

# Beyond the Outer Limits with User Extensions

*IOUG-A Live! 2000*

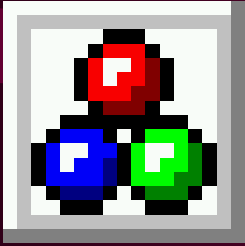
Douglas Scherer, Core Paradigm  
Peter Koletzke, Millennium Vision  
Corporation

# What is Oracle Designer User Extensibility?

- Also called User Extensions
- Lets you add to the native repository elements
- Lets you add properties to existing elements
- Works on the meta-model objects

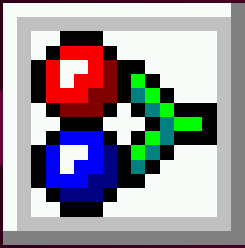
# The Meta-Models

- Repository Meta-Model
  - Links one element to another
  - For example: Table Definitions and Column Definitions
- Types Meta-Model
  - How the element *types* are linked
  - Actually the meta-model of the Repository Meta-Model
  - Contains 3 types: Element, Association, Text



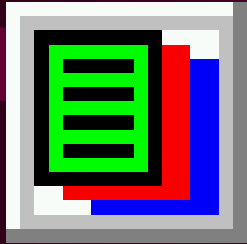
# Element Types

- Basically, the main objects you define in the repository
  - For example: Functions, Modules, Tables, Columns
- High level nodes in Repository Object Navigator
  - For example: Tables
- Also some lower level nodes
  - For example: Columns



# Association Types

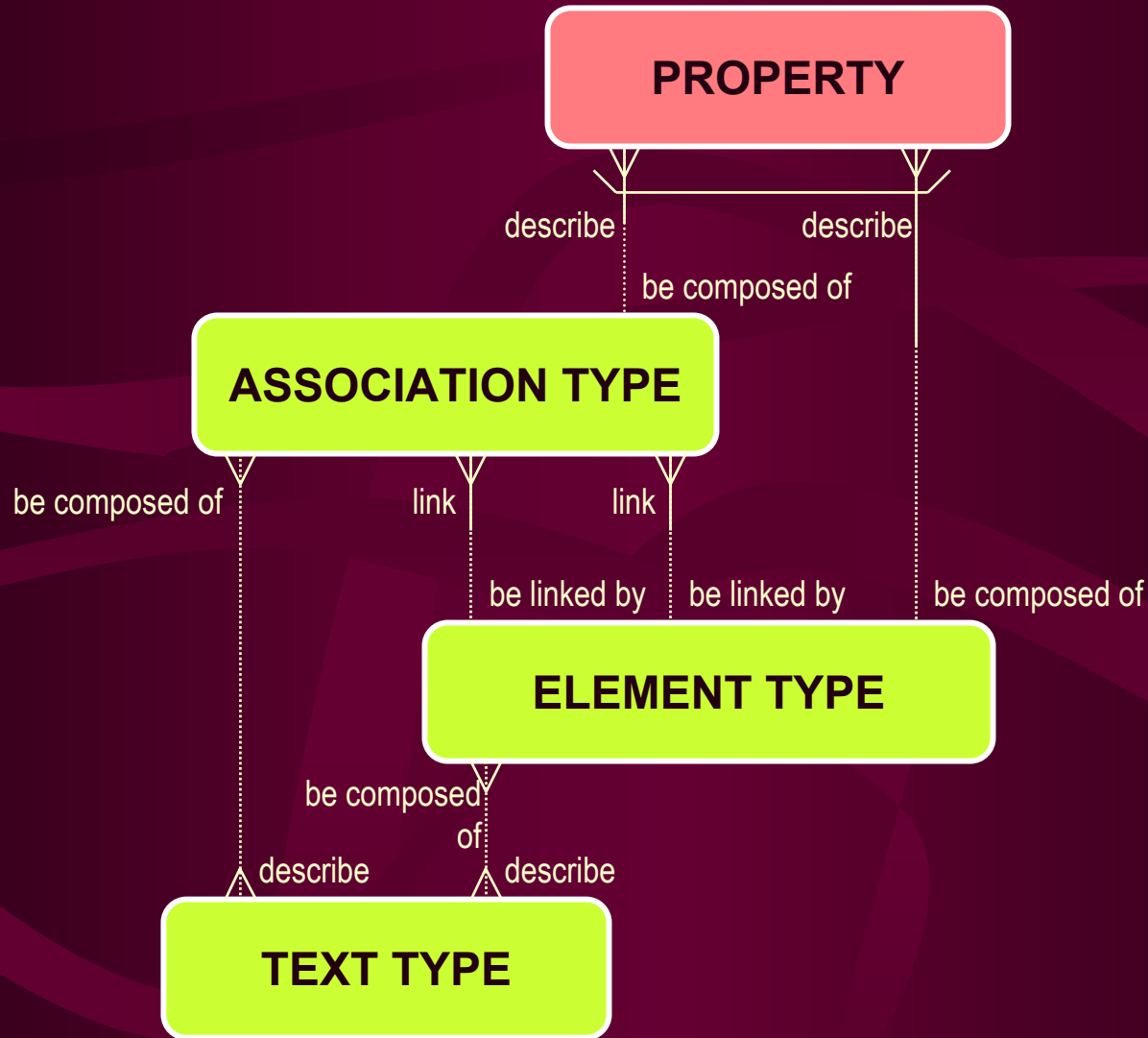
- The links between elements
- Appear under the element nodes that they link
- For example: Function Entity Usages, Module Table Usages, Module Column Usages
- Usually resolves a many-to-many relationship between elements
- Has properties itself - e.g., CRUD



# Text Types

- Text information about an element or association
- Shows in RON as a property but is really a separate thing
- For example: Description, Notes, Select Text
- Has a many-to-many relationship with elements and associations

# The Types Meta-Model



# What can you extend?

- 20 Properties in existing (System Defined) elements
  - User Extension properties (Usrx0 - 19)
  - Can query and insert into these with the API
- 500 Element types and Association Types (E0 - 499) and (A0 - 499)
- 20 Properties per each type
- Unlimited number of Text types and Usages

Demo

# Where Does UE fit in Oracle Designer?

- Hand-in-hand with API
- Oracle Designer has only 3 tools to help
  - Repository Object Navigator
  - Matrix Diagrammer
  - Design Editor
- You need the API to manage User Extensions
  - If you do UE, you have to do API

# Some Examples

- Legacy Reports
  - Status flag to indicate how they will be used
- Requirements element
  - Associations to Function, Module, Entity, Table
  - Can't link to Columns though
- Extra user documentation properties
  - Describe the element further

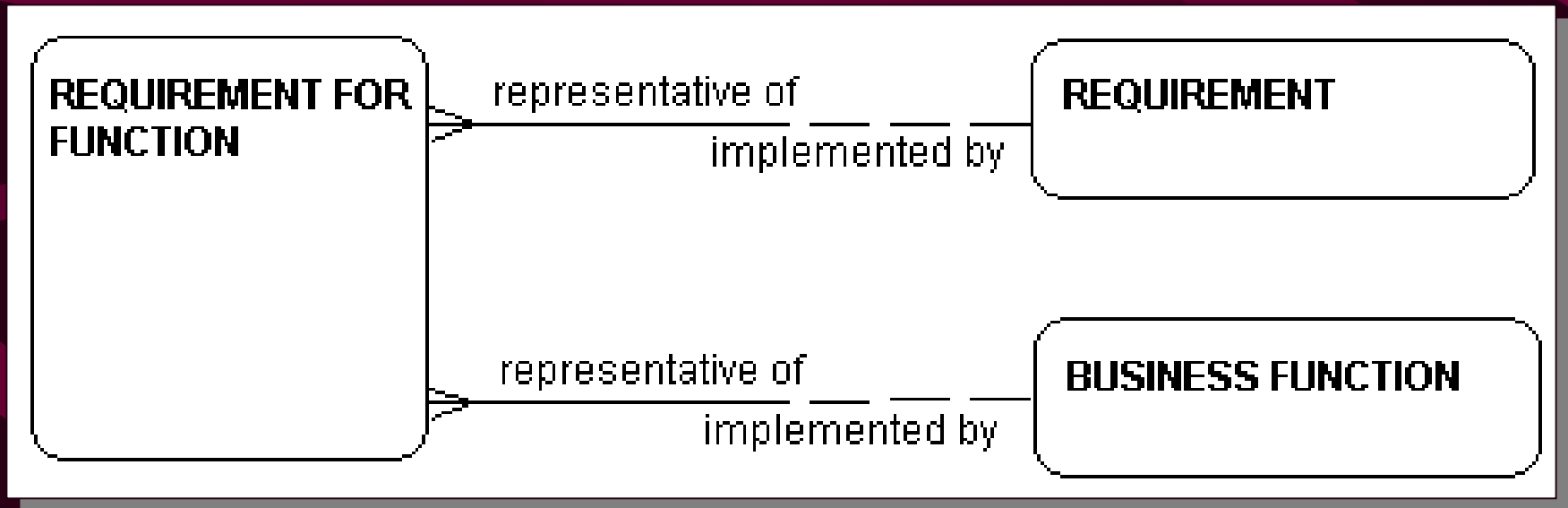
Demo

# How to Create User Extensions

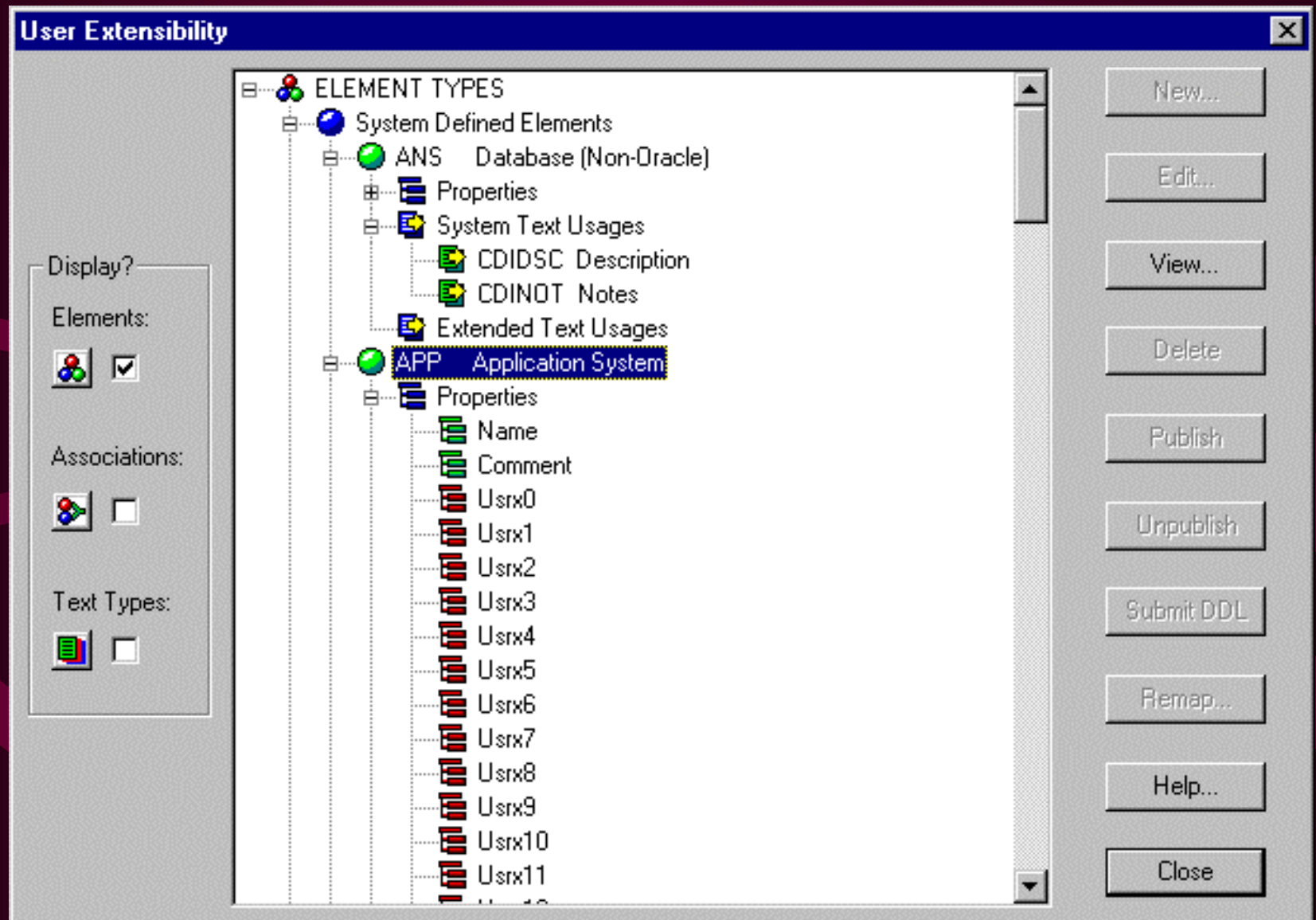
1. Plan
2. Define in Repository Administration Utility
3. Publish
4. Reconcile grants

# 1. Plan

- Draw an ERD if necessary
- Be sure of cardinality and optionality



## 2. Define in RAU



# Filling Out the Definition

- Elements require mostly descriptions
- Associations need cardinality to both elements
- Properties require datatype, size
  - Can't do valid values
- Text types need link to element(s)

# An Association Type Details Definition Window

Association Type Details

Short Name

Name

Plural Name

Display Name

Display Plural Name

Details

OWNING ELEMENT: Element Type

Cardinality

TO: Element Type

Cardinality

Package Name

View Name

User element type

Has been extended

Published

OK Cancel Help

# How To Extend Properties

- Same process, except
  - Pick existing element or association
  - Pick existent Usrx property
  - Edit the property definition

# A Property Details Window

**Property Details** [X]

Element Type: DST DATASTORE

Property Name: USER\_DEFINED\_PROPERTY\_0

Displayed Name: Existing System

**Display details**

Datatype: CHAR

Length: 5

Precision: 0

Scale: 0

Sequence: 5

Updateable

**Other details**

Case mode: UpperCase

Mandatory

User property

Has been extended

Published

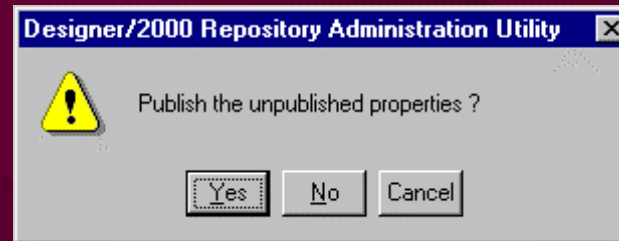
Base table: SDD\_ELEMENTS

Base column: EL\_USRX0

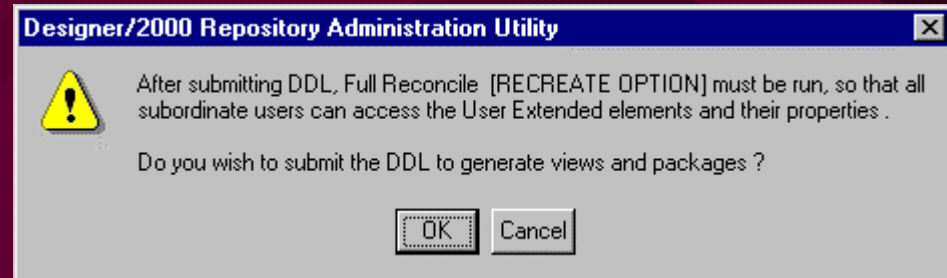
OK Cancel Help

# 3. Publish

- You can wait on this if you want
- New types will prompt for publishing properties



- Prompt for running the DDL scripts created



- You can unpublish later

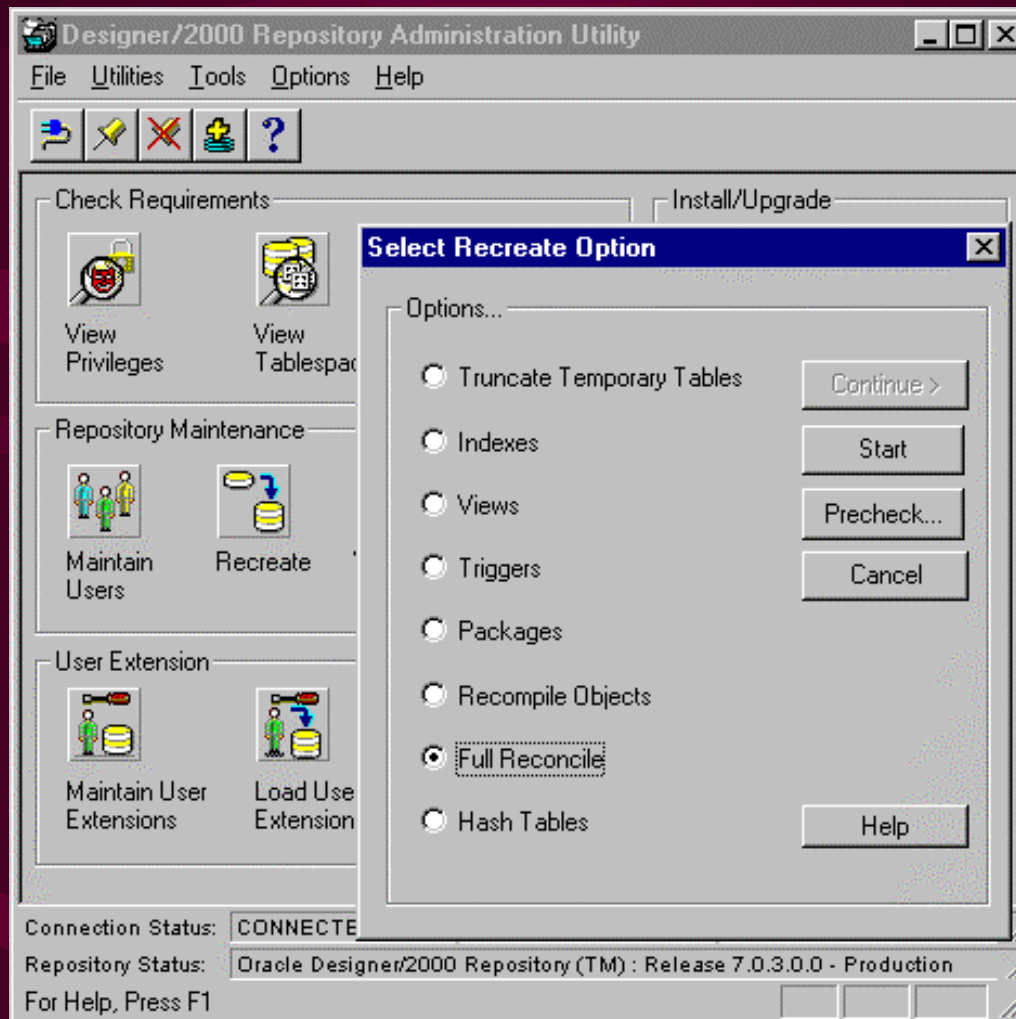
# What Publishing Creates

- New type
  - The type itself
  - API view (*CI\_Plural Name*)
  - API package (*CIOName*)
- New property
  - Additional, user-named property in existing type

## 4. Reconcile Grants

- Creates grants and synonyms for repository users
- Click on Recreate button in RAU
- Choose Full Recompile
- Check it out in RON

# Windows Used for Full Reconcile



# Oops, I made a mistake!

- Must unpublish an extension before deleting it from the repository
- Cannot unpublish if
  - extension has been used to add elements to the repository
  - Other association types reference the extension
  - You have not submitted the API creation DDL
- Select the item then click Unpublish

# Access the User Extension

- May need to first reopen the tool
- RON
  - All extensions are available
  - Select the User Extensions check box in View->Include Navigator Groups
- Matrix Diagrammer
  - Only some extensions are available
  - Element types will be seen on either side of extended associations
- Design Editor
  - Property extensions only are available

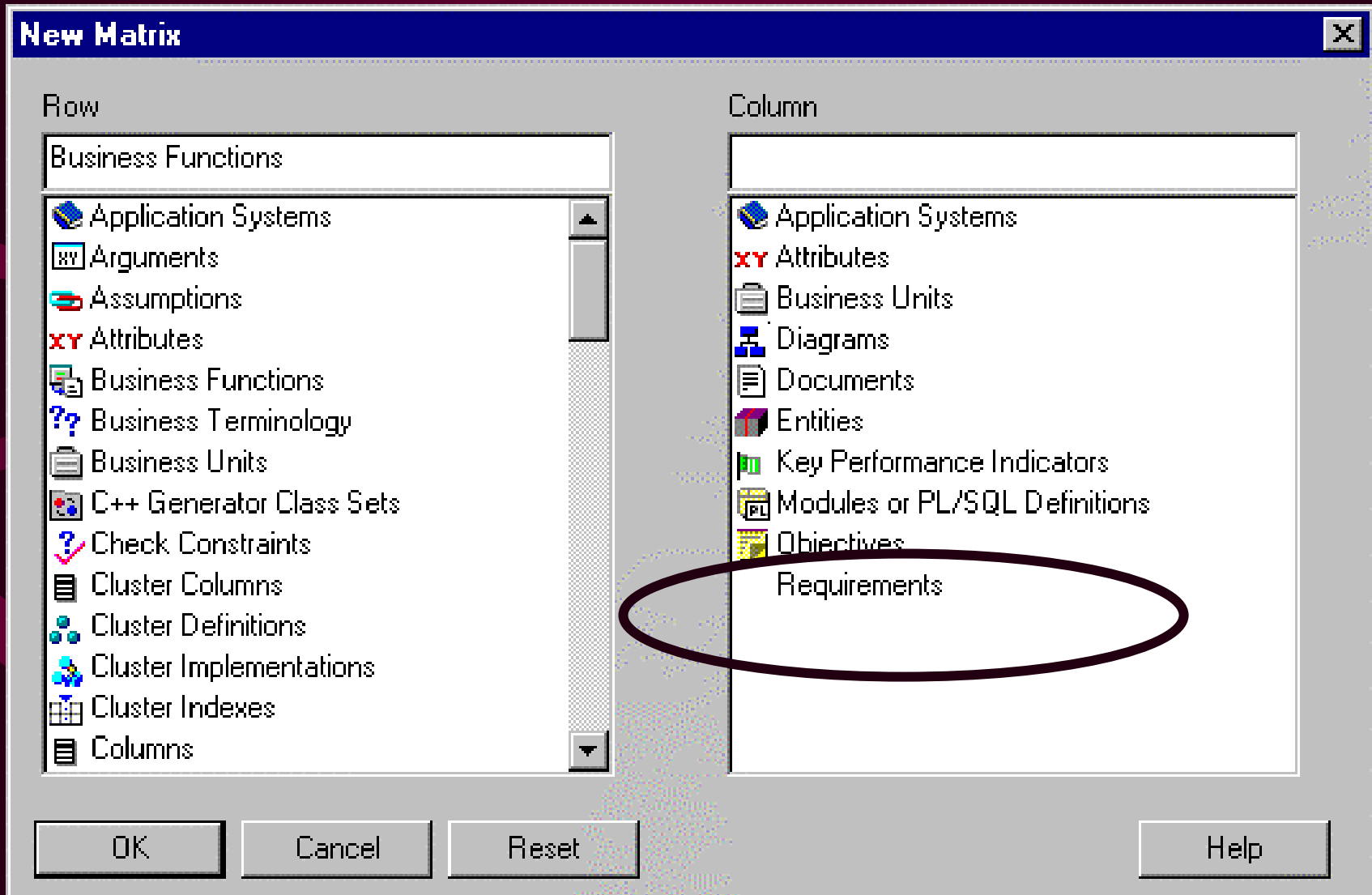
# A User Extension in RON

The screenshot displays the RON (Requirements Organizer) software interface. On the left, the 'Navigator' window shows a hierarchical tree structure for a project named 'CTA(1)'. The tree includes various modeling categories such as Reference Data Definition, Enterprise Modeling, Entity/Relationship Modeling, Dataflow Modeling, Function Event Modeling, Type Modeling, C++ Object Generation, Server Model Definition, File / Record Definition, Module Design, Database and Network Design, Sets, and User Extensions. Under 'User Extensions', there is a 'Requirements' folder containing 'END OF MONTH REPORT', 'User Extended Associations', 'Business Functions (User Reqs to Functions)', and 'Usages'. The 'Business Functions (User Reqs to Functions)' folder is expanded, showing a sub-entry 'CTAENROLL' which is highlighted with a red oval. Below this, there are folders for 'Documents', 'Included in Sets', and 'Included in Applications', with 'CTA(1)' listed under the last one.

On the right, the 'User Reqs to Functions Properties' window is open, showing a table with the following data:

Application Owner	CTA
Requirements	END OF MONTH REPORT
UE	
Comment	

# A User Extension in Matrix Diagrammer



# A User Extension in Design Editor

The screenshot displays the Design Editor interface for a table definition. The main window is titled "Design Editor - CTA[2] - CTA: Table Definition Properties". The interface is divided into several panes:

- Server Model - Navigator:** Shows a tree view of the database structure. The "CTA[2]" folder is expanded to show "Relational Table Definitions", which includes a list of tables: COMMUTERS, COURSES, DEPT, DEPT\_EMP, EMP, ENROLLMENTS, GRADE\_BASES, GRADE\_CONVERSIONS, GRADE\_TYPES, INSTRUCTORS, REGIONS, RESIDENTS, SECTIONS, and STUDENTS. The "COMMUTERS" table is selected.
- CTA: Table Definition Properties:** This pane shows the configuration for the selected table. It includes sections for:
  - Volumes:** Start Rows, End Rows.
  - Storage:** Index-organized? (No), Cluster.
  - Datawarehouse:** Datawarehouse Type.
  - Documentation:** Comment, Description, Notes, User/Help Text.
  - Visual Basic / C++:** Insert Code, Update Code, Delete Code, Lock Code.
  - UE (User Extension):** Growth Rate in Rows per Year (20000).

A red oval highlights the "UE" section in the properties pane, specifically the "Growth Rate in Rows per Year" field, which is set to 20000. This field is a user extension not typically found in standard database table definitions.

COMMUTERS

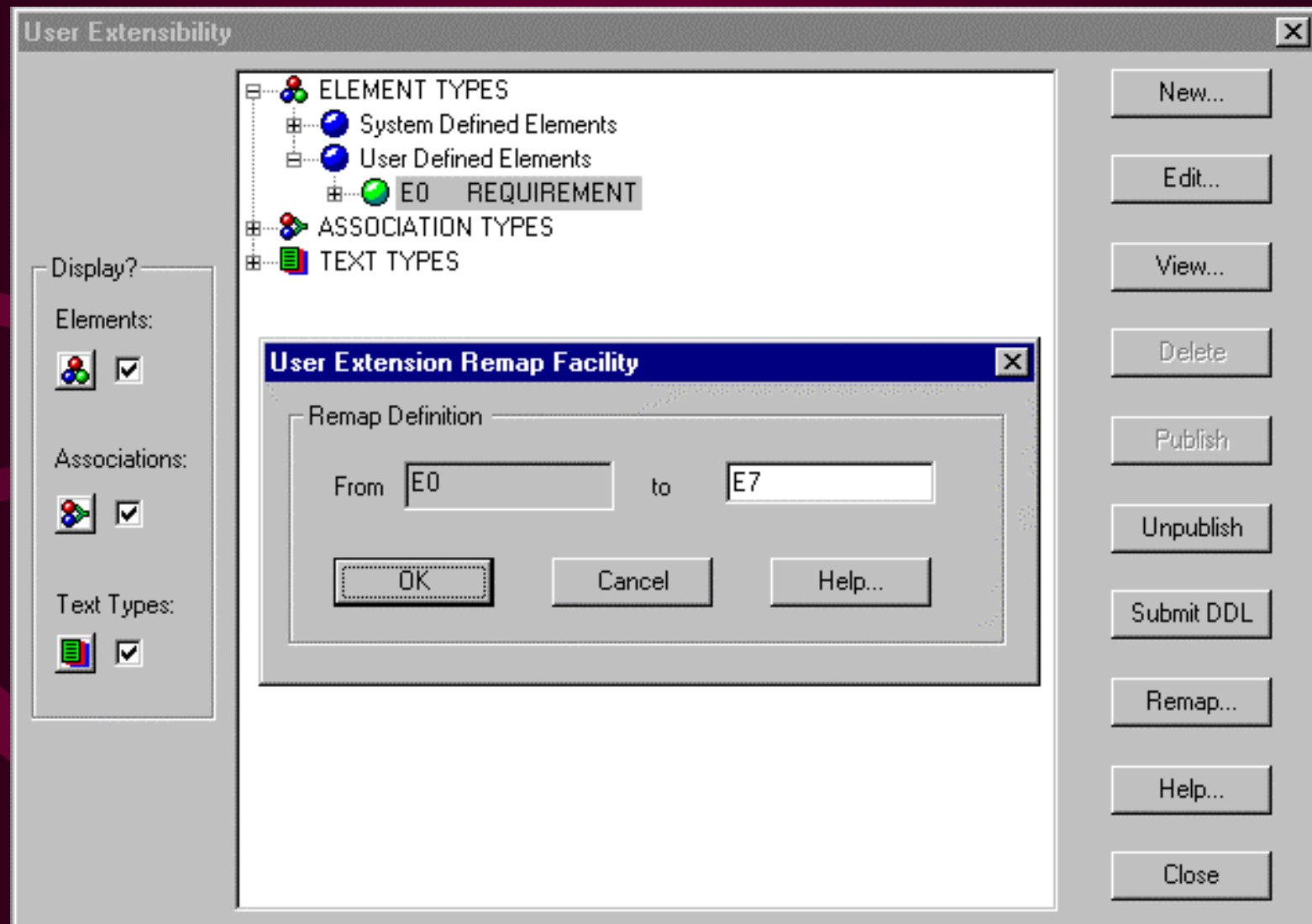


**Demo**

# Remapping the User Extension

- Changes element name (e.g. from E0 to E7)
- Used when importing same named extension from another repository
- Steps
  - Select item to be remapped
  - Click Remap button
  - Do not need to republish republish or submit DDL

# The Dialog for Remapping



# Repository Reports

- Text Definition - `cduetxt`
  - Definitions and usages of text types
- Detailed `<Element>` Definition - `cdueela`
  - Properties of a specified element's use
- Element Type Definition - `cduedef`
  - Definition for a given element type including its associations
- All Text Held against an Element - `cdeltxt`
  - Details about the element
- or ... Roll your own

# Exporting Concerns

- Application ↙ Archive in RON
- Target repository must have same extensions too
  - Use RAU to export these
- You could still have problems
  - If both repositories have an E4 type (for example) used for different purposes

# Customization Through the API

- Can publish your extension through the API
- Can change or insert definitions of pre-published extensions
- API packages have names like
  - `cioue_element_type`
  - `cioue_association_type`
- See online API help documentation topic: “User Extensibility Support”

# Caveats

- Don't create mandatory properties for diagrammed elements
  - Diagrammer will not be able to insert or update elements
  - e.g., table definition
- Repository-wide
  - Not just one app system
- Only works on Primary Access Controlled units (PAC)
  - E.g., Tables
  - Not Columns (SAC)

# Conclusions

- UE is quite powerful
  - Makes up for some of Oracle Designer's limitations and bugs
  - Coupled with the API, you can do anything
- It is a simple procedure, but:
  - There are some concerns, so be aware
  - Do it only if you really have a task that needs it

# Author Contacts

- **Please fill out the evaluations**
- **Douglas Scherer**
  - <http://www.coreparadigm.com>
  - [dscherer@coreparadigm.com](mailto:dscherer@coreparadigm.com)
- **Peter Koletzke**
  - <http://www.mvsn.com>
  - [peter\\_koletzke@compuserve.com](mailto:peter_koletzke@compuserve.com)
- ***Oracle Designer Handbook, 2nd Ed***  
**Osborne McGraw-Hill, Oracle Press,**  
**ISBN 0-07-882417-6, co-authored by**  
**Dr. Paul Dorsey with lots of help**  
**from Douglas.**