Exploring Edition-Based Redefinition

Presented by: John Jay King
King Training Resources - john@kingtraining.com

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

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 & member Oak Table Network
- Providing training to Oracle and IT community for over 20 years – http://www.kingtraining.com
- "Techie" who knows Oracle, SQL, Java, and PL/SQL pretty well (along with many other topics)
- Leader in Service Oriented Architecture (SOA)
- Member of ODTUG (Oracle Development Tools User Group) Board of Directors
- Member of RMOUG and IOUG
- Home is Scottsdale, Arizona

Agenda

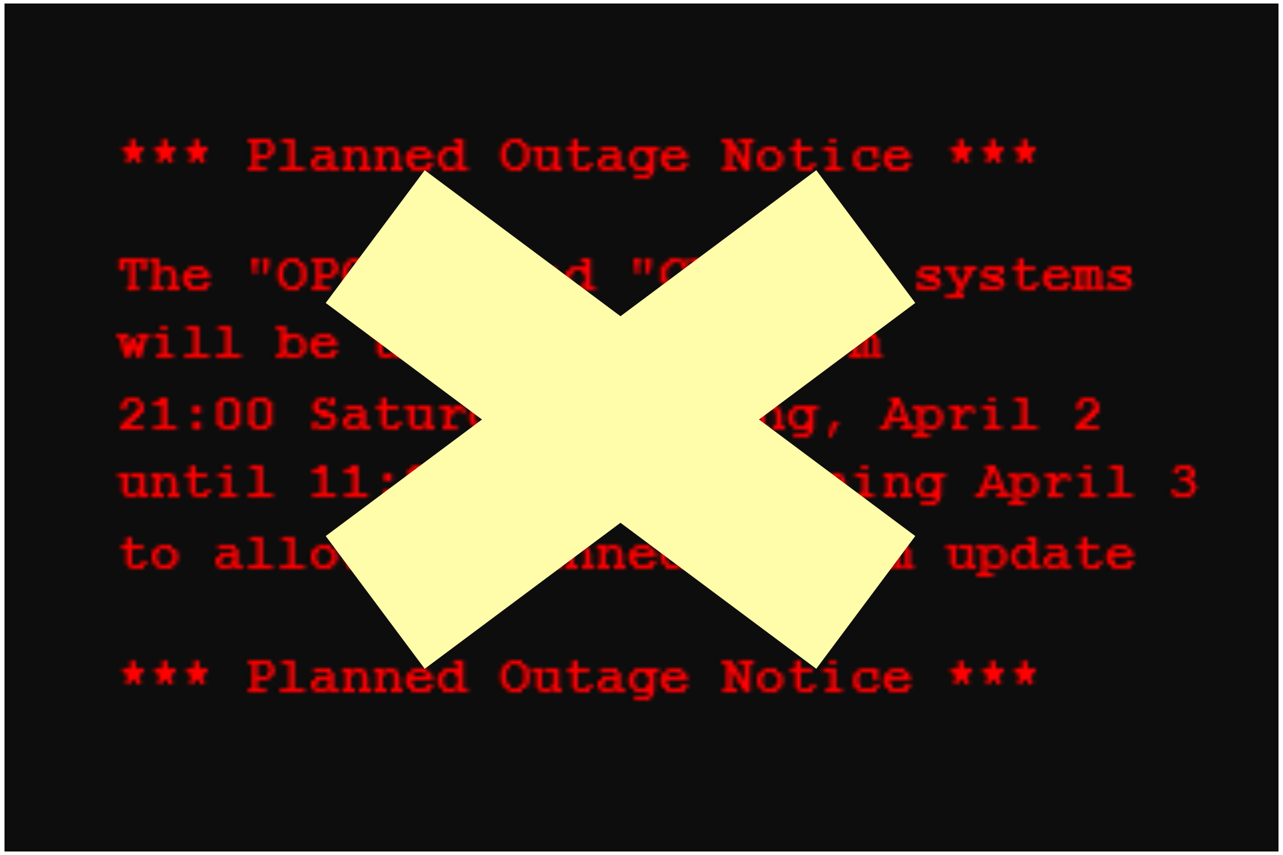
I. Introduction to EBR
II. EBR Administration
III. Editions
IV. Editioning Views
V. Cross-Edition Triggers
VI. Demo

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

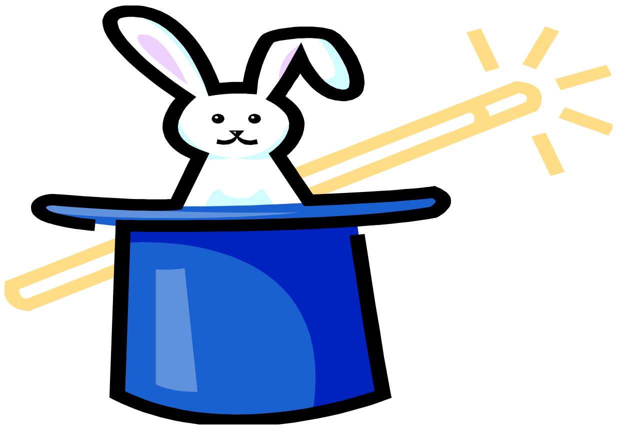
- The quest to eliminate downtime has led to a desire to provide "Online Application Upgrade"
  - An application need not be taken down when upgrades are applied
  - Users of the existing system continue uninterrupted
  - Users of the upgraded system use new code immediately

EBR to the Rescue!

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

Is EBR Magic?

- It just seems like it!
- Oracle 11gR2's 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

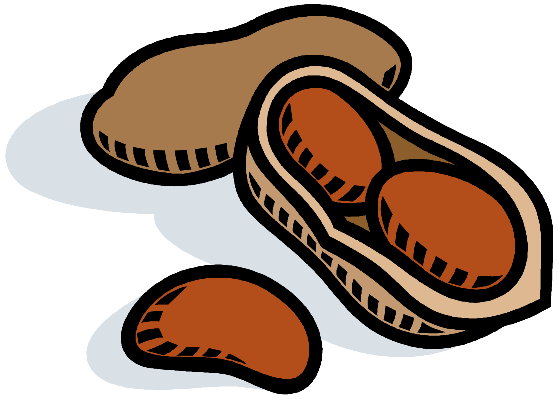


How?

- Administrators install, alter, compile, and verify Stored Procedures, Triggers, Grants, and Views (PL/SQL) using new (child) edition
- Existing systems continue uninterrupted with old (parent) edition
- Early adopters use new (child) code base others use existing (parent) code
- All users have access to upgraded system immediately after set by administrator

In a Nutshell

- EBR provides
  - Insulation mechanism where changes are made in privacy to the post-upgrade edition
  - Control mechanism allows users to actively use different editions
  - Support for “Hot Rollover” providing high availability



What is EBR?

- EBR provides a revolutionary ability to manage changes to “editionable objects”
- Editionable object types:
  - PL/SQL objects of all kinds
  - Synonyms
  - Views

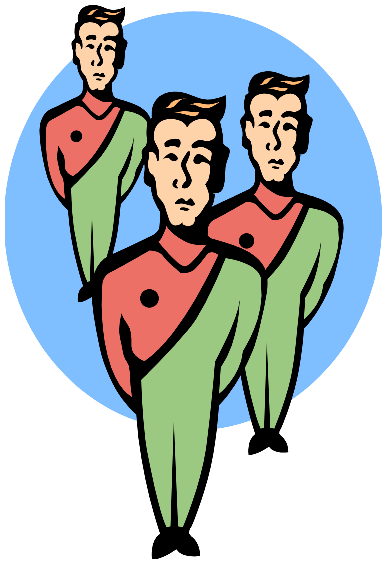
EBR Support

- EBR is safe, secure, and part of Oracle 11g (both EE and SE)
- EBR is built-in and is without additional 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



New 11g 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 Edition

- Non-schema objects that have no owner (says SYS in dba_objects but not “owned”)
- Editions are part of 11gR2; used 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

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 as:

  edition_name.schema_name.object_name

- Before Oracle 11g R2 database objects were identified as:

  schema_name.object_name

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 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 will have some performance impact

EBR Cross-Edition Trigger

- Used to propagate changes between editions
  - Changes in Parent propagated to Child (Forward)
  - Changes in Child propagated to Parent (Reverse)
- Cross-Edition Triggers allow shops to make changes like adding columns or changing table definitions without causing an outage
- Cross-Edition Triggers are temporary

Oracle Catalog Support

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

Putting EBR Together

- Cross-Edition Triggers allow shops to make changes like adding columns or changing table definitions without causing an outage
- In the past, some changes (e.g. alter column datatype) would cascade through an application and bringing the change to production would require locking table data and shutting down applications to propagate the change

Note: Oracle re-engineered Alter Table DDL to make most column additions non-blocking and improved dependency-tracking to allow “fine-grain dependency” in support of EBR

EBR in Use

- Current edition's (version1/release1) views represent table data
- A new edition (version2/release2) is created building new Editioning Views in the schema; Editioning View references 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 Triggers

- Rows/changed added in the current version will forward new data into the new version; the dbms_parallel_execute package (11g R2) is used to make changes in smaller chunks limiting locking issues
- Updates to the current version/release fire Cross-Edition triggers forwarding all changes to the new version/release
- Both current version/release and the new version/release are running simultaneously

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

EBR and Table Modifications

- Post-Upgrade edition processing must not interfere with any Pre-Upgrade edition processing
- 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 column values

EBR Access

- Access to editions is user based
- To allow a user access to enable editions

```
alter user someschema enable editions
```

(dba_users.editions_enabled=Y or N)

- To create a new edition and make it available to a user

```
create edition yyy as child of xxx
-- Requires CREATE ANY EDITION privilege
grant use on edition yyy to someuser
```

EBR Planning and Readying

- EBR use should be planned carefully
- 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

Editioning Rules

- A schema object of an editionable type is editioned if its owner is editions-enabled; otherwise, it is potentially editioned
- A schema object of a noneditionable type is always noneditioned, even if its owner is editions-enabled
- Non-editioned objects may not depend upon editioned objects
- Editioning Views must have same owner as base table

"NE on E" Prohibition

- A simple concept keeps data making sense

Non-Editioned objects may not depend upon Editioned objects

Non-Editioned Item Issues, 1

- Public Synonyms cannot refer to editioned objects
- Function-based Indexes cannot depend upon editioned functions
- Materialized Views cannot refer to editioned objects

Non-Editioned Item Issues, 2

- Tables cannot have columns based upon user-defined types (collection/ADT) whose owner is editions-enabled
  (Editioned ADTs may not be evolved)
- Non-editioned subprograms cannot reference an editioned subprogram
- Editioning Views may not be part of Foreign Key definitions

User Grants

```
GRANT USE ON EDITION xxx
      TO USER
```

```
GRANT USE ON EDITION xxx
      TO PUBLIC
```

– (automatically performed by ALTER DATABASE SET DEFAULT EDITION)

Demo Setup

The demo illustrates the following steps:

1. Create EBR users, base table(s), function
2. Create child edition, set session to use it
3. Modify function; illustrate both editions
4. Rename base table and editioning view
5. Test (should work as before)
6. Create child edition, set session to use it
7. Add columns to base table
8. Create new editioning view & triggers
9. Test editions

Create EBR Administrator

```
-- Could create new user but will
-- most like add to duties of existing
grant
   -- perhaps other permissions
   create any edition,
   drop any edition,
   alter session,
   create any view to ebr_dba;
--
alter user ebr_dba enable editions;
```

Edition Enable APP Schema

```
alter user app_schema enable editions;
```

- 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

- Editions are not selected directly by code; rather a session is associated with a specific edition in one of several ways:
  - Database service used to connect session specifies a specific edition
  - Database in use by session has default edition
  - Connection to database specifies specific edition
  - ALTER SESSION is used to switch editions

Switching Editions: Developer

- SQL*Plus users may set session on login

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

- Set database default edition

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

- Set session edition

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

Switching Editions for Users

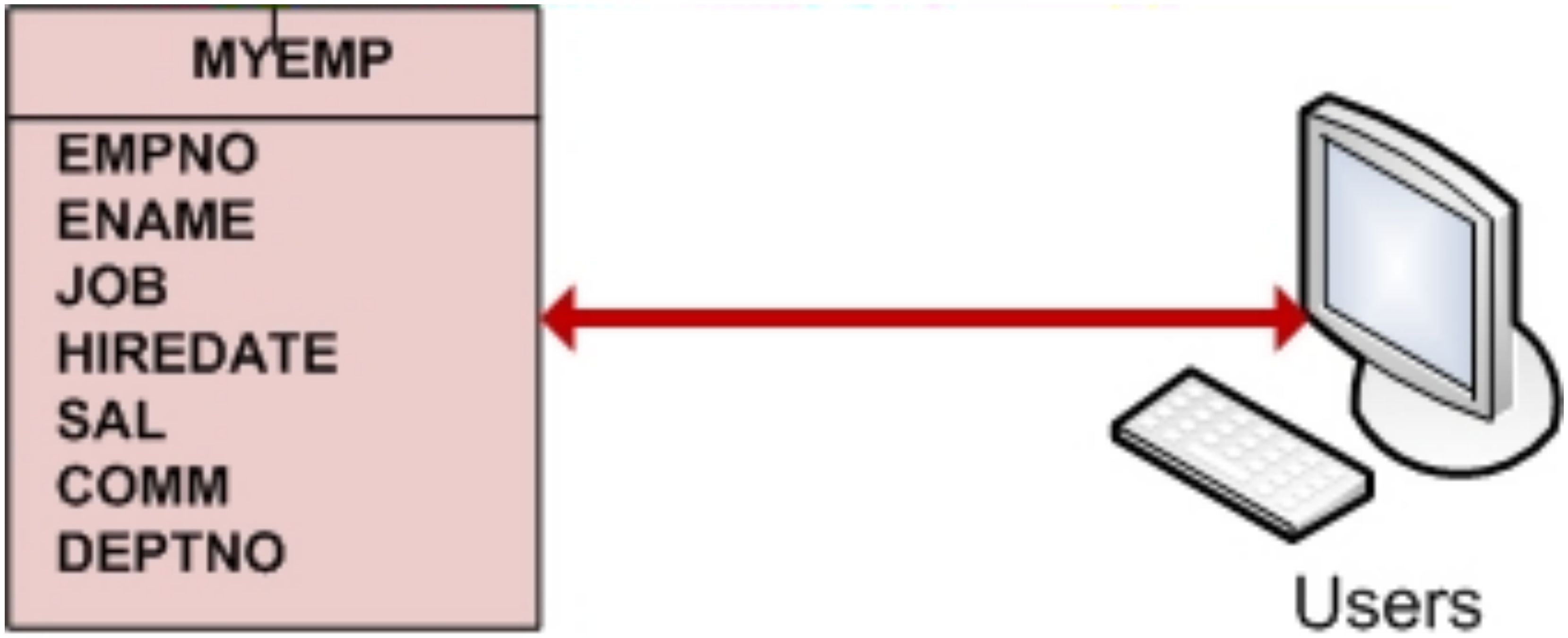
- Default edition for database

```
alter database default edition = myedition;
```

- Database service edition; DBMS_SERVICE CREATE_SERVICE and MODIFY_SERVICE now have EDITION parameter (overrides default edition, set NULL to revert)

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

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

Testing New Edition PL/SQL

```
connect app_schema/app_schema
alter session set edition = myedition1;
create or replace function my_ebr_test
  … more code here …
connect ebr_user/ebr_user
  edition=myedition1
select app_schema.my_ebr_test from
  dual;
connect ebr_user/ebr_user edition=myORA
  $BASE
select app_schema.my_ebr_test from
  dual;
```

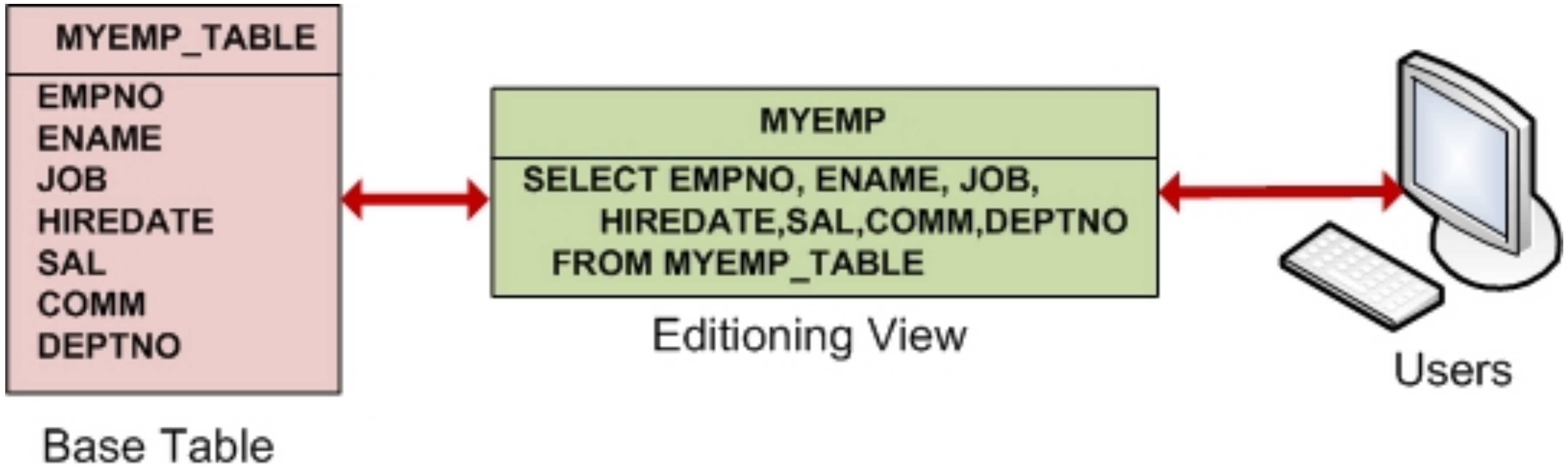
Beyond PL/SQL Only

- 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)
  - – 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



```
SELECT EMPNO, ENAME, JOB,
       HIREDATE,SAL,COMM,DEPTNO
  FROM MYEMP_TABLE
```

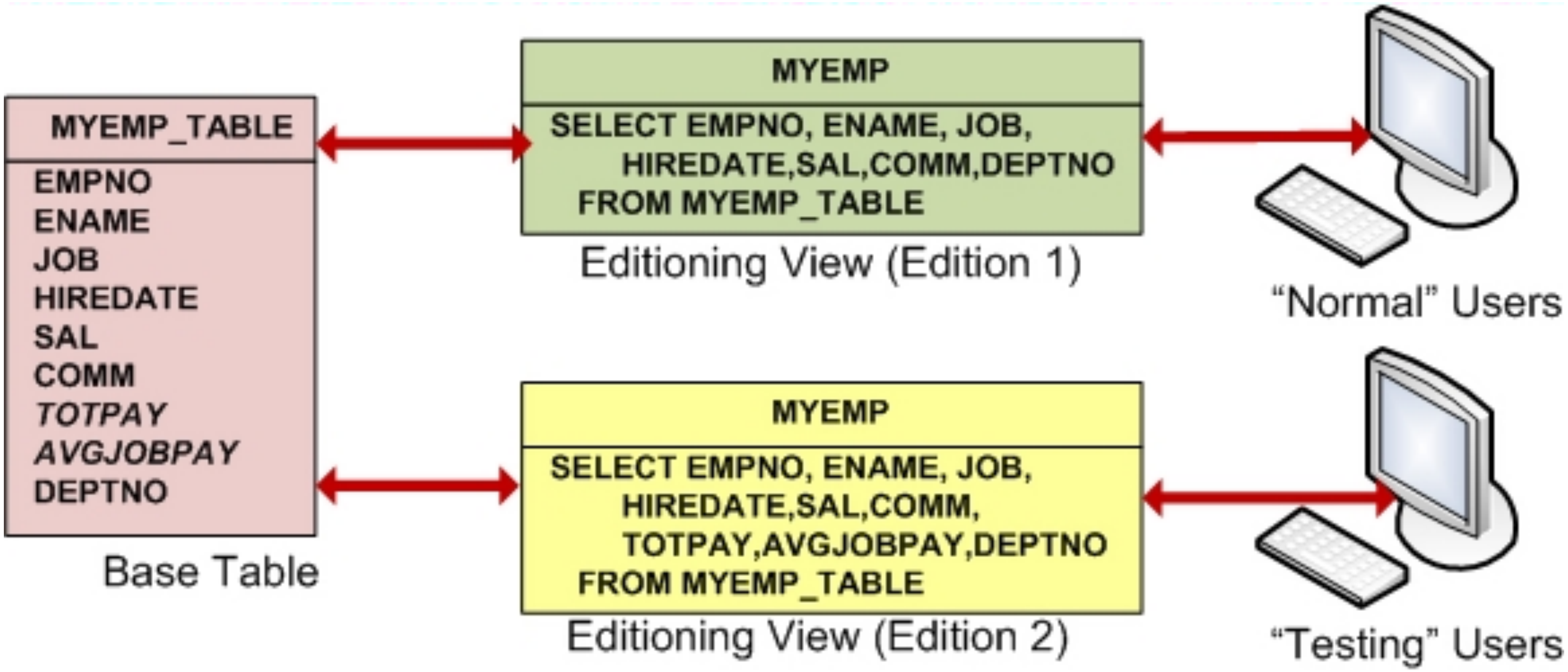
Build Next Edition

```
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


MYEMP_TABLE
EMPNO
ENAME
JOB
HIREDATE
SAL
COMM
TOTPAY
AVGJOBPAY
DEPTNO
Base Table
MYEMP
SELECT EMPNO, ENAME, JOB, HIREDATE,SAL,COMM,DEPTNO FROM MYEMP_TABLE
Editioning View (Edition 1)
“Normal” Users
MYEMP
SELECT EMPNO, ENAME, JOB, HIREDATE,SAL,COMM, TOTPAY,AVGJOBPAY,DEPTNO FROM MYEMP_TABLE
Editioning View (Edition 2)
“Testing” Users

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;
```

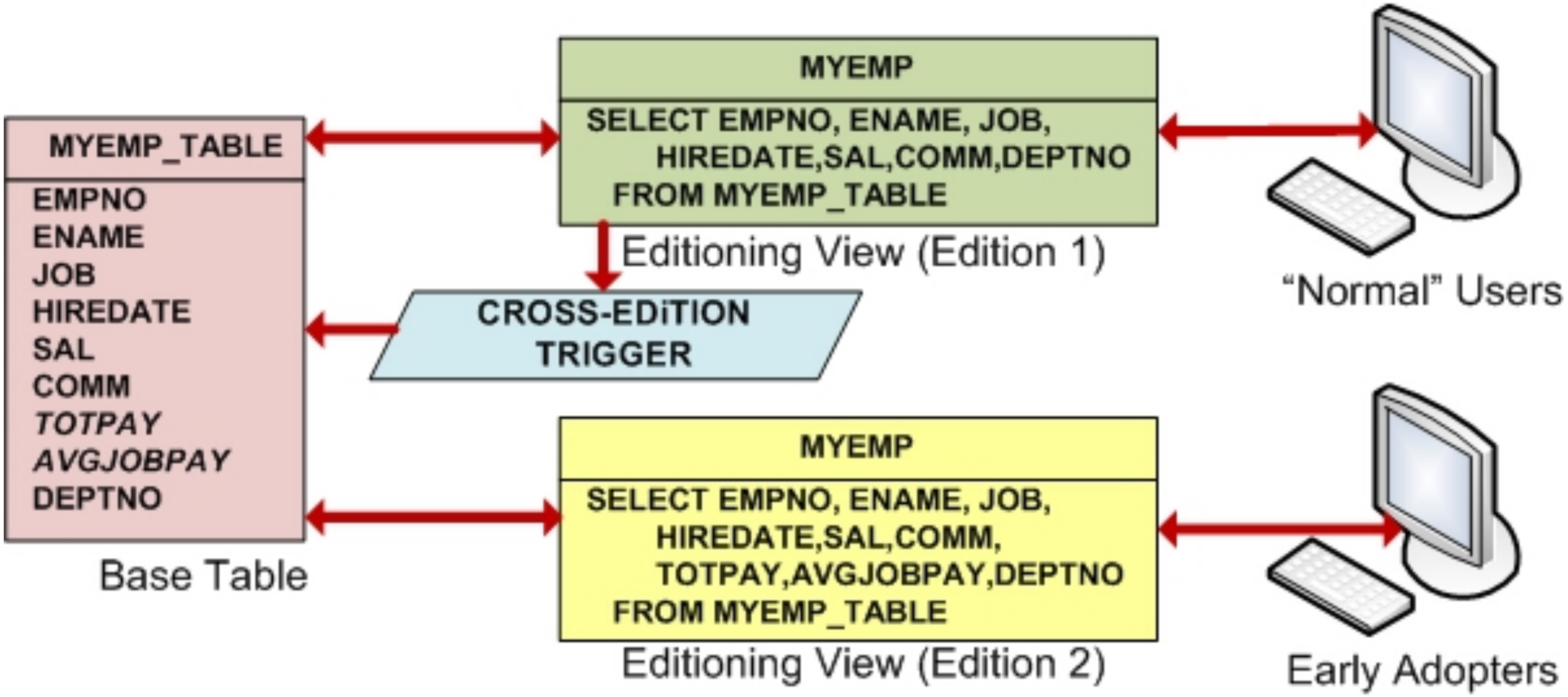
Apply Cross-Edition Trigger, 1

```
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;
alter session set edition = myedition1;
-- edition1
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:
  - Two editions are “live” for different users
  - 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
    - others
- The example shown in these notes was contrived to make the demonstration 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 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 (probably 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
- Readying tables “only as you need to”
  - Requires future outages
  - Extends work required to being using 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?
- Redesign of update scripts
- All-at-once or phased approach?

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 11g's Edition-Based Redefinition (EBR) provides the capability to reduce (actually nearly eliminate) planned outages required due to Oracle application upgrades
- Oracle provides detailed supporting documentation and tutorials – use it…
- You may be approaching your LAST PLANNED ORACLE OUTAGE!

RMOUG

ROCKY MOUNTAIN ORACLE USERS GROUP

Training Days 2014

2 Days of inexpensive Oracle-related training in Denver !!

February 6-7

February 5: University day: More low-cost training!
Check the website for details

www.rmoug.org

See you in Denver Colorado!

COLLABORATE13

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

April 2013 – Get Ready to Go!

ODTUG Kscope13

NEW ORLEANS, LA ⚜ JUNE 23-27

Topics

Application Express
ADF and Fusion Development
Developer's Toolkit
The Database
Building Better Software
Business Intelligence
Essbase
Planning
Financial Close
EPM Reporting
EPM Foundations & Data Management
EPM Business Content

ODTUG - Leading Developers into the Future with Oracle Tools

www.kscope13.com

Please Fill Out Session Evaluations

Exploring Edition-Based Redefinition

To contact the author:

John King

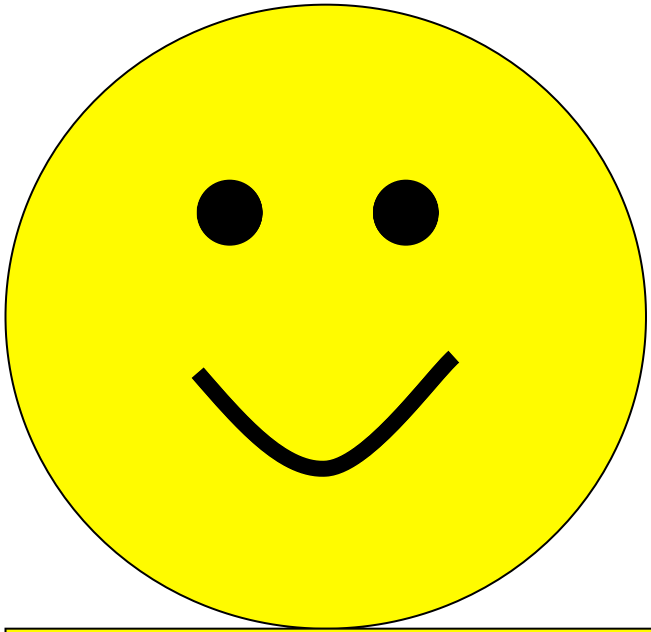
King Training Resources

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:

http://www.kingtraining.com