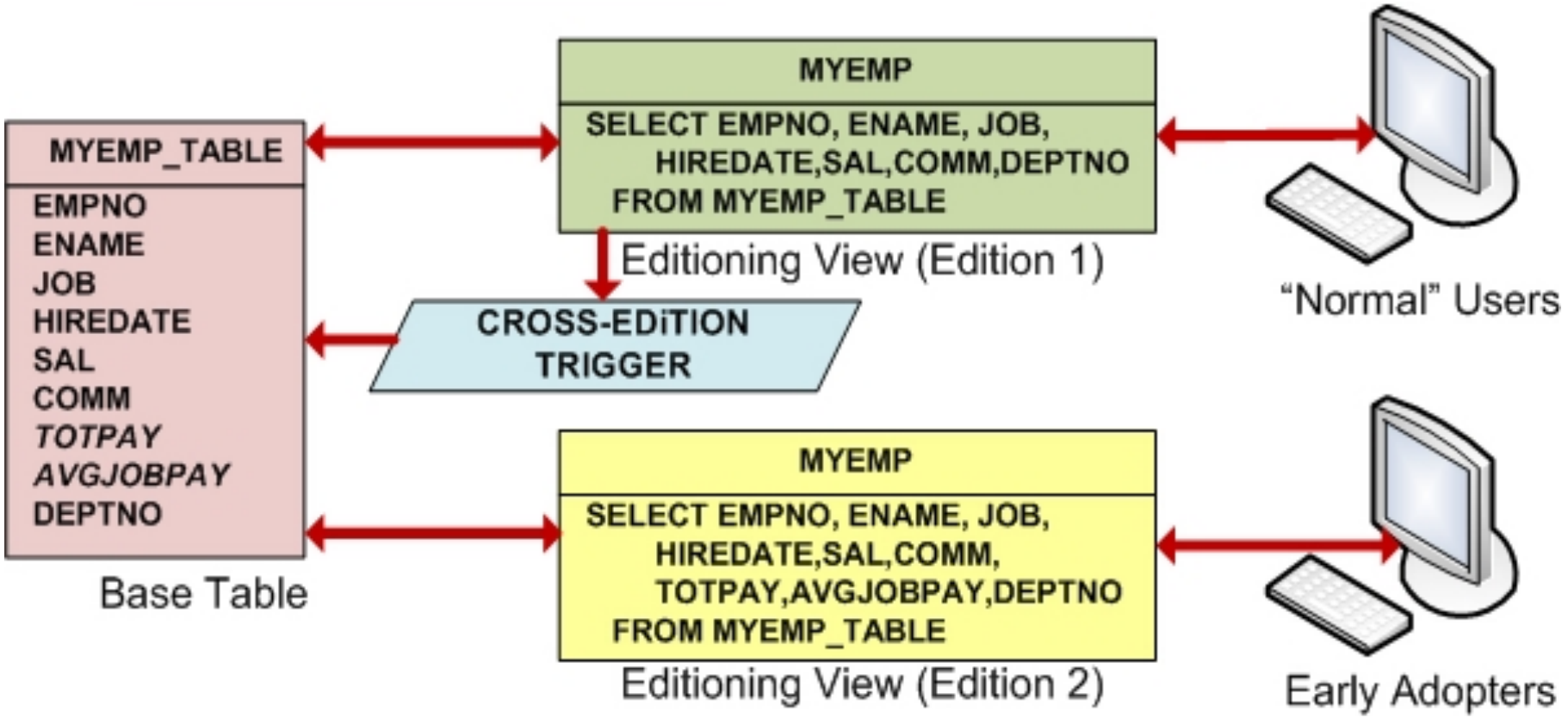
Apply Cross-Edition Trigger, 1

```
declare
   cptr number;
   retv number;
begin
  cptr := dbms_sql.open_cursor();
  dbms_sql.parse(c=>cptr,
     language_flag=>dbms_sql.native,
    statement=>'update myemp_table
      set empno = empno',
    apply_crossedition_trigger=>
        'MYEMP_TABLE_AVGPAY');
```

Apply Cross-Edition Trigger, 2

```
   retv := dbms_sql.execute(cptr);
   dbms_sql.close_cursor(cptr);
   commit;
end;
```

EBR – Edition 2 + Trigger



Testing Editions, 1

```
-- edition 2
select * from myemp;
--
-- switch to edition1 & test
alter session set edition = myedition1;
select * from myemp;
update myemp set sal = 1600, comm = 200
  where empno = 7934;
commit;
```

Testing Editions, 2

```
-- edition 1
select * from myemp;
-- edition2
alter session set edition =
  myedition2;
select * from myemp;
```

Cross-Edition Revisited

- Cross-Edition triggers should be used if:
  - Changes in one edition must be matched (approximated?) by changes in the other edition, for instance:
    - Column width or datatype changes
    - Column split/consolidation changes
    - Often needed to keep newly-changed column data inline with current users during testing (forward)
  - Two editions must be “live” for different users
- The example shown in these notes was contrived to make it simple and obvious; it is missing a “reverse” trigger

"Retiring" Editions

- Editions are retired when the system administrator revokes "use" privileges
- Once an edition is no longer in use:
  - Cross-edition triggers may (should) be removed
  - Superfluous columns may be dropped
  - Perform object "cleanup" as described on the OTN website's EBR page (see "Self-contained Edition-based Redefinition Exercise")
- Drop editions only as documented in the Advanced Application Developer's Guide

Planning for EBR Adoption

- Complexity of implementation matters!

PL/SQL Only	Pretty easy, not much planning required
Editioning Views representing table data but only one version in use at a time	More complex; requires planning of table-view creation, miscellaneous changes
Editioning Views represent tables with simultaneous changes allowed; requiring use of Editioning Triggers	Much more complex; testing must be planned thoroughly; probably requires schedule for retirement of Editioning Triggers

EBR Adoption

- EBR can be adopted all at once or it may be phased-in
  1. EBR used for PL/SQL, synonyms, & views
  2. EBR for PL/SQL, synonyms, views, and Editioning Views but only supporting one active edition at a time
     (often using “forward” Cross-Edition triggers)
  3. Using EBR for PL/SQL, synonyms, views, and Editioning Views with “Hot Rollover” enabled by both “forward” and “reverse” Cross-Edition Triggers

Readying for EBR

- Editioning View "readying" work will require an outage (hopefully your last planned one)
- Oracle suggests tables be replaced by Editioning Views in one step
  - Reduces future outages
  - Limits work required to begin using EBR
- Or, you may "ready" tables as needed
  - Requires future outages
  - Extends work required to use EBR
  - Presumes no unanticipated table relationships

Planning Issues

- Who will be EBR Administrator(s)?
- How will you control/stage editions?
- PL/SQL only or Editioning Views too?
- Will you attempt to have simultaneous updates of different versions (needing Cross-Edition Triggers)?
- Redesign of update scripts may be needed
- 12c modification of some Materialized Views, and virtual columns may be needed
- All-at-once or phased approach?

EBR 12c Improvements

- Edition-Based Redefinition made its debut in Oracle 11g and provides an ability to significantly reduce downtime due to changes in PL/SQL and/or SQL
- Oracle 12c removes some limitations present in 11gR2' implementation of EBR:
  - Materialized Views may use editioned resources
  - Virtual Columns may use editioned functions
  - Public synonyms may be editioned or noneditionable
  - User objects may be marked noneditionable

12c Materialized Views

- Materialized Views are not-editionable; but, Oracle 12c allows them to depend upon editioned objects
  - EVALUATE USING clause indicates that a referenced object is editioned (invisible otherwise)
  - ENABLE QUERY REWRITE may be limited to selected editions

EBR & Materialized Views

- CREATE/ALTER MATERIALIZED VIEW now add the ability to specify use with editioning:
  - EVALUATE USING
    CURRENT EDITION or EDITION XXX
    or NULL EDITION (eliminates editioning use)
  - ENABLE QUERY REWRITE UNUSABLE BEFORE
    CURRENT EDITION or EDITION XXX
  - ENABLE QUERY REWRITE UNUSABLE BEGINNING WITH
    CURRENT EDITION or EDITION XXX
    or NULL EDITION (eliminates editioning use)

Example Materialized View

```
create materialized view myemp_summary
  (deptno,nbr_emps,sum_sal,sum_comm,sum_totpay,avg_totpay)
  refresh complete
  start with sysdate + 1
  evaluate using current edition
  enable query rewrite unusable before edition myedition3
  as
    select myemp.deptno,count(empno),
           coalesce(sum(sal),0),coalesce(sum(comm),0),
           coalesce(sum(totpay),0),coalesce(avg(totpay),0)
      from myemp right join scott.dept
            on myemp.deptno = scott.dept.deptno
      group by myemp.deptno
      order by myemp.deptno;
```

EBR & Virtual Columns

- Non-editioned Virtual Columns may depend upon editioned objects such as editioned PL/SQL functions
  - May specify expression is to be resolved by specifying EVALUATION EDITION:
    - CURRENT EDITION
    - EDITION XXX
    - NULL EDITION
  - May use UNUSABLE EDITION or UNUSABLE BEGINNING clause (see previous pages) to limit Virtual Columns "visibility" in editions

Example Virtual Column

```
create table testit (
  empno number(4) not null,
  ename varchar2(10),
  job   varchar2(9),
  mgr   number(4),
  hiredate date,
  totpay number(8,2)
   as (get_totpay(sal,comm))
        evaluate using current edition
         unusable before edition myedition4,
  sal   number(7,2),
  comm  number(7,2));
```

More Information on EBR

- EBR support on OTN: http://www.oracle.com/technetwork/database/features/availability/ebr-455513.html
  - White Paper
  - Tutorial
  - More…
- Oracle Database Advanced Application Developer's Guide 11g Release 2 (11.2) (Part Number E25518-03)
- Bryn Llewellyn Oracle Development Tools User Group (ODTUG) interview http://www.odtug.com

Wrapping it all Up

- Oracle Edition-Based Redefinition (EBR) provides the capability to reduce (actually nearly eliminate) planned outages due to Oracle application upgrades
- Oracle 12c adds the ability to use editioned objects with Materialized Views and Virtual Columns; or make objects Noneditionable
- Oracle provides detailed supporting documentation and tutorials – use it…
- You may be approaching your LAST planned Oracle outage!

RMOUG Training Days 2016

February 9-11, 2016

(Tuesday-Thursday)

Denver Convention Center

RMOUG
ROCKY MOUNTAIN ORACLE USERS GROUP

Tracks

- Application Development
- Business Intelligence
- Database Administration
- DBA Deep Dive
- Database Tools of the Trade
- Hyperion
- Middleware
- Professional Empowerment

PHOTO CREDIT: Mike Landrum, SQL Developer and the "Data Tsunami" from i-Behavior

www.rmoug.org

COLLABORATE 16 – IOUG Forum

April 10 – 14, 2016

Mandalay Bay
Las Vegas, NV

ODTUG Kscope15

HOLLYWOOD, FLORIDA · JUNE 21-25

Please Complete Session Evaluations

Paper 781
Edition-Based Redefinition in 12c

To contact the author:

John King

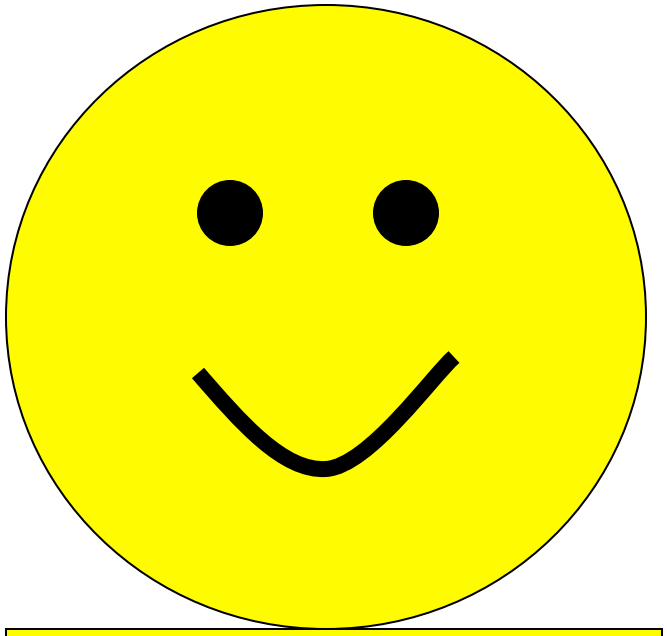
King Training Resources

P. O. Box 1780

Scottsdale, AZ USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com



Thanks for your attention!

Today's slides and examples are on the web:

www.kingtraining.com

- Adios!