

Oracle Designer API: Multiline Text and Other Secrets *IOUW '99 - Paper 705*



Douglas Scherer, Core Paradigm
Peter Koletzke, Millennia Vision Corporation

API Overview

What is the API?

- Used by Oracle Designer tools to work with the repository
- The API includes:
 - Views of tables representing actual objects (entities and attributes)
 - PL/SQL packages
- Documented method to insert or modify data within the repository
- Opens the repository to allow developers to write extensions to tools and safely manipulate virtually any element in the repository
- **If you aren't using the API, you are only using 50% of Oracle Designer**

When do you need the API?

- Some activities cannot be done with Oracle Designer tools and repository elements
- Specific system information may have no default place in repository
- User Extensibility features (such as RON) may not be fast or easy enough
- API packages provide the only method for writing to the repository

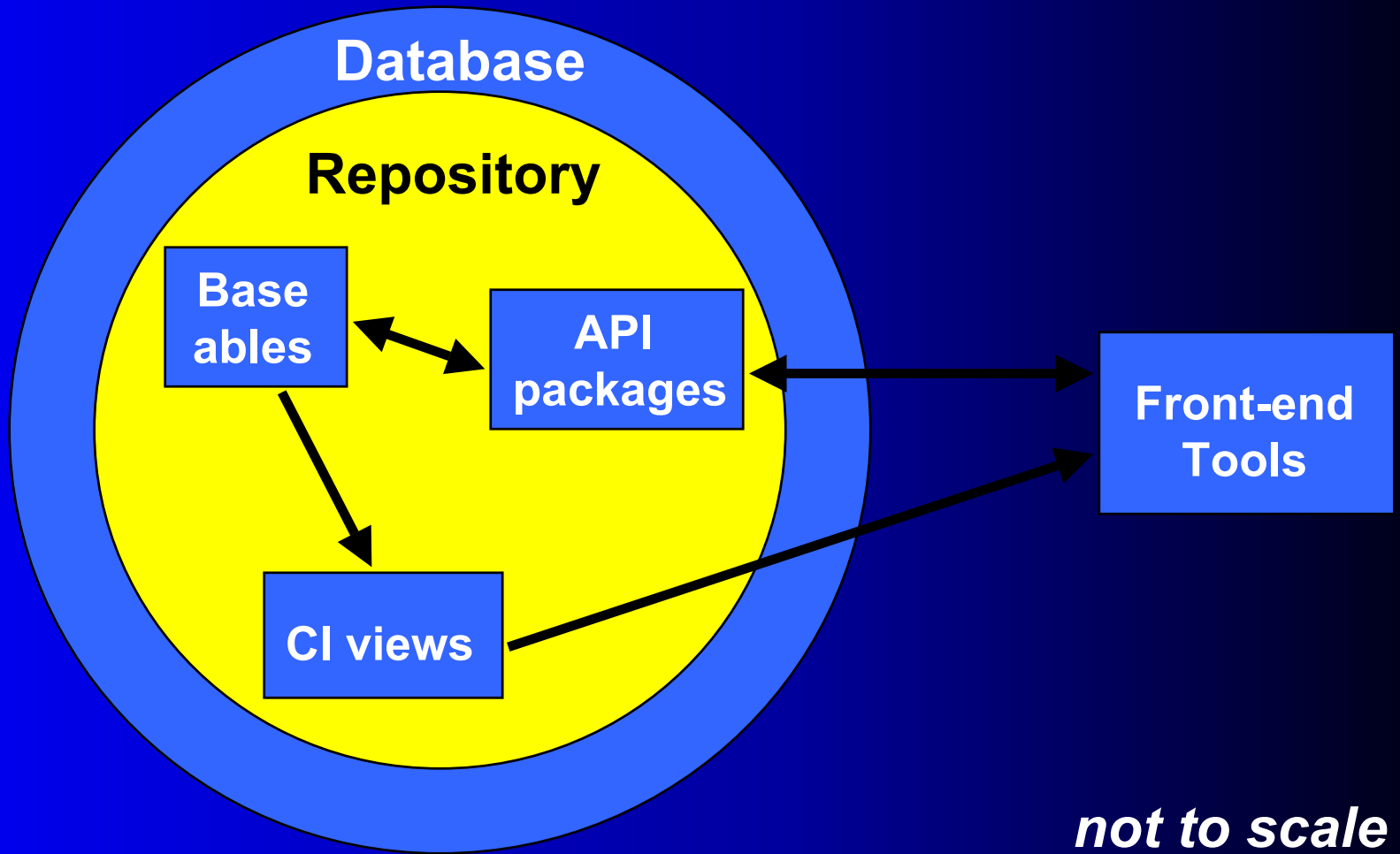
What can you do with the API?

- Input and output information from base repository elements
- Examine definitions created in your application systems
- Change the contents of tables safely outside Oracle Designer front-end
- Supplement Oracle Designer diagrammers with your own front-end programs or code
- Input and output information from User Extensibility elements

Getting Started with the API

- Identify the views and packages you need
 - You can look in RON to find the names
 - API views have names starting with CI_plural name of the element
- Obtain detailed information on those views and packages in Repository Model package diagrams.
- Create the API code
 - If just querying views, SQL SELECT script will work
 - Use API packages
 - To perform insert/update/delete operations - write a PL/SQL procedure or package

API in Context



API Naming Conventions

- Base table names
 - Prefixed with SDD_ and CI_
 - SDD_ELEMENTS
 - SDD_STRUCTURE_ELEMENTS
 - CDI_TEXT
- View names
 - Prefixed with CI_, end with the element's plural name
 - CI_ENTITIES
 - CI_TABLE_DEFINITIONS
- Package names
 - Start with CIO, end with the element's singular name
 - CIOTABLE_DEFINITION
 - CDAPI
 - Used for API transaction management

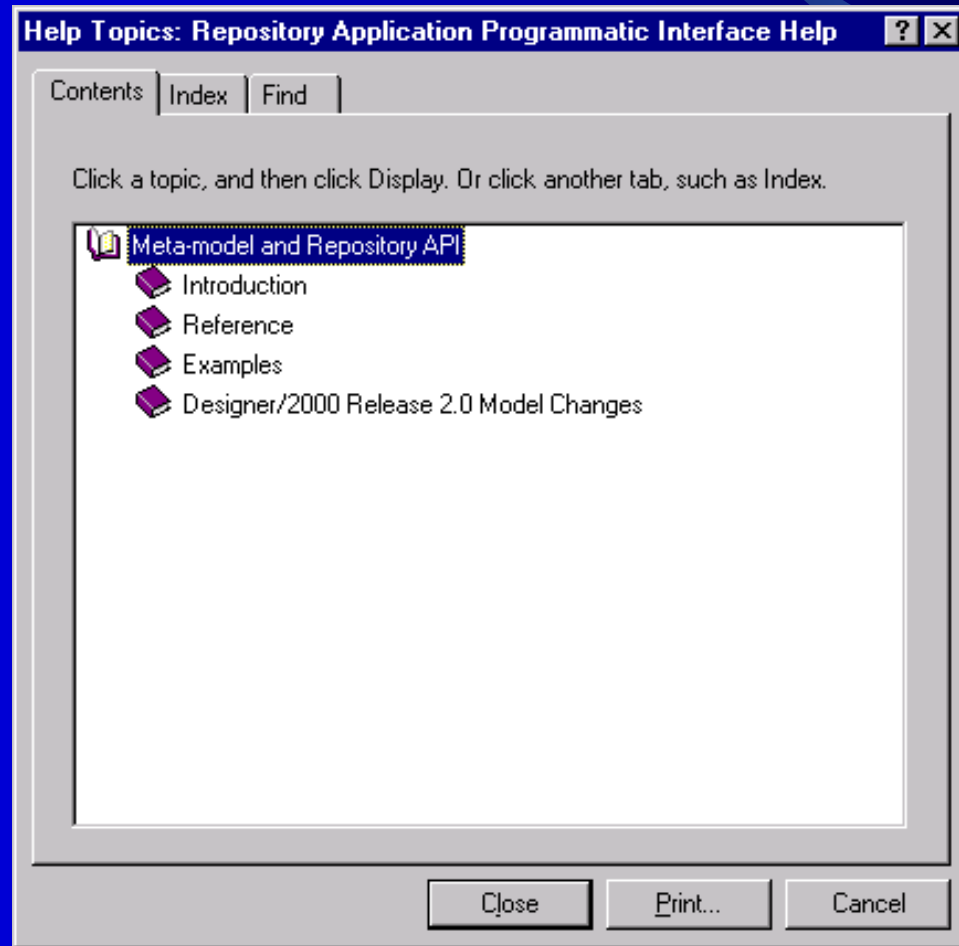
Learning about the API

- Some learning curve required to successfully use the API
- Time needed to create program code to access the repository
- Spend some analysis and design time before building an API routine
- A good understanding of PL/SQL is useful in working with the API

Getting API Help

- Look at Oracle Designer Help System help topic link “API”
- *Designer 2.1* → *Designer 2000 API* from Windows Start menu
- *Designer 2.1 Bulletins* → *Repository API SRB* from Windows Start menu
 - System Release Bulletins
- Meta-model diagram
- Meta-model application system

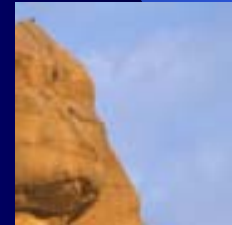
Designer 2.1 → Designer 2000 API from Windows Start menu





Secret 1 - Getting API Help in RON

- Problem
 - You need the ID number of an element you're working on in RON to query it in an API view
 - You cannot see the ID number for elements in RON or the Design Editor (or Object Database Designer).
- Resolution
 - Select the element in the hierarchy
 - Click a property in the property palette
 - Press F5.
 - Review the Property Details window for:
 - The ID of the element along with the object type, base table name, and details of the property.
- Demonstration



Programming the Application Programmatic Interface

What to do Now?

- Develop those front-end programs
- Supplement Oracle Designer
- Create Designer/2001
- Brush up on your PL/SQL

An API Query

```
SELECT a.name, a.format,  
       a.maximum_length,  
       a.optional_flag  
FROM   ci_entities e,  
       ci_attributes a,  
       ci_application_systems s  
WHERE  e.id = a.entity_reference  
       AND s.id = e.application_system_owned_by  
       AND s.name = 'CTA'  
       AND s.version = 1  
       AND e.name = 'EMPLOYEE';
```

The API Packages

- PL/SQL Packages correspond to the views
 - That is, packages correspond to repository elements
 - Example:
 - View: CI_TABLE_DEFINITIONS
 - Package: CIOTABLE_DEFINITION
- More than 400 PL/SQL packages make up the API.
- Use your favorite PL/SQL front-end

Package Contents - Procedures

- DML operations
 - INS for insert
 - UPD for update
 - DEL for delete
 - SEL for select (same as views)
- Example: CIOTABLE_DEFINITION.UPD

Package Contents - Procedure Parameters

- Parameters of procedures
 - ID number, record variable
- ID number is repository element ID
 - Can get this ID using the Secret 1 technique
- ID number is also in the API view

Package Contents - Record Variable

- Called data
 - Record of records
 - `r_tab ciotable_definition.data;`
- Overall record has 2 parts
- v for value
 - actual property value
 - `r_tab.v.display_title := 'Employees';`
- i for indicator
 - Boolean TRUE if the value is to be processed
 - `r_tab.i.display_title := TRUE;`

Package Contents - Record Variable

- v and i are each records
- components are the properties of the element
- Example
 - data.v.null_indicator
 - data.I.null_indicator
- Each property has a member variable

Package Use in a Nutshell

- Set the record variable `v` member
 - values of properties
 - don't ignore mandatory values
- Set the record variable `i` member to `TRUE`
 - Only for the property variables that changed
- Call the DML procedure
 - pass the record variable (for `INS` and `UPD`)
 - pass the ID number (for `UPD`, `DEL`)
 - `ciotable_definition.upd(v_tabid, r_tab);`

The API Transaction Model

- Like the SQL commit-rollback model
- Ensures consistency in repository
- Validates dependencies

Remember that before you try any technique, outside of the Oracle Designer front-end tools, that alters your repository you should either backup your database or export your repository information.

Transaction Model - Consistency and Dependency

- Nullable property is False
 - % Used property must be 100%
 - API does that for you
- A relationship has to have 2 ends
 - Both ends must reference existing objects
 - Like foreign key relationship but the table is highly normalized table
- Normal Commit mechanism will not do

Transaction Model - Units of Work

- Based on an Activity
 - An activity works on one application system
 - An activity is like a transaction
- Each package has a record variable
 - Members of the variable correspond to the properties
- Procedures contained in CDI API package
 - `cdapi.close_activity`

Transaction Model - The Steps

- For example: Updating a Column Property as Follows
 1. Declare Record Variables (Different from Paper)
 2. Prepare the Activity
 - Initialize
 - Open Activity
 3. Load record variables
 4. Call “DML” procedure
 5. Validate
 6. Close Activity, handle errors

1. Declare Variables

```
DECLARE
```

```
    r_col    ciocolumn.data;  
    v_status cdapi.activity_status%TYPE;
```

2. Prepare the Activity

- Initialize (app system, version)
`cdapi.initialize('CTA',1);`
- Open
`cdapi.open_activity;`

3. Load Record Variables

```
r_col.v.null_indicator :=  
    'NOT NULL';  
r_col.i.null_indicator := TRUE;
```

4. Call DML Procedure

- Call “DML” procedure
`ciocolumn.upd(3174, r_col);`
- Record variable `r_col` contains all properties and indicators

5. Validate the Transaction

- Validate

- A couple of steps

```
cdapi.validate_activity(v_status,  
    v_warning);  
cdapi.instantiate_message;
```

- if errors

```
cdapi.abort_activity;
```

6. Close the Activity

- Close Activity
 - `cdapi.close_activity(v_status);`
 - `v_status` holds return value from the Close
- Don't close yet if you want to do something else in that same activity
- Otherwise, you are now ready for the next call

The Whole Enchilada

```
DECLARE
    r_col ciocolumn.data;
BEGIN
    cdapi.initialize('CTA',1);
    cdapi.open_activity;
    r_col.v.null_indicator := 'NOT NULL';
    r_col.i.null_indicator := TRUE;
    ciocolumn.upd(3174, r_col);
    cdapi.validate_activity(v_status, v_warning);
    cdapi.instantiate_message;
    cdapi.abort_activity;
    cdapi.close_activity(v_status);
END
```



Secret 2 - The Validation Performance Hit

- You can call `cdapi.close_activity` without first calling `cdapi.validate_activity`. This will provide an increase in performance while maintaining the same level of data integrity checking, but will not display any warnings.



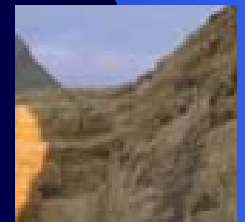
More Information on the API

- Documented in on-line help system
 - Top Level
 - Sample program excerpt there
- Oracle White Paper
 - `tiburon.us.oracle.com/opd/archive/index.htm`
- Load the meta-model app system
- Chapter 28 from the Oracle Designer Handbook, 2nd Edition
- IOUW '99 Paper 705



Secret 3 - Manipulating Multiline Text

- No API for Text types
- Solution for Queries: Do DML directly on CDI_TEXT table
- Solution for DML: RMOTEXT.READALL and RMOTEXT.WRITEALL



Querying Multiline Text

Name	Null?	Type
-----	-----	-----
TXT_REF	NOT NULL	NUMBER(38)
TXT_SEQ	NOT NULL	NUMBER(6)
TXT_TYPE	NOT NULL	VARCHAR2(10)

- TXT_REF = ID number of the element this text describes

- TXT_SEQ column is the line number

- TXT_TYPE is a code

- For example, CDINOT is notes, CDIPLS is the PL/SQL block text.

TXT_NOTM	NUMBER(38)
-----------------	-------------------

- TXT_NOTM currently unused , intended to store the number of times that this record was modified

TXT_TEXT	VARCHAR2(2000)
-----------------	-----------------------

- TXT_TEXT is the actual text

Sample query to CDI_TEXT

- To extract the User Help text for the STUDENT table in the version 1 CTA application system:

```
SELECT txt_text
FROM cdi_text
WHERE txt_type = 'CDHELP'
AND txt_ref =
    (SELECT t.id
     FROM ci_table_definitions t,
          ci_application_systems a
     WHERE t.application_system_owned_by =
           a.id
           AND a.name = 'CTA'
           AND a.version = 1
           AND t.name = 'STUDENT')
ORDER by txt_seq;
```

Extra Tip: RM_TEXT_TYPES

- *You can view a list of the text types for the and descriptions of those types by querying the RM_TEXT_TYPES view
CDI_TEXT.TXT_TYPE column*

```
SELECT text_type, nls_desc
       FROM rm_text_types
ORDER BY text_type
```

TEXT_TYPE	NLS_DESC
ABTRUL	Abstract Type Rules
ALCMNT	Application Logic Comment
ALCODE	Application Logic Code
BPRCIN	Competitive Index Notes
...	

Overview of Multiline Text DML



Secret 4 - Create Use Case Type Diagrams

- Demonstration

Note: The Business Process Modeller truly shows ownership of a process with a Business Unit rather than interactivity or communication with that Business Unit. So, if you choose to use this method to create Use Case diagrams you are choosing to use the Process Modeller outside of the scope of its original purpose.



Conclusions

- API is quite powerful
 - Makes up for some of Designer's limitations and bugs
- Also are a lot of work
 - Do it only if you really have a task that needs it
- If you do want to do a lot of this, make it a product

Author Contacts

- **Please fill out the evaluations - paper #705**
- **Douglas Scherer**
 - www.coreparadigm.com
 - dscherer@coreparadigm.com
- **Peter Koletzke**
 - <http://www.mvsn.com>
 - 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.

