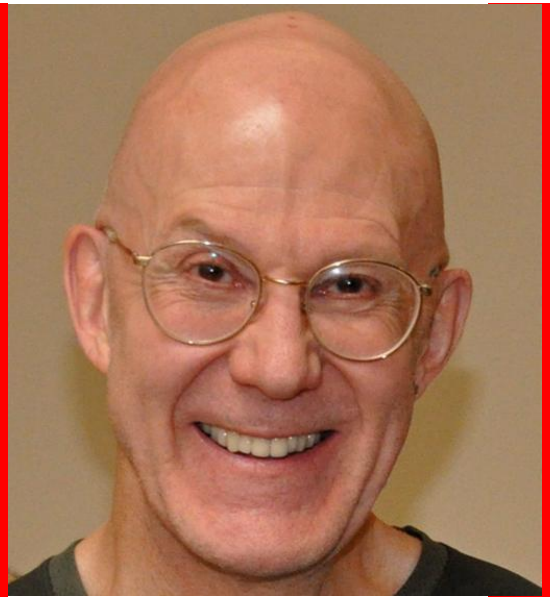


ORACLE®

Edition-based redefinition: the key to online application upgrade

Bryn Llewellyn
Product Manager, Database Server Technologies, Oracle HQ

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. 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 for Oracle's products remain at the sole discretion of Oracle.

By the way, it's free!

- Not to keep you in suspense…
- EBR is not a priced option
- Nor is it even restricted to just the Enterprise Edition
- Any customer who licenses any edition of Oracle Database 11g Release 2, or later, is entitled to use EBR

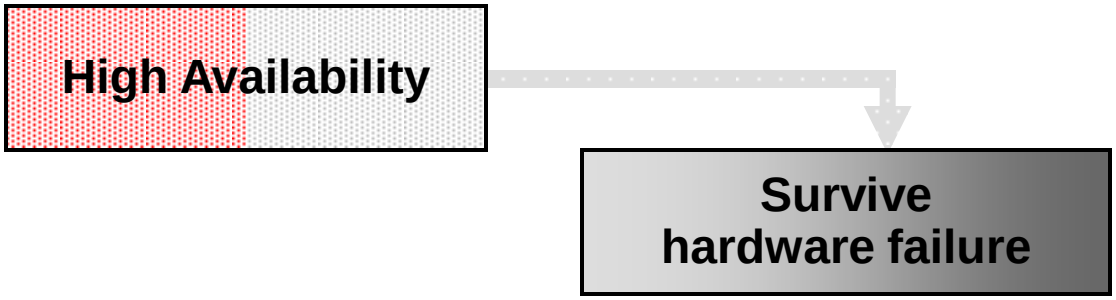
Online Application Upgrade

– the final piece of the HA jigsaw puzzle



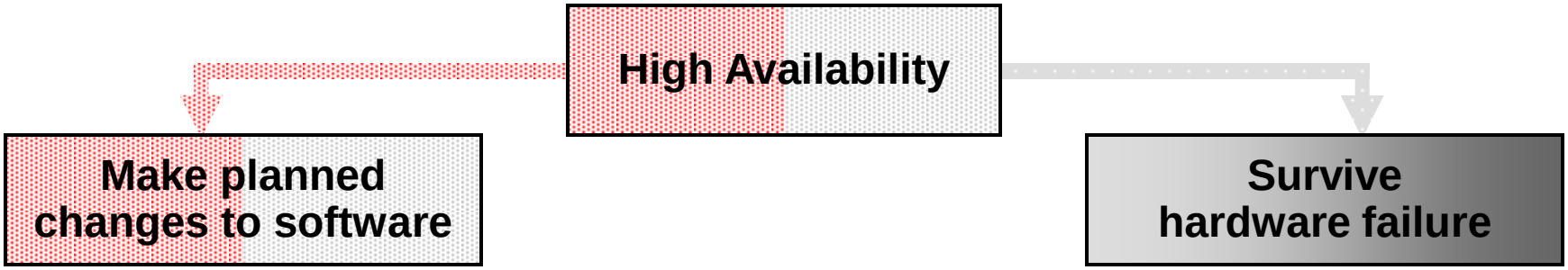
Online Application Upgrade

– the final piece of the HA jigsaw puzzle



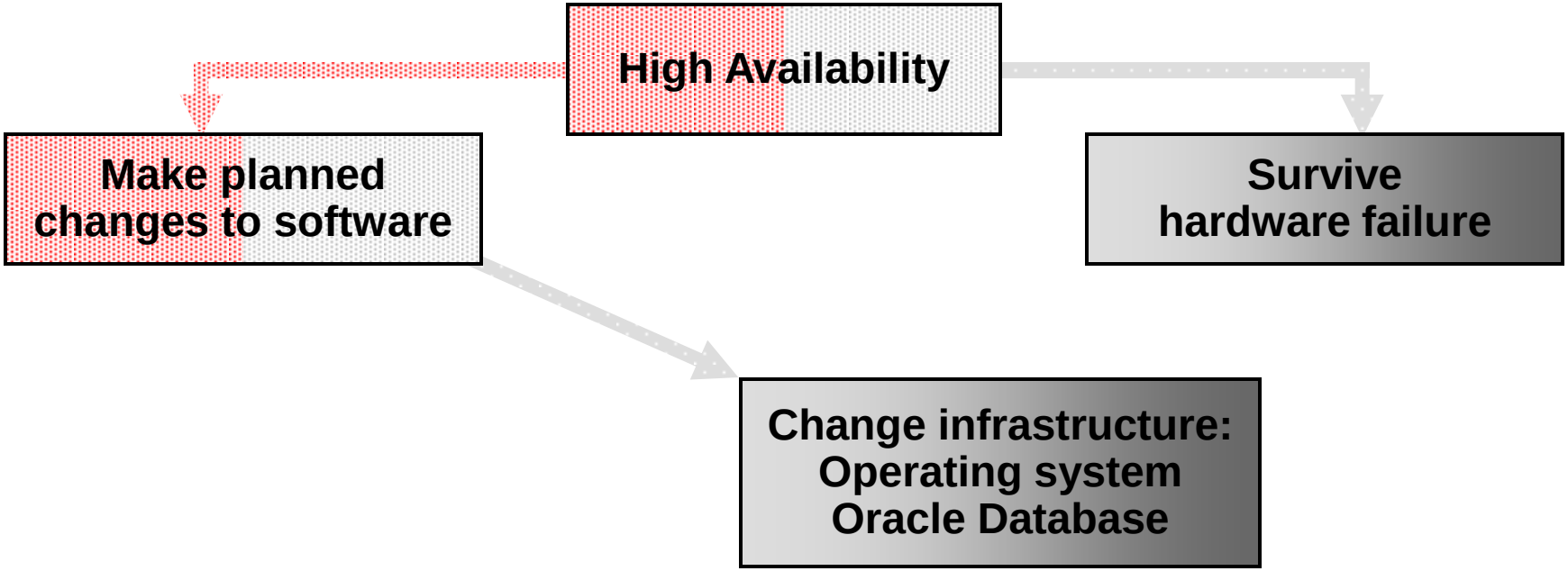
Online Application Upgrade

– the final piece of the HA jigsaw puzzle



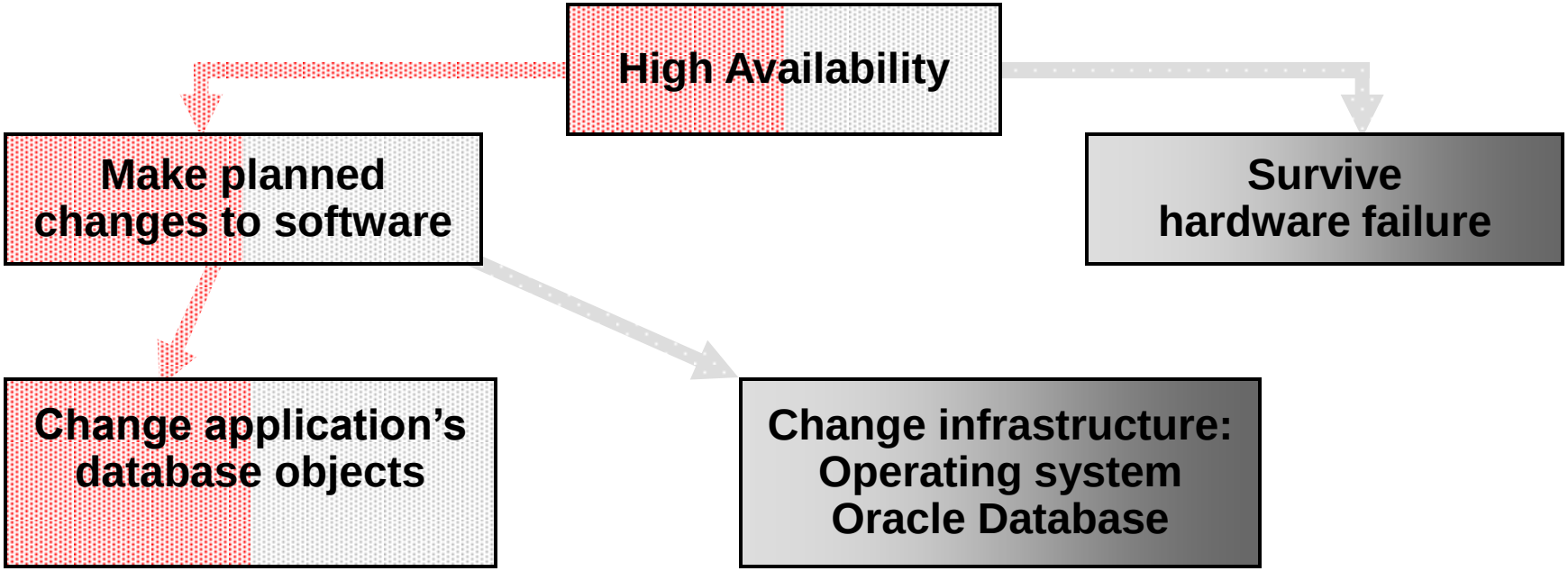
Online Application Upgrade

– the final piece of the HA jigsaw puzzle



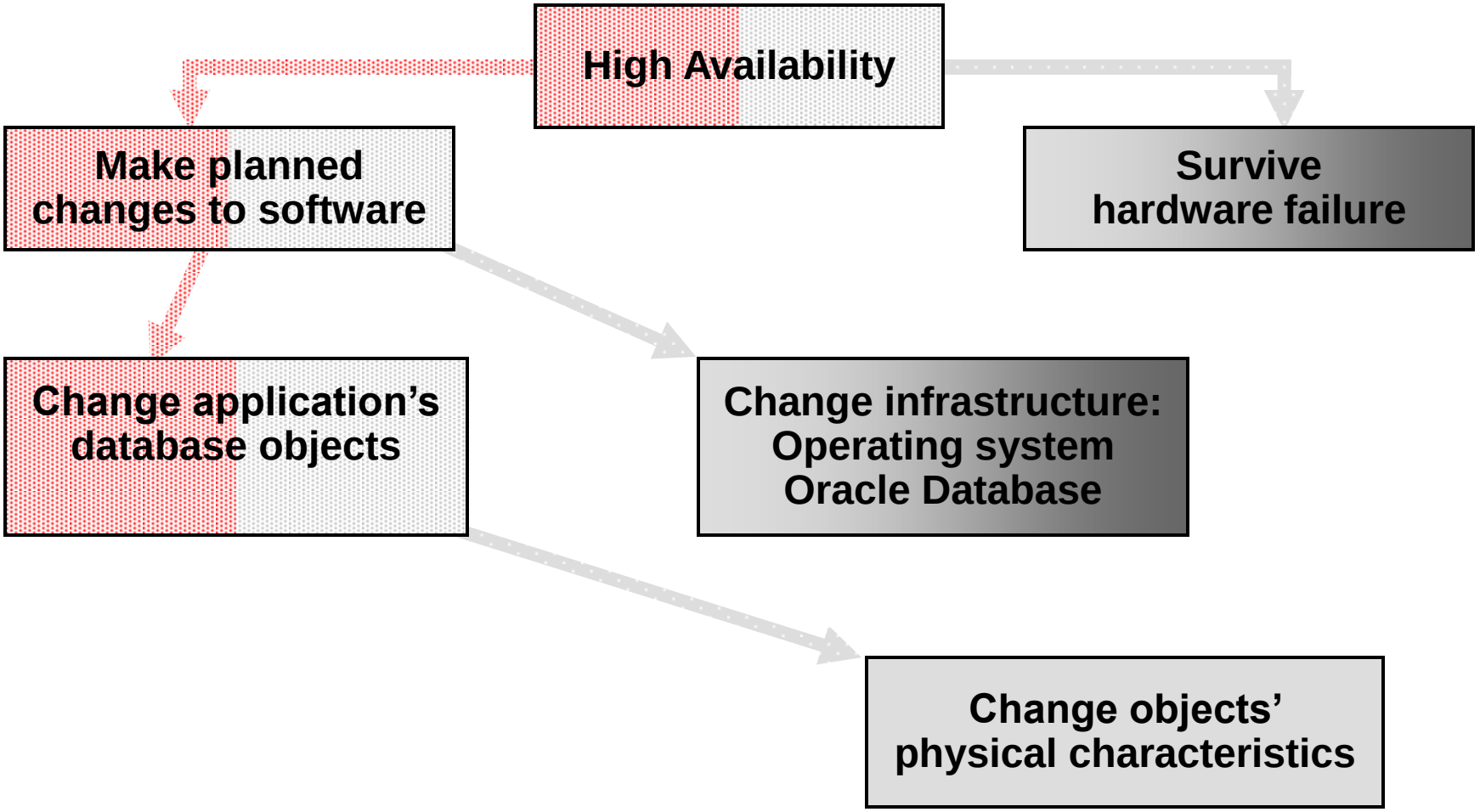
Online Application Upgrade

– the final piece of the HA jigsaw puzzle



Online Application Upgrade

– the final piece of the HA jigsaw puzzle



Online Application Upgrade

– the final piece of the HA jigsaw puzzle

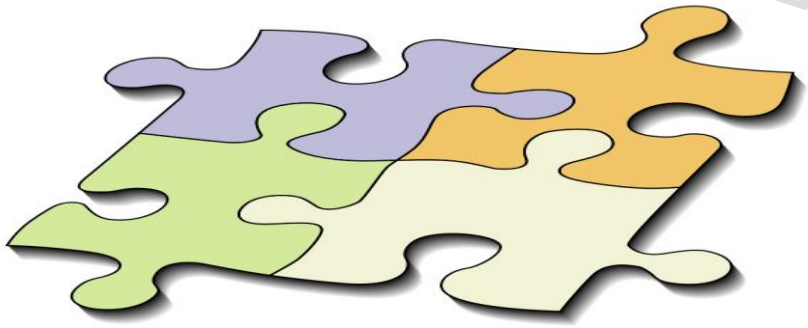
High Availability

Make planned changes to software

Survive hardware failure

Change application's database objects

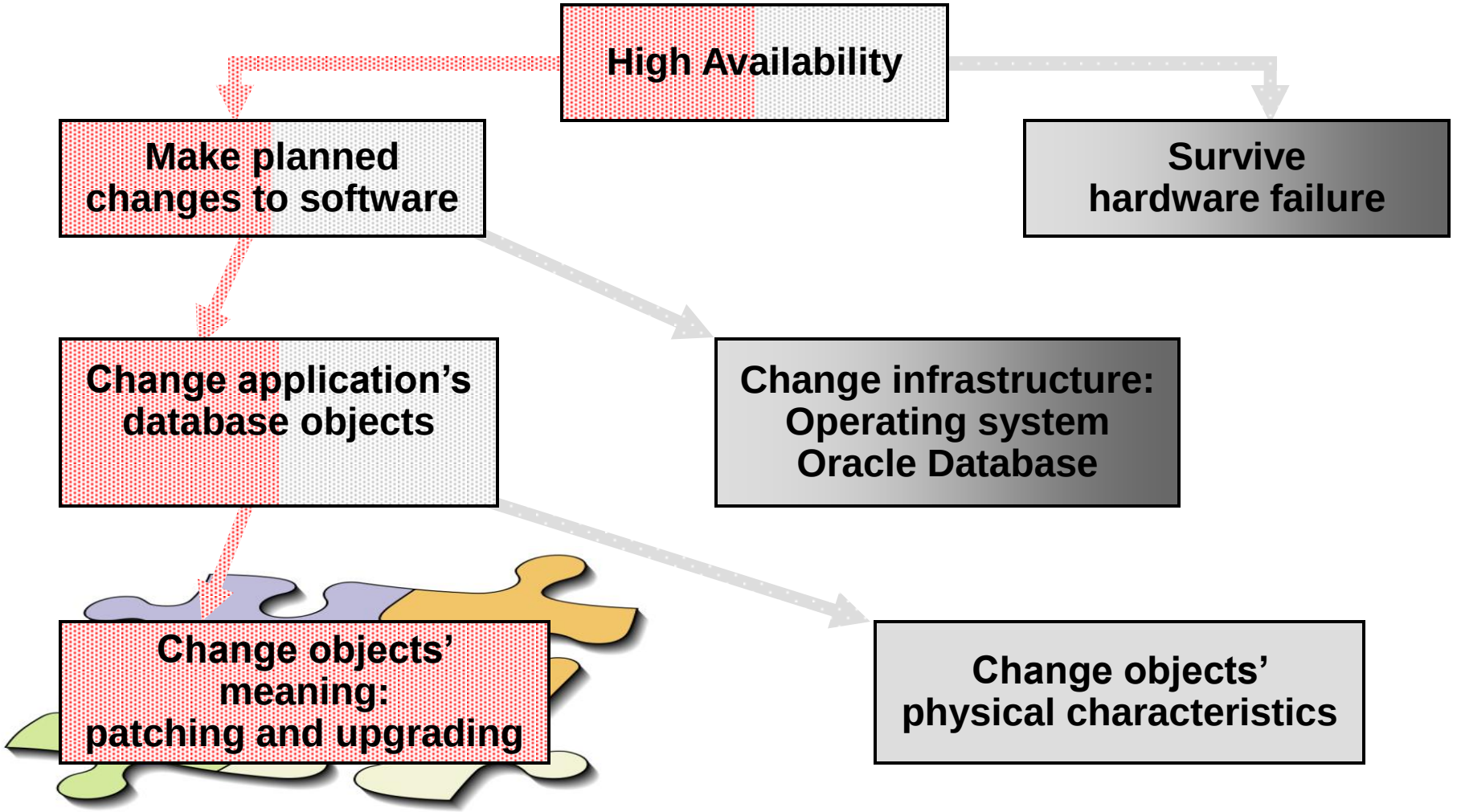
Change infrastructure: Operating system Oracle Database



Change objects' physical characteristics

Online Application Upgrade

– the final piece of the HA jigsaw puzzle



Agenda

- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- Explanation of the *edition*
- Explanation of the *editioning view*
- Explanation of the *crossedition trigger*
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

Scope

- This presentation explains new capabilities in Oracle Database 11g Release 2 that support online upgrade of the database tier of an application
- The online upgrade of other tiers of the application will need their own specific solutions – not discussed in this presentation
- The take-away from this presentation is that Oracle Database offers *both* an isolation mechanism to allow pre- and post-upgrade schemas to co-exist, *and* a way for client code to choose the particular isolated environment that it wants

Agenda

- Scope of this presentation
- **The challenge and the solution stated**
- Case study stated
- Explanation of the edition
- Explanation of the editioning view
- Explanation of the crossedition trigger
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

Online Application Upgrade

- Supporting online application upgrade means maintaining *uninterrupted availability* of the application
- But end-user sessions can last tens of minutes or longer
  - Users of the old app don't want to abandon an ongoing session
  - Users wanting to start a session must use the new app, but cannot wait until no-one is using the old app
- This implies that it must be possible to use the pre-upgrade application and the post-upgrade application at the same time – a.k.a. *hot rollover*

The challenge

- The installation of the upgrade into the production database must not perturb live users of the pre-upgrade application
  - Many objects must be changed in concert. The changes must be made *in privacy*
- Transactions done by the users of the pre-upgrade application must by reflected in the post-upgrade application
- For hot rollover, we also need the *reverse* of this:
  - Transactions done by the users of the *post*-upgrade application must by reflected in the *pre*-upgrade application

The solution: edition-based redefinition

- 11.2 brings these revolutionary new features: the *edition*, the *editioning view*, and the *crossedition trigger*
  - Code changes are installed in the privacy of a new *edition*
  - Data changes are made safely by writing only to new columns or new tables not seen by the old edition
    - An *editioning view* exposes a different projection of a table into each edition to allow each to see just its own columns
    - A *crossedition trigger* propagates data changes made by the old edition into the new edition's columns, or (in hot-rollover) vice-versa

Agenda

- Scope of this presentation
- The challenge and the solution stated
- **Case study stated**
- Explanation of the edition
- Explanation of the editioning view
- Explanation of the crossedition trigger
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

Case study

- The HR sample schema, that ships with Oracle Database, represents phone numbers in a single column:
  - Diana Lorentz 590.423.5567
  - John Russell 011.44.1344.429268
- Users now need to ring phone numbers from any country in the world
- So we want a uniform representation with two columns: Country Code; and Number Within Country.

Case study

1_Show_20_Rows_Repeat_Pre_Upgrade

First Name	Last Name	Phone
Steven	King	011.32.242.647.4719
Neena	Kochhar	708.108.8233
Lex	De Haan	205.621.9819
Alexander	Hunold	011.38.209.317.1291
Bruce	Ernst	431.800.6569
David	Austin	
Valli	Pataba	
Diana	Lorent	
Nancy	Greenb	
Daniel	Faviet	
John	Chen	
Ismael	Sciarr	
Jose Manuel	Urman	
Luis	Popp	
Den	Raphae	
Alexander	Khoo	
Shelli	Baida	
Sigal	Tobias	
Guy	Himuro	
Karen	Colmen	

2_Show_20_Rows_Repeat_Post_Upgrade

First Name	Last Name	Cntry #
Steven	King	+32 242-647-4719
Neena	Kochhar	+1 708-108-8233
Lex	De Haan	+1 205-621-9819
Alexander	Hunold	+38 209-317-1291
Bruce	Ernst	+1 431-800-6569
David	Austin	+1 207-718-4492
Valli	Pataballa	+45 662-851-6340
Diana	Lorentz	+47 951-260-7204
Nancy	Greenberg	+1 434-531-4024
Daniel	Faviet	+1 665-985-7057
John	Chen	+1 189-665-7741
Ismael	Sciarra	+1 235-670-4647
Jose Manuel	Urman	+1 782-701-7504
Luis	Popp	+40 224-526-8013
Den	Raphaely	+35 385-817-3247
Alexander	Khoo	+1 754-359-1977
Shelli	Baida	+36 949-238-6312
Sigal	Tobias	+40 440-870-9983
Guy	Himuro	+1 550-189-7906
Karen	Colmenares	+1 411-573-4711

Agenda

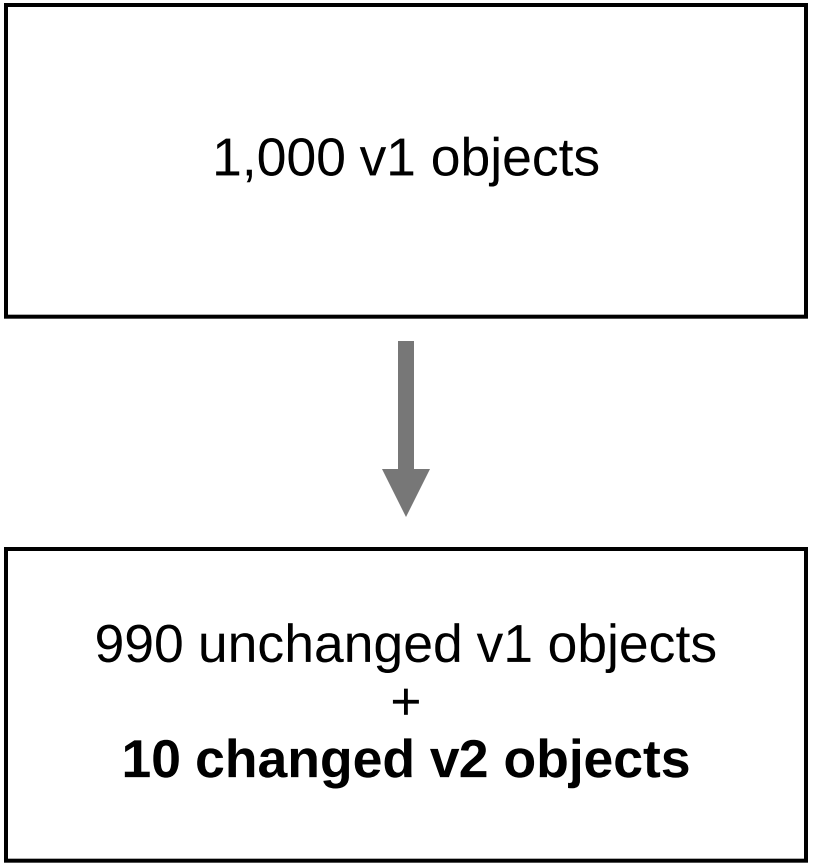
- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- **Explanation of the *edition***
- Explanation of the editioning view
- Explanation of the crossedition trigger
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

Application versioning: the challenge

- Scenario
  - The application has 1,000 mutually dependent tables, synonyms, views, PL/SQL units, and other objects
  - There's more than one schema
  - They refer to each other by name – and often by schema-qualified name
  - The upgrade needs to change 10 of these

Application versioning: the challenge

Pre-upgrade app



Post-upgrade app

Application versioning: the challenge

- Of course, you can't change the 10 objects in place because this would change the pre-upgrade app
- How can an old and a new occurrence of the "same" object co-exist?
- Through 11.1, the only dimensions that determine which object you mean, when one object refers to another, are its name and its owner
- In short, the *naming mechanisms* through 11.1 are not rich enough to support online application upgrade

The solution: editions

- 11.2 introduces the new nonschema object type, *edition* – each edition can have its own private occurrence of “the same” object
- A database must have at least one edition
- You create a new edition as the child of an existing edition – and an edition can’t have more than one child
- A database session specifies which edition to use

  (of course, the database has a default edition)

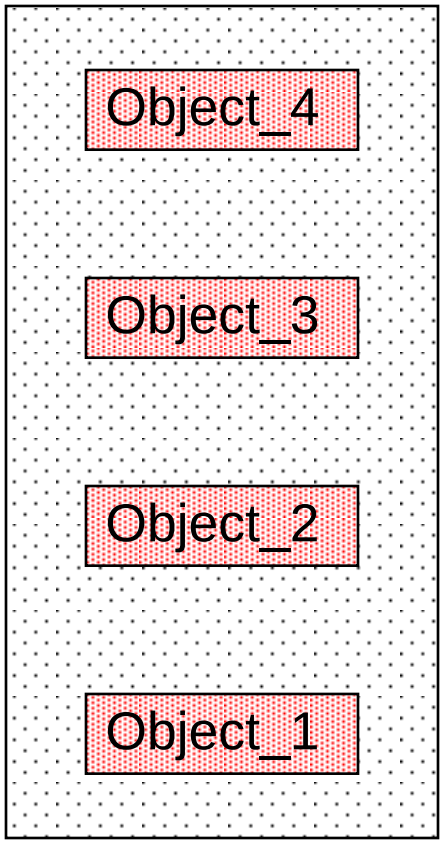
The solution: editions

- Through 11.1, an object is identified by its name and its owner
- From 11.2, an editioned object is identified by its name, its owner, and the edition where it was created
- However, when you identify it you can mention *only* its name and owner. This reference is interpreted in the context of a current edition
  - live SQL
  - the text of a stored object

Editions: mental model

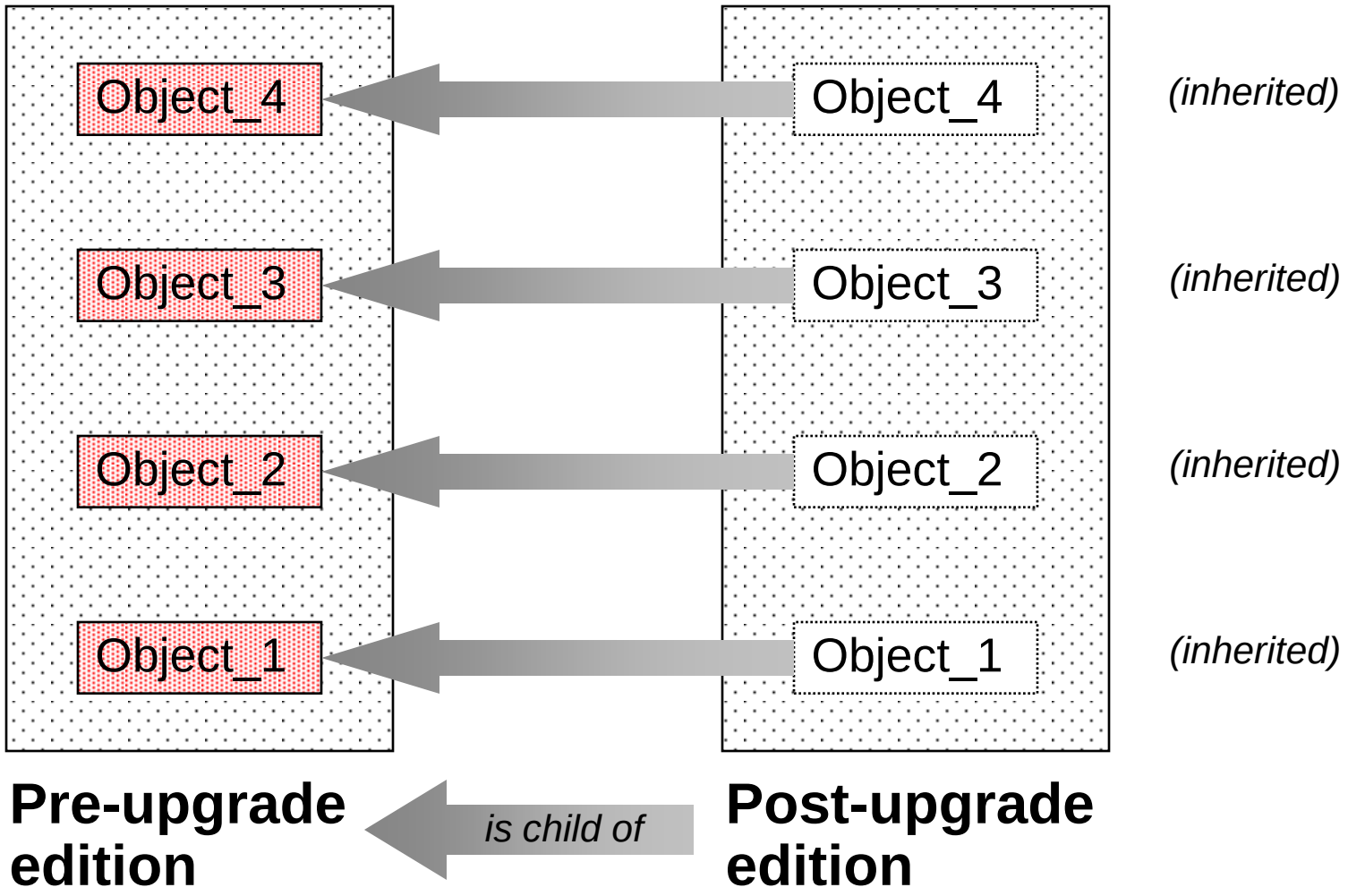
- When you create a new edition, every editioned object in the parent edition is copied into the new edition

Editions: implementation model

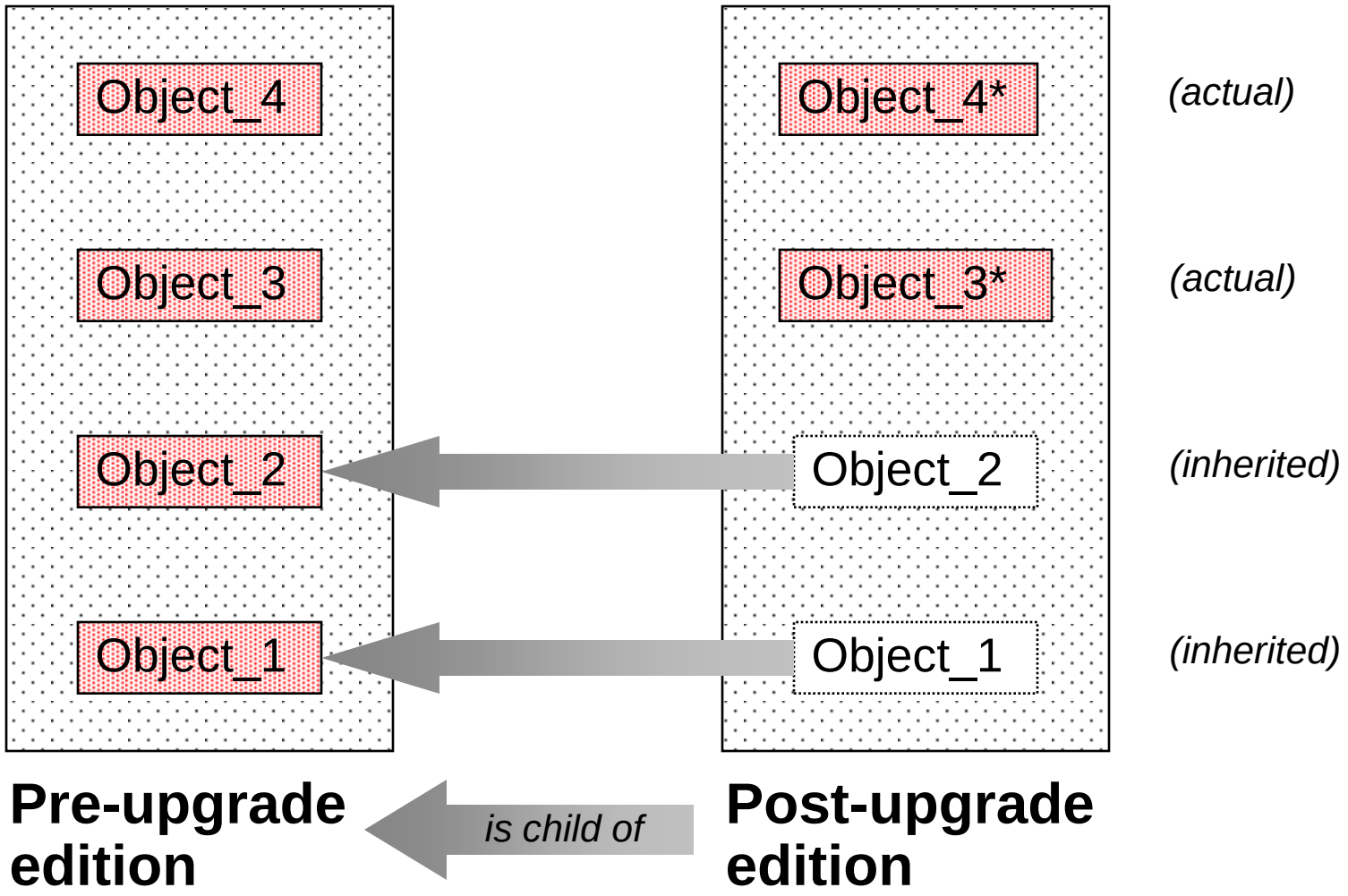


Pre-upgrade edition

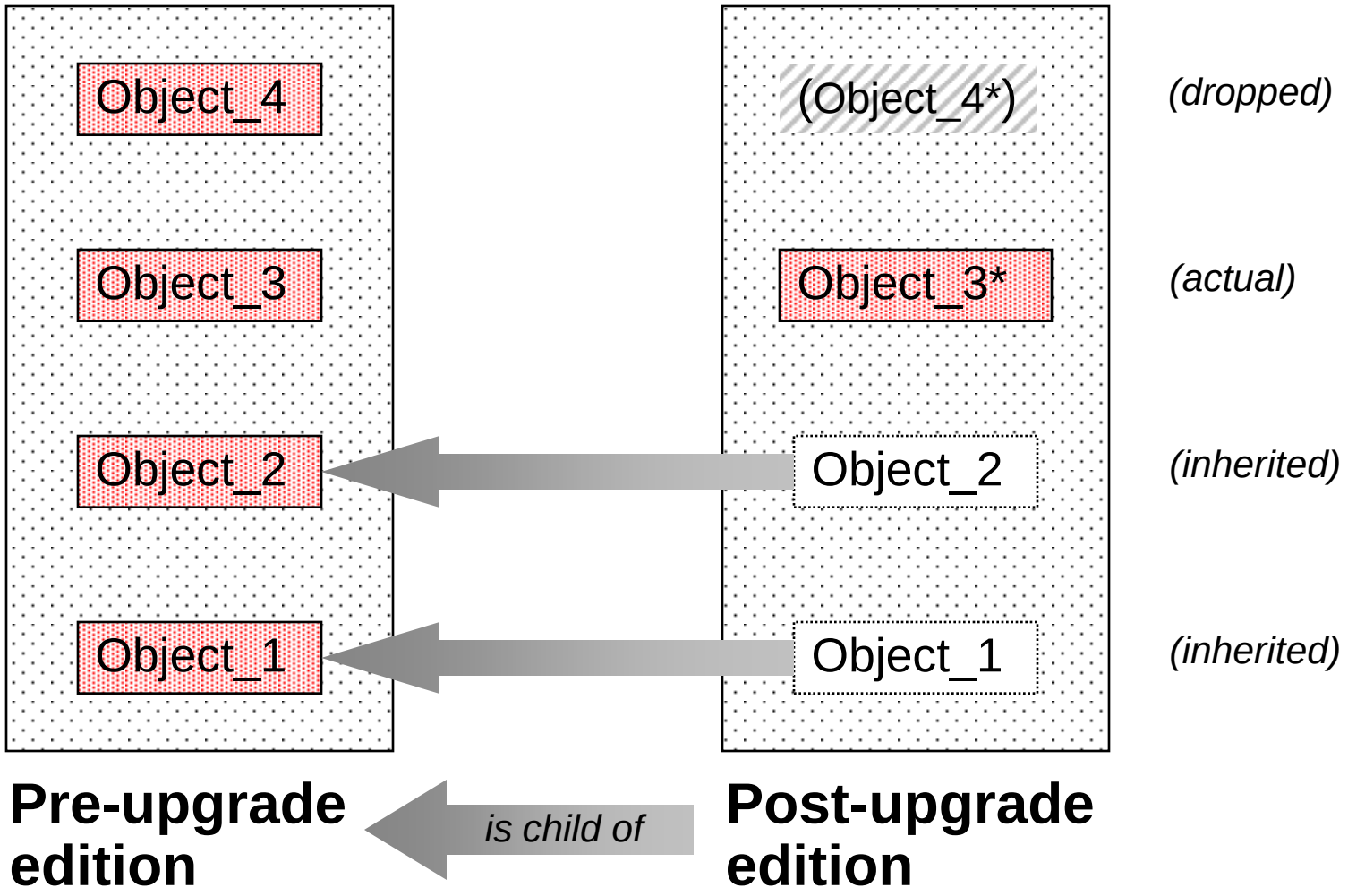
Editions: implementation model



Editions: implementation model



Editions: implementation model



Editions

- If your upgrade needs only to change synonyms, views, or PL/SQL units, you now have all the tools you need

Agenda

- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- Explanation of the *edition*
- **Explanation of the editioning view**
- Explanation of the *crossedition trigger*
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

Editionable and noneditionable object types

- Not all object types are editionable
  - Synonyms, views, and PL/SQL units of all kinds (including, therefore, triggers and libraries), and are editionable
  - Objects of all other object types – for example tables – are noneditionable
- You version the structure of a table manually
  - Instead of *changing* a column, you *add a replacement* column
  - Then you rely on the fact that a *view* is editionable

The solution: editioning views

- An *editioning view* may only project and rename columns

Editioning views

- Think of it like a noneditioned *physical* table body with an editioned *logical* table spec
- So, of course, you can't have more than one editioning view for a particular table in a particular edition
- The EV must be owned by the table's owner
- Application code should refer only to the logical world
- You can create table-style triggers (*before* or *after statement* or *each row*) on an editioning view using the "logical" column names
- A SQL optimizer hint can request an index on the physical table by specifying the "logical" column names

Editioning views

- If you can tolerate only read access to the underlying data for an editioning view that the upgrade will change*, you now have all the tools you need
- Like all views, an editioning view can be read-only
- Ordinary SQL updates can be used safely to install values in the replacement columns – there's no DML that might be missed because of SQL's read-consistency

* Think of "configuration data"
plus… it's probably acceptable to freeze the catalog of wares for a store's online shopping site

Editioned and noneditioned objects – *slight return*

- An object whose type is noneditionable is never editioned
- An object whose type is editionable is editioned only when its owner is *editions-enabled*
- *Theorem:* a noneditioned object cannot depend on an editioned object
  - For example, a table cannot depend on an editioned type
  - If you want to use a type as the datatype for a column, its owner must not be editions enabled

Agenda

- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- Explanation of the *edition*
- Explanation of the *editioning view*
- **Explanation of the *crossedition trigger***
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

What if DML cannot stop during upgrade?

- If the upgrade needs to change the structure that stores transactional data – like the orders customers make using an online shopping site – then the installation of values into the replacement columns must keep pace with these changes
- Triggers have the ideal properties to do this safely
- Each trigger must fire appropriately to propagate changes to pre-upgrade columns into the post-upgrade columns – and vice versa

The solution: crossedition triggers

- Crossedition triggers directly access the table.
- The new crossedition trigger has special firing rules
- You create crossedition triggers in the *Post_Upgrade* edition
  - The paradigm is: don't interfere with the *Pre_Upgrade* edition
- The firing rules rules assume that
  - Pre-upgrade columns are changed – by ordinary application code – only by sessions using the *Pre_Upgrade* edition
  - Post-upgrade columns are changed only by sessions using the *Post_Upgrade* edition

The solution: crossedition triggers

- A *forward* crossedition trigger is fired by application DML issued by sessions using the *Pre_Upgrade* edition

- A *reverse* crossedition trigger is fired by application DML issued by sessions using the *Post_Upgrade* edition

- The SQL that a crossedition trigger issues always executes in the edition that owns it:
  the *Post_Upgrade* edition

  (even though, for a forward crossedition trigger, the session is using the *Pre_Upgrade* edition)

Why such a long name?

- DDL stands for *data definition language*
- *“create or replace”* and *“alter”* re-define an existing object
- These bare commands are *in-place redefinition*
- Online table redefinition (there’s that word again) creates a secret copy, keeps it in step, and then does the twizzle. Similar for online index rebuild
- This is *copy-based redefinition*
- Edition-based redefinition lets you redefine *many* objects in concert

Agenda

- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- Explanation of the *edition*
- Explanation of the *editioning view*
- Explanation of the *crossedition trigger*
- **Case study explained**
- EBR exercise vs offline upgrade: incremental extra effort
- Conclusion / Q&A

Case study

1_Show_20_Rows_Repeat_Pre_Upgrade

First Name	Last Name	Phone
Steven	King	011.32.242.647.4719
Neena	Kochhar	708.108.8233
Lex	De Haan	205.621.9819
Alexander	Hunold	011.38.209.317.1291
Bruce	Ernst	431.800.6569
David	Austin	
Valli	Pataba	
Diana	Lorent	
Nancy	Greenb	
Daniel	Faviet	
John	Chen	
Ismael	Sciarr	
Jose Manuel	Urman	
Luis	Popp	
Den	Raphae	
Alexander	Khoo	
Shelli	Baida	
Sigal	Tobias	
Guy	Himuro	
Karen	Colmen	

2_Show_20_Rows_Repeat_Post_Upgrade

First Name	Last Name	Cntry #
Steven	King	+32 242-647-4719
Neena	Kochhar	+1 708-108-8233
Lex	De Haan	+1 205-621-9819
Alexander	Hunold	+38 209-317-1291
Bruce	Ernst	+1 431-800-6569
David	Austin	+1 207-718-4492
Valli	Pataballa	+45 662-851-6340
Diana	Lorentz	+47 951-260-7204
Nancy	Greenberg	+1 434-531-4024
Daniel	Faviet	+1 665-985-7057
John	Chen	+1 189-665-7741
Ismael	Sciarra	+1 235-670-4647
Jose Manuel	Urman	+1 782-701-7504
Luis	Popp	+40 224-526-8013
Den	Raphaely	+35 385-817-3247
Alexander	Khoo	+1 754-359-1977
Shelli	Baida	+36 949-238-6312
Sigal	Tobias	+40 440-870-9983
Guy	Himuro	+1 550-189-7906
Karen	Colmenares	+1 411-573-4711

Pre-11.2 design

- Application code accesses tables directly, in the ordinary way

Readying the application for editions

- "Slide in" an editioning view in front of every table
  - Rename each table to an obscure but related name (e.g. an exotic name that ends with underscore)
  - Create an editioning view for each table that has the same name that the table originally had
  - "Move" triggers to the editioning views … *(next slide)*
  - Revoke privileges from the tables and grant them to the editioning views
  - Move VPD policies to the editioning views
- There may be a need for schema reorganization to accommodate the rule that a noneditioned object cannot depend on an editioned object

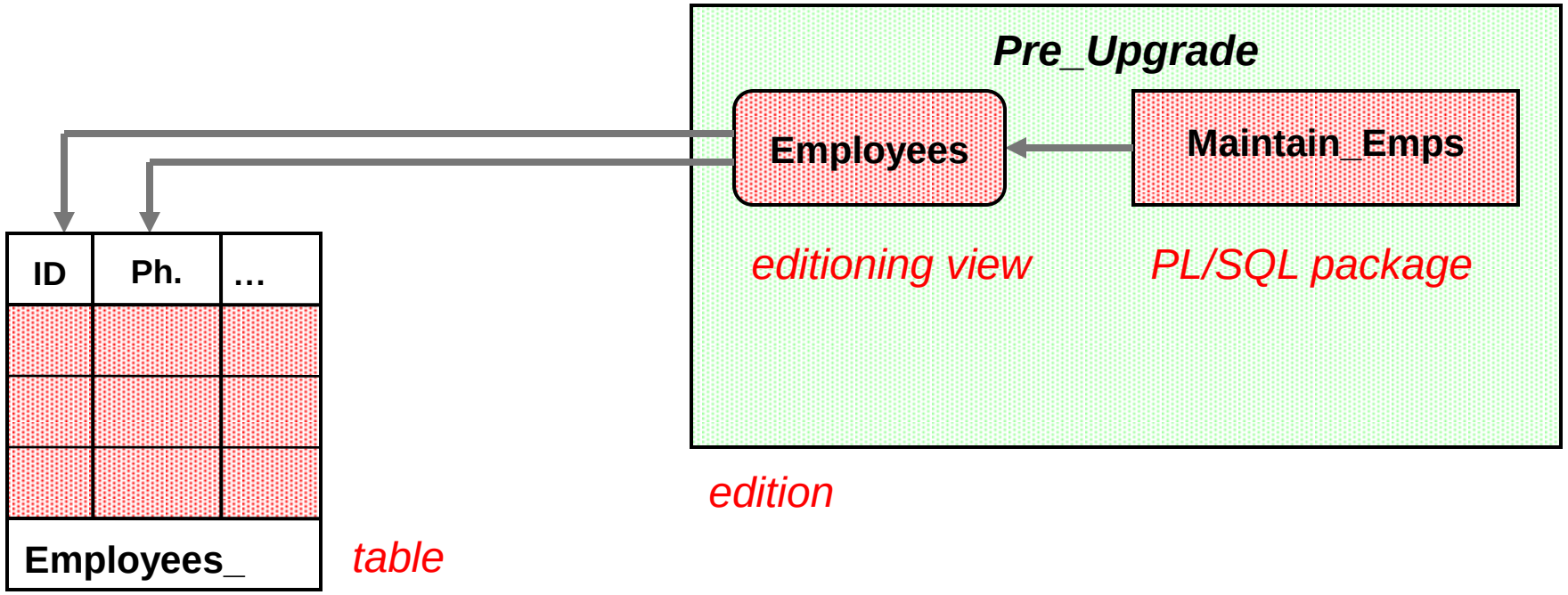
Readying the application for editions

- Of course,
  - All indexes on the original *Employees* remain valid but *User_Ind_Columns* now shows the new values for *Table_Name* and *Column_Name*
  - All constraints (foreign key and so on) on the original *Employees* remain in force for *Employees_*
- However,
  - Triggers don't fully "follow" the rename – and anyway that's not what you want
  - Rather, just drop the trigger and re-run the original create trigger statement to "move" the trigger onto the editioning view

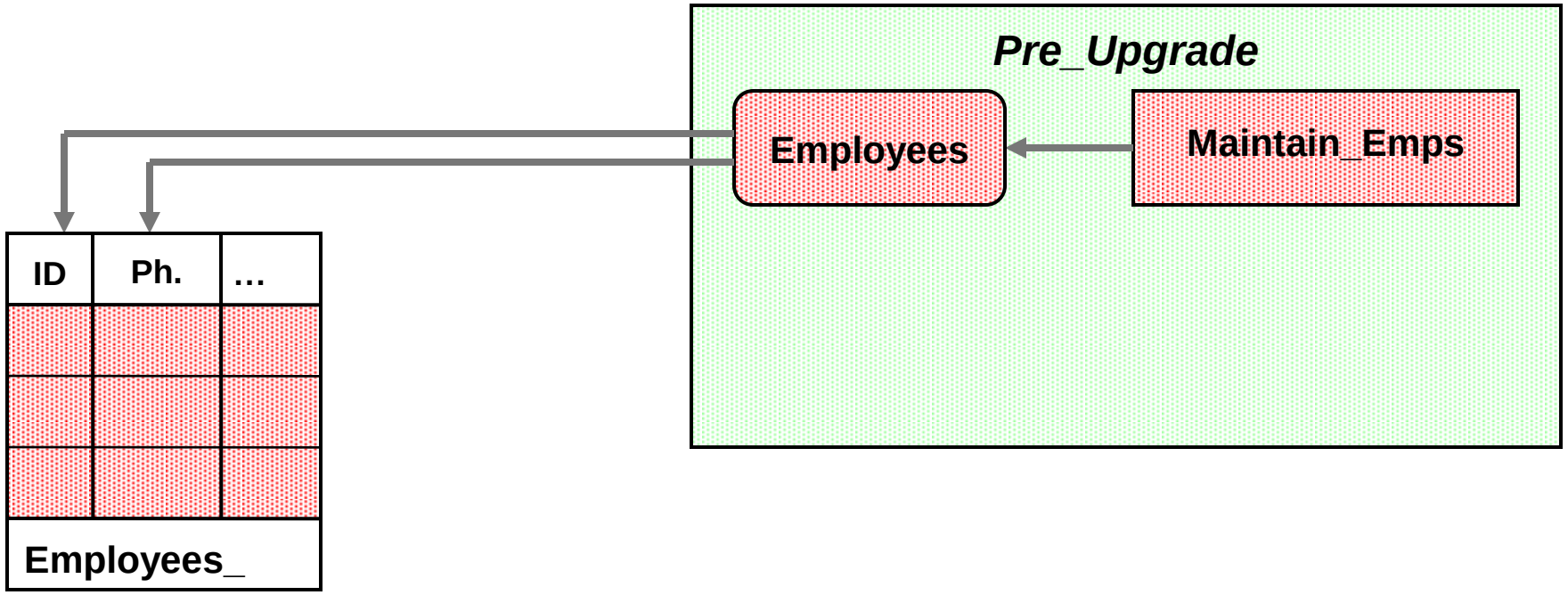
Case study – The edition-based redefinition exercise proper



Starting point.
Pre-upgrade app in normal use.



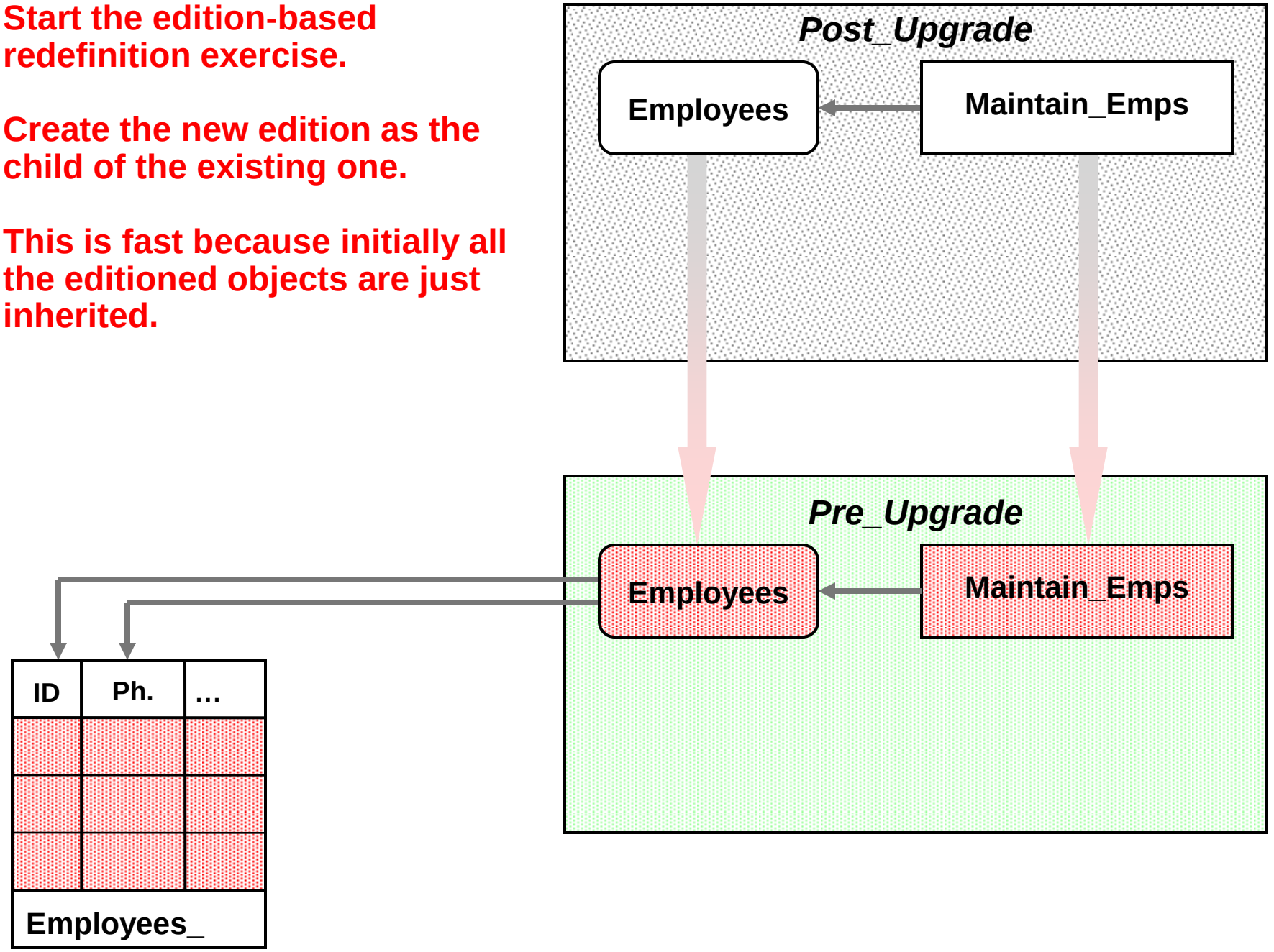
Starting point.
Pre-upgrade app in normal use.



Start the edition-based redefinition exercise.

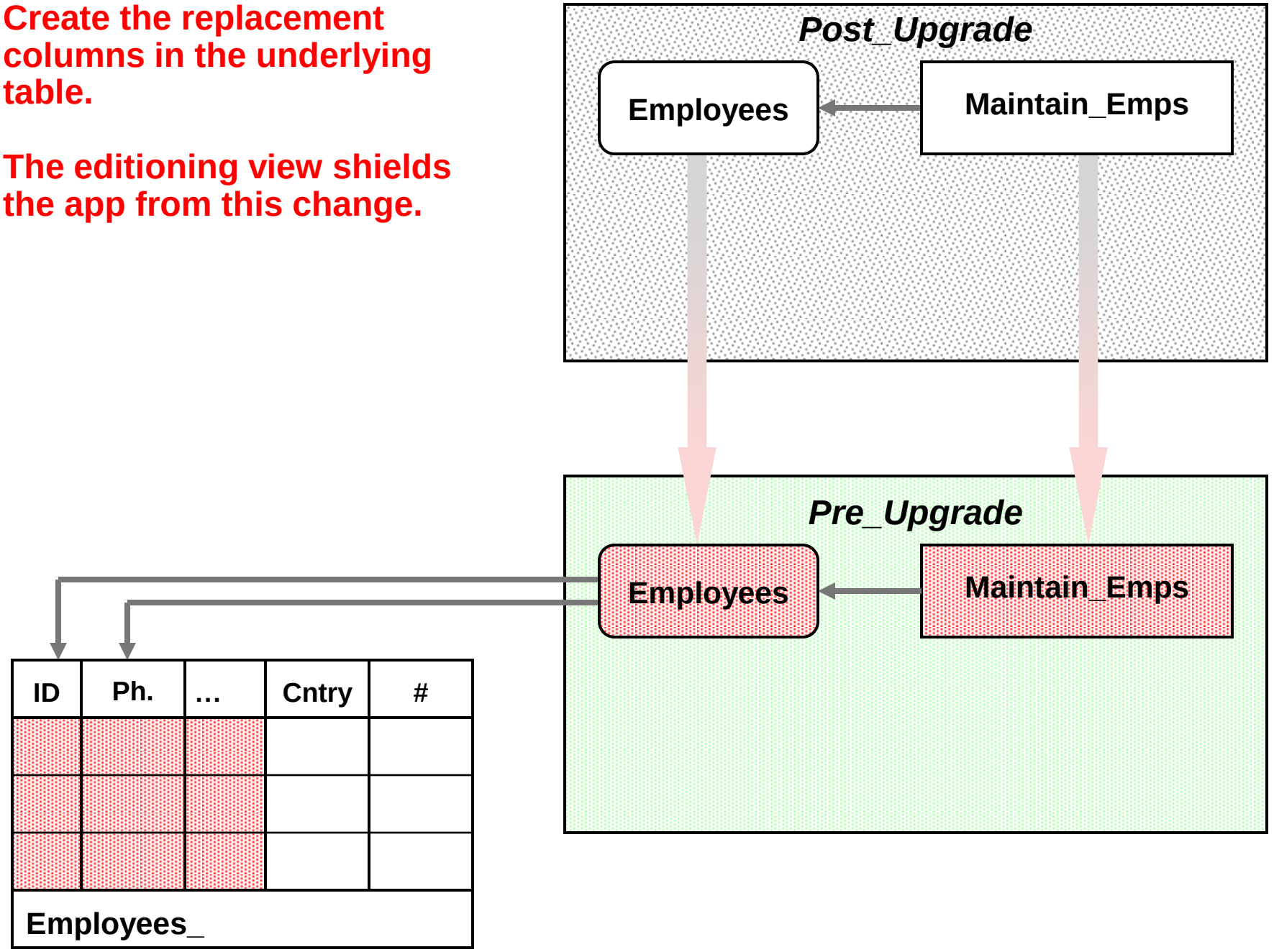
Create the new edition as the child of the existing one.

This is fast because initially all the editioned objects are just inherited.



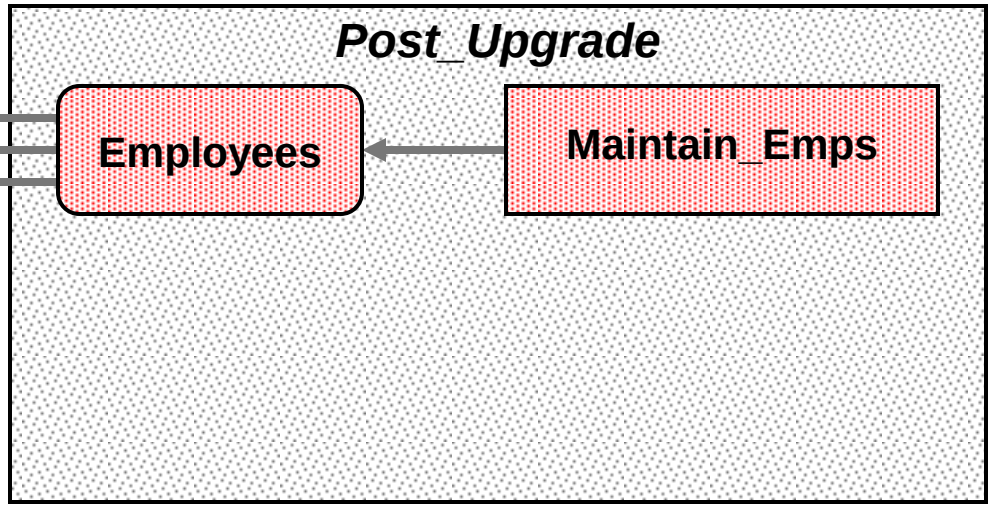
Create the replacement columns in the underlying table.

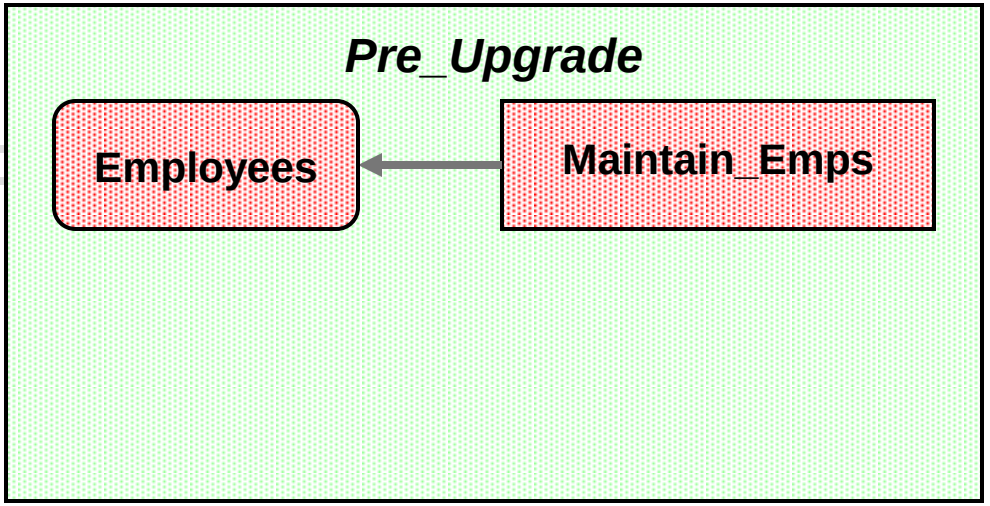
The editioning view shields the app from this change.

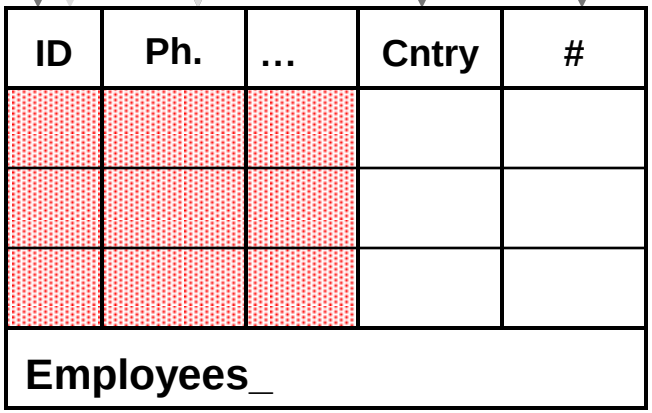


Change *Employees* to select the new columns.

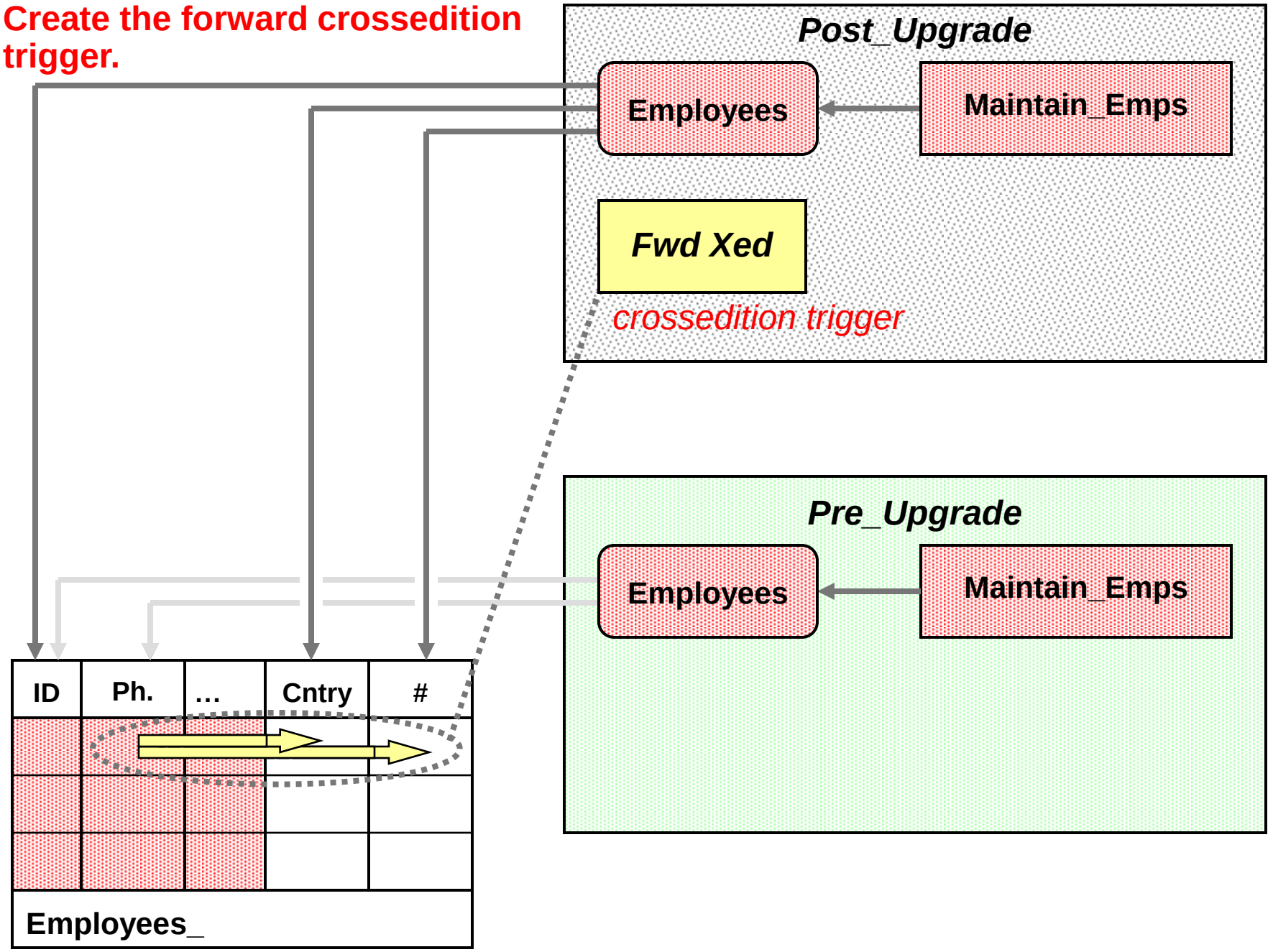
Change *Show_Employeess* to implement the new behavior.



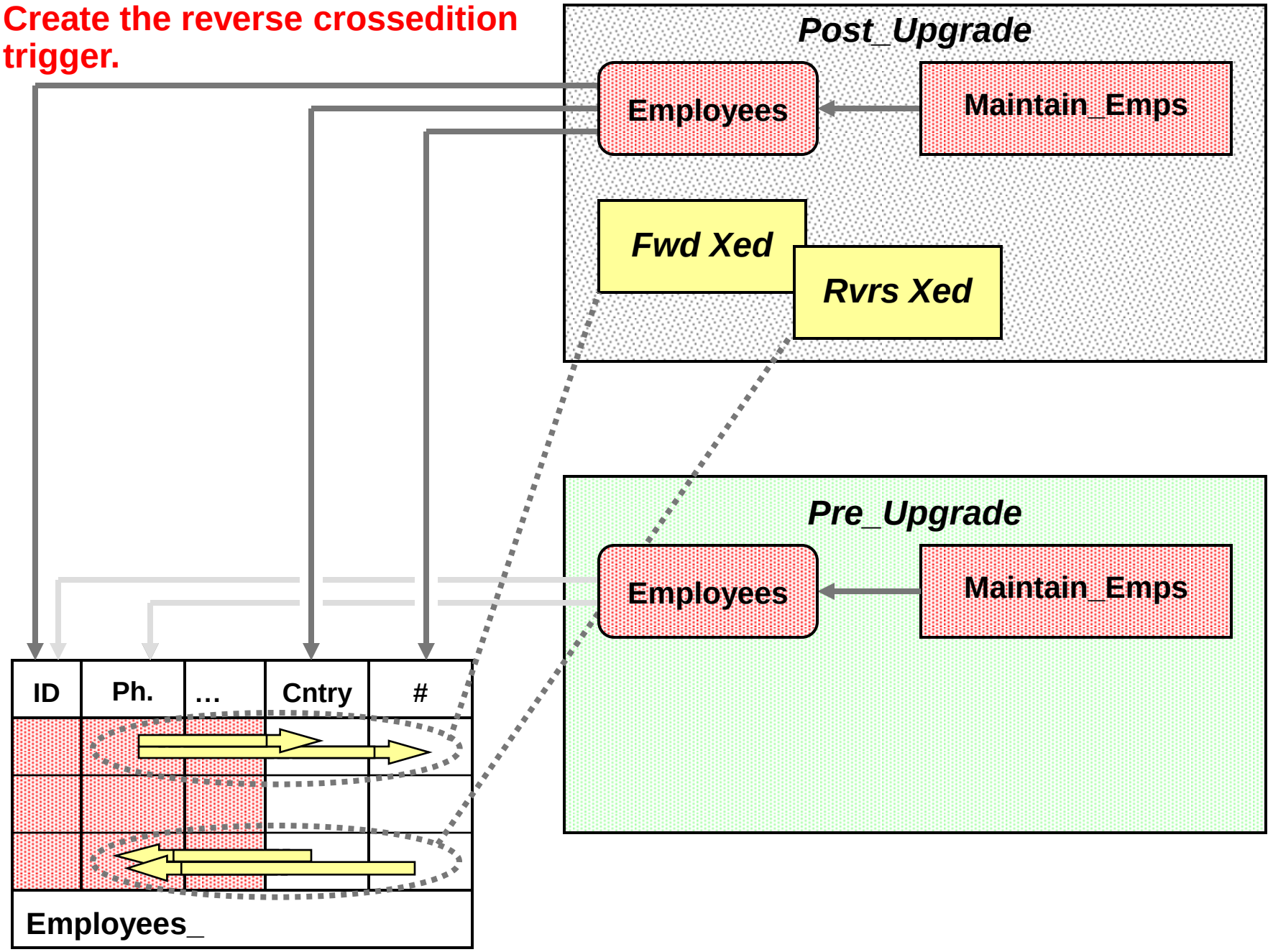




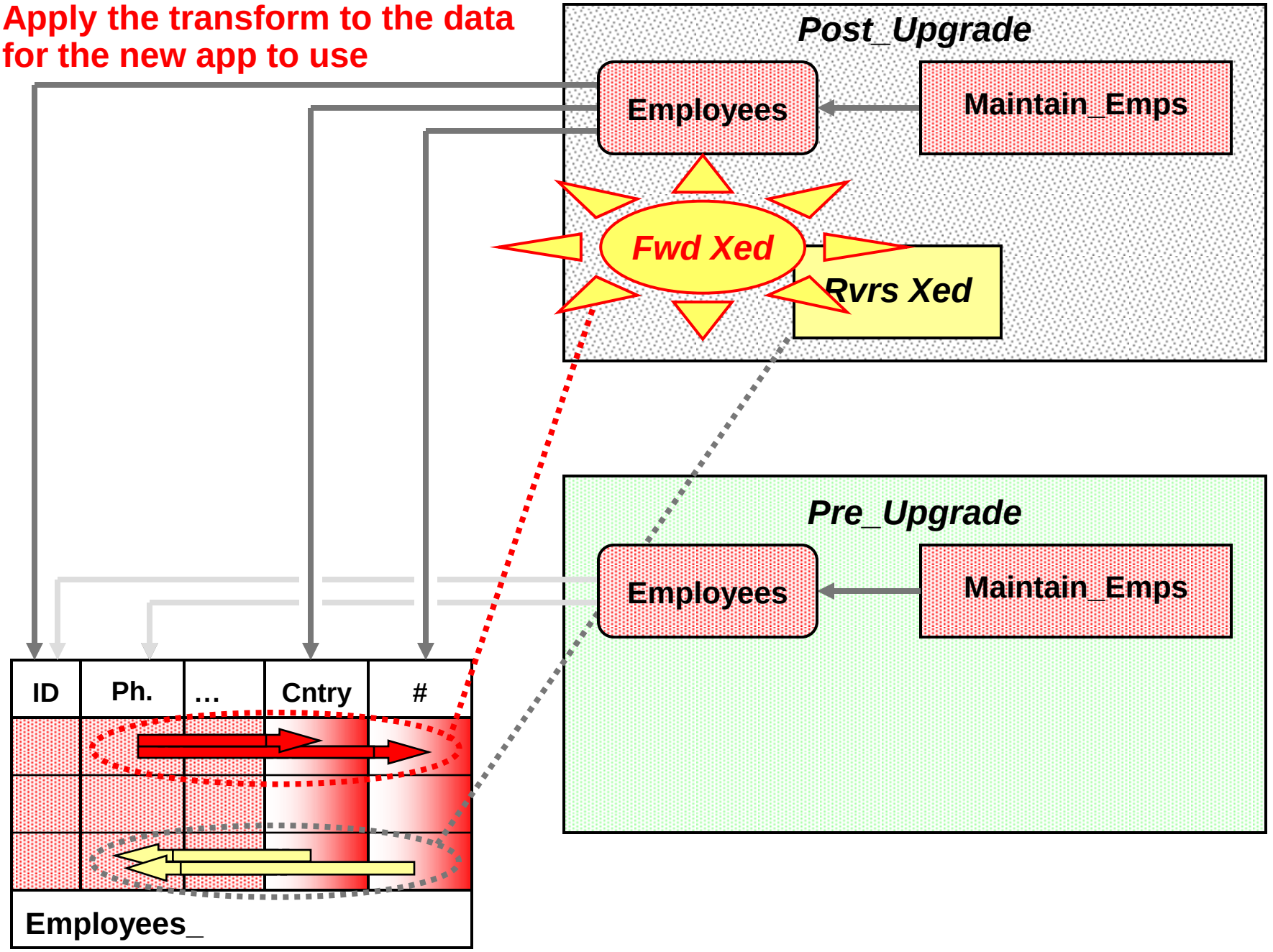
Create the forward crossedition trigger.



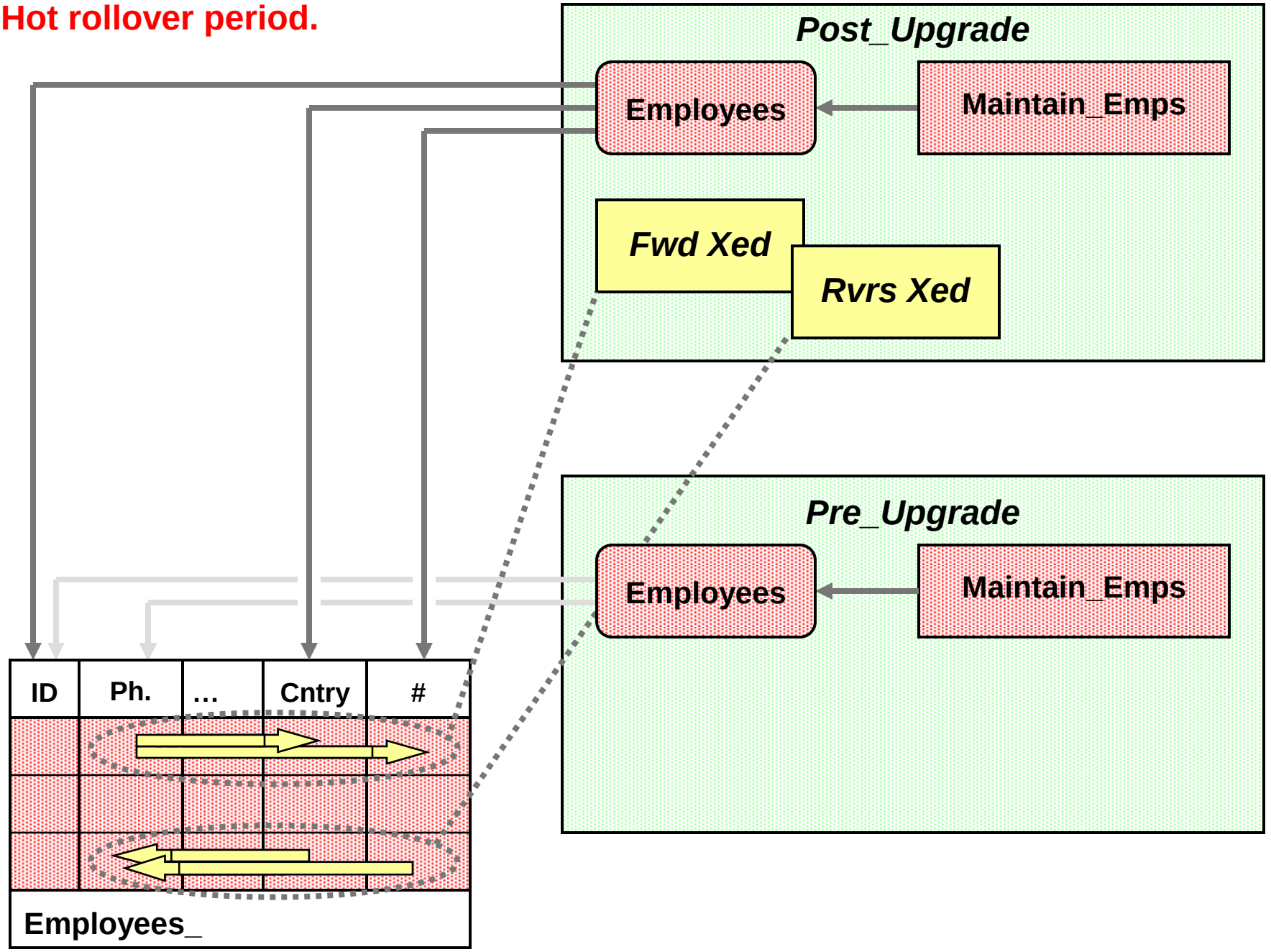
Create the reverse crossedition trigger.



Apply the transform to the data for the new app to use

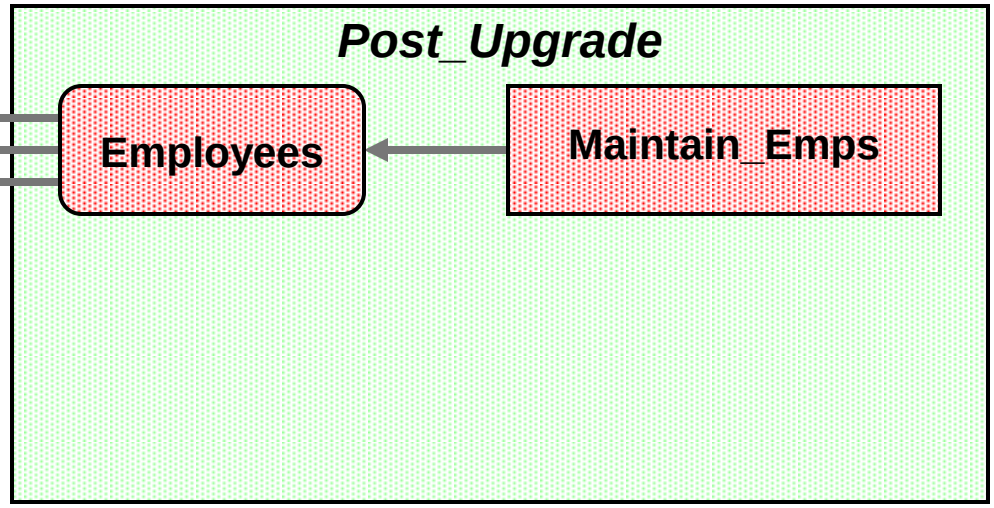


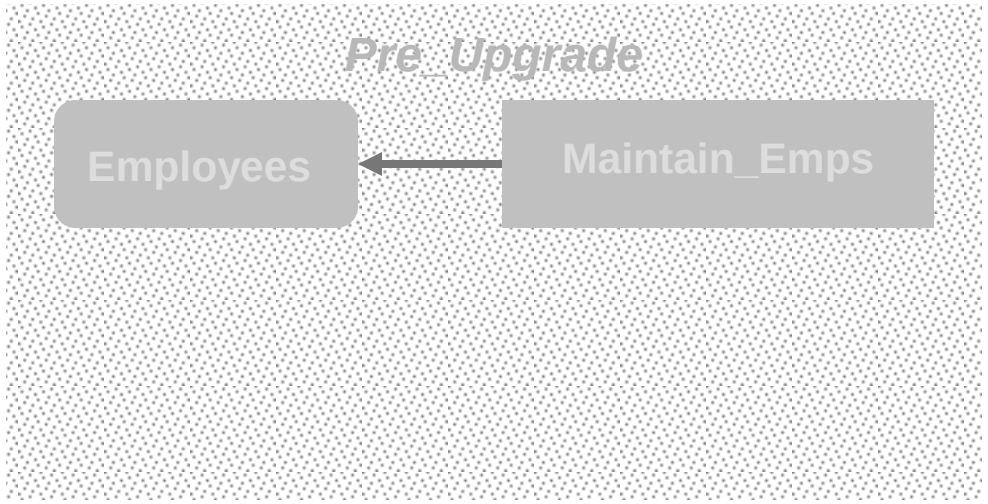
Hot rollover period.

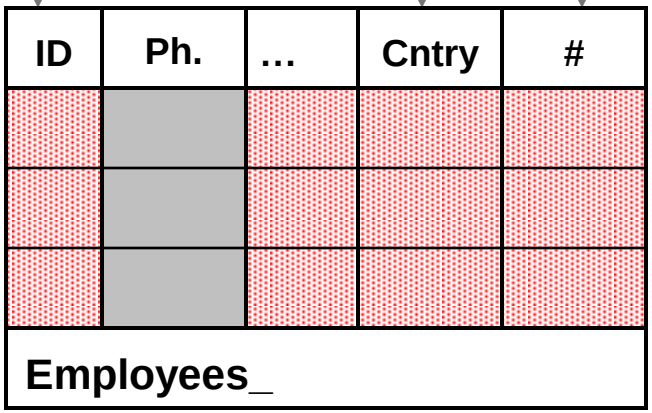


The *Pre_Upgrade* edition is retired.

The edition-based redefinition exercise is complete.







Case study – continued

Rolling back the upgrade

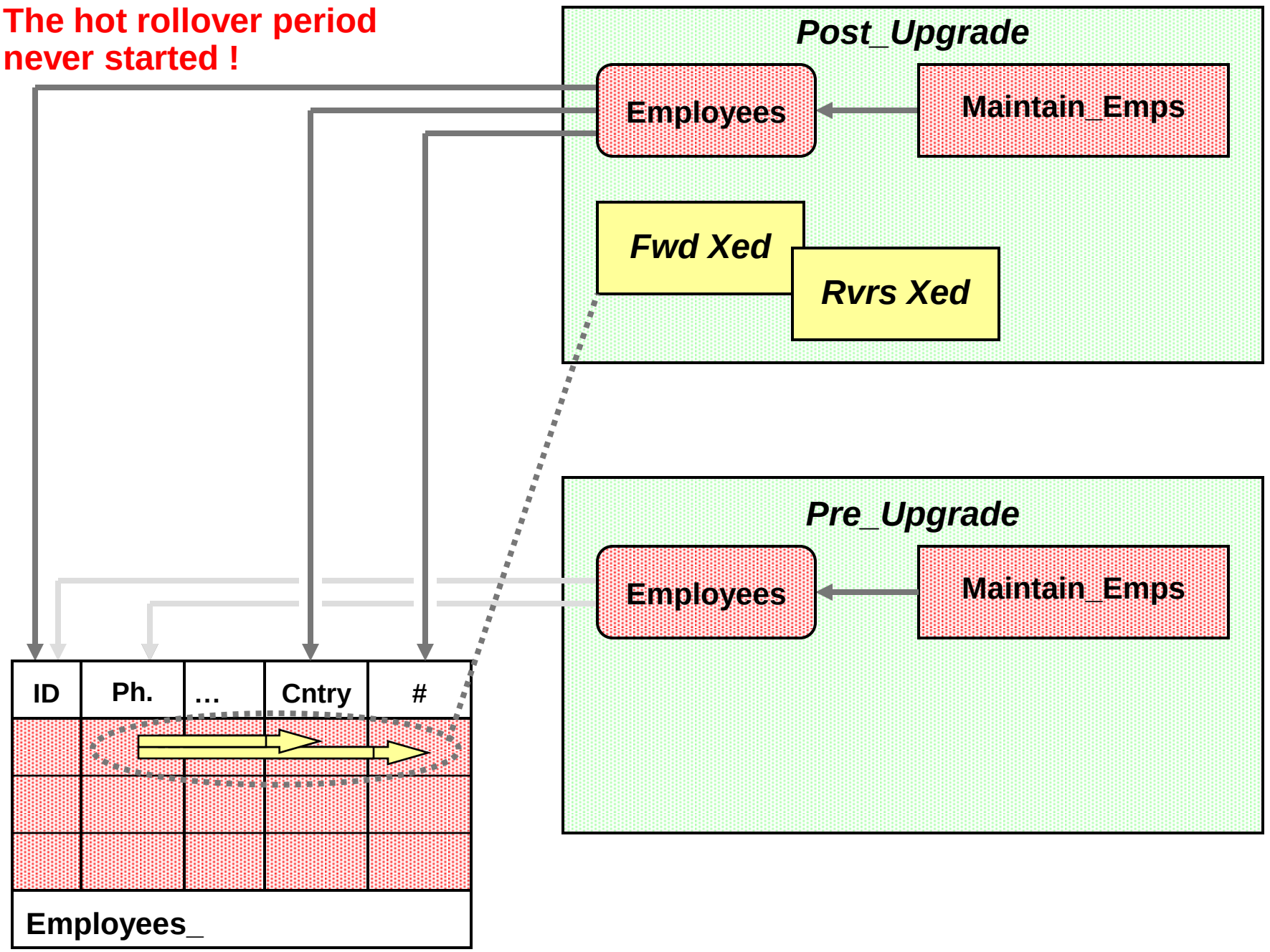


Rolling back an online app upgrade

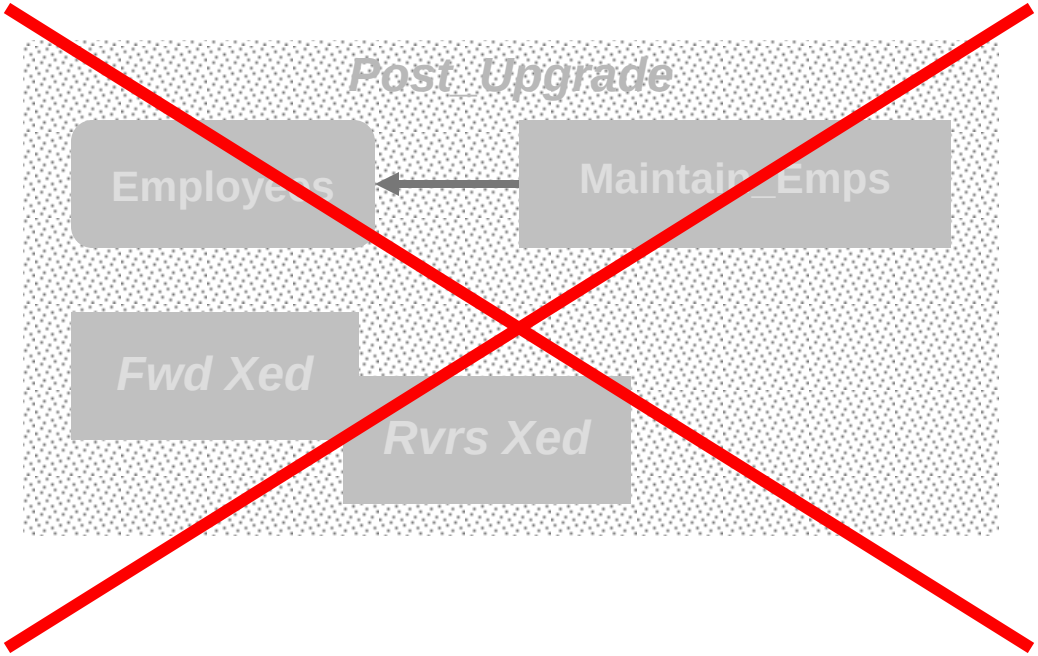
- Rolling back an application upgrade that's been installed classically is easy until you go live with the post-upgrade application
  - Presumably you took a backup at the start of the offline period and you just restore to that
- But once you go live with the post-upgrade application, you can't rollback to the pre-upgrade one
  - If you did this, you'd lose transactions made during the live use of the post-upgrade application
- It's just the same with online application upgrade
  - Your grace-period ends when you go live with the post-upgrade application

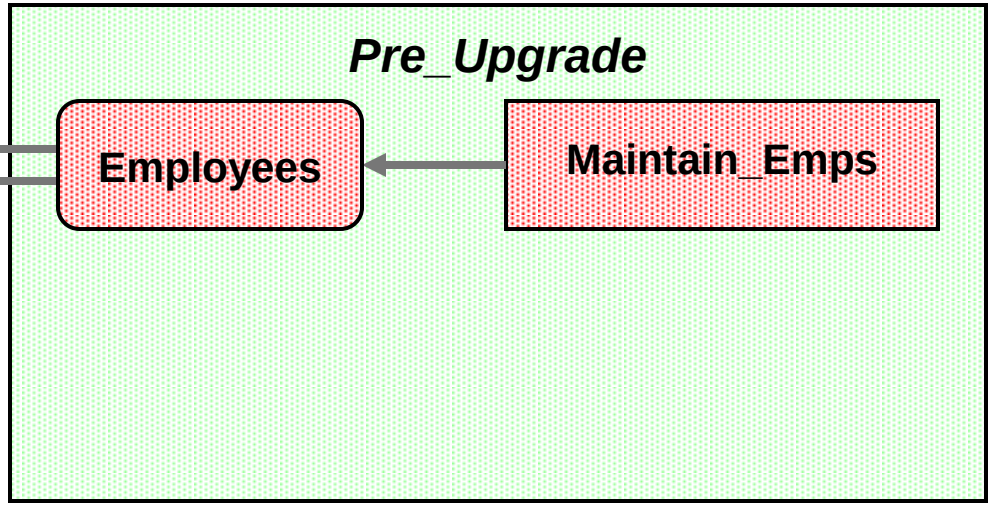
Rolling back an online app upgrade

- If you haven’t gone live with the post-upgrade application
  - Drop the *Post_Upgrade* edition (cascade)
  - Set any new replacement columns you created *unused*
  - At a convenient later time, recoup the space


The hot rollover period never started !
Post_Upgrade
Employees
Maintain_Emps
Fwd Xed
Rvrs Xed
Pre_Upgrade
Employees
Maintain_Emps
ID
Ph.
…
Cntry
#
Employees_

The pristine Pre_Upgrade is intact !





ID	Ph.	…	Cntry	#

Employees_

Agenda

- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- Explanation of the *edition*
- Explanation of the *editioning view*
- Explanation of the *crossedition trigger*
- Case study explained
- **EBR exercise vs offline upgrade: incremental extra effort**
- Conclusion / Q&A

EBR exercise vs offline upgrade:
incremental extra effort

Proportional occurrence		*editions*	*editioning views*	*forward crossedition triggers*	*reverse crossedition triggers*
?%	Change only editioned objects	✓			
?%	Make only additive table changes	✓	✓		
?%	Change only non-transaction tables	✓	✓		
?%	Change the structure of transaction tables non-additively	✓	✓	✓	
n/a	Support hot rollover	✓	✓	✓	✓

Agenda

- Scope of this presentation
- The challenge and the solution stated
- Case study stated
- Explanation of the *edition*
- Explanation of the *editioning view*
- Explanation of the *crossedition trigger*
- Case study explained
- EBR exercise vs offline upgrade: incremental extra effort
- **Conclusion / Q&A**

Edition-based redefinition

- 11.2 brings the *edition*, the *editioning view*, and the *crossedition trigger*
  - Code changes are installed in the privacy of a new *edition*
  - Data changes are made safely by writing only to new columns or new tables not seen by the old edition
    - An *editioning view* exposes a different projection of a table into each edition to allow each to see just its own columns
    - A *crossedition trigger* propagates data changes made by the old edition into the new edition's columns, or (in hot-rollover) vice-versa

Evolutionary capability improvements

- Some table DDLs that used to fail if another session had outstanding DML now always succeed
- Others, that cannot succeed while there's outstanding DML, are now governed by a timeout parameter
- Online index creation and rebuild now never cause other sessions to wait
- The dependency model is now fine-grained:
  e.g. adding a new column to a table, or a new subprogram to a package spec, no longer invalidates the dependants

Nota bene

- Online application upgrade is a high availability subgoal
- Traditionally, HA goals are met by features that the administrator can choose to use at the site of the deployed application
  - independently of the design of the application
  - without the knowledge of the application “vendor”
- The features for online application upgrade are used by the application “vendor”
  - when preparing the application for EBR
  - when implementing an EBR exercise
- Site administrators, of course, will need to understand the features

Online Application Upgrade

- Large, mission critical applications can now be continuously available while a patch or an upgrade is installed
- The pre-upgrade application and the post-upgrade application can be used at the same time
- End-user sessions therefore enjoy hot rollover
- The pre-upgrade application is retired only when no sessions any longer are using it

Edition-based redefinition

- Operates within the single database that implements the back end of the application
- Data synchronization is “ordinarily” transactional: potential conflicting changes, and violations of business rules, are detected and prevented before they can be committed
- Ordinary application code – and especially all application components implemented outside of the database – need no change in order to accommodate EBR

Next steps…

- Demo booth W-013 (under Oracle Database)
  Moscone West

- Read the edition-based redefinition chapter
  in the Oracle Database
  Advanced Application Developers' Guide, 11.2

- Read my whitepaper:
  published on the High Availability subpage
  under the Database page on OTN

- Internet search for *edition-based redefinition*

Q&A

