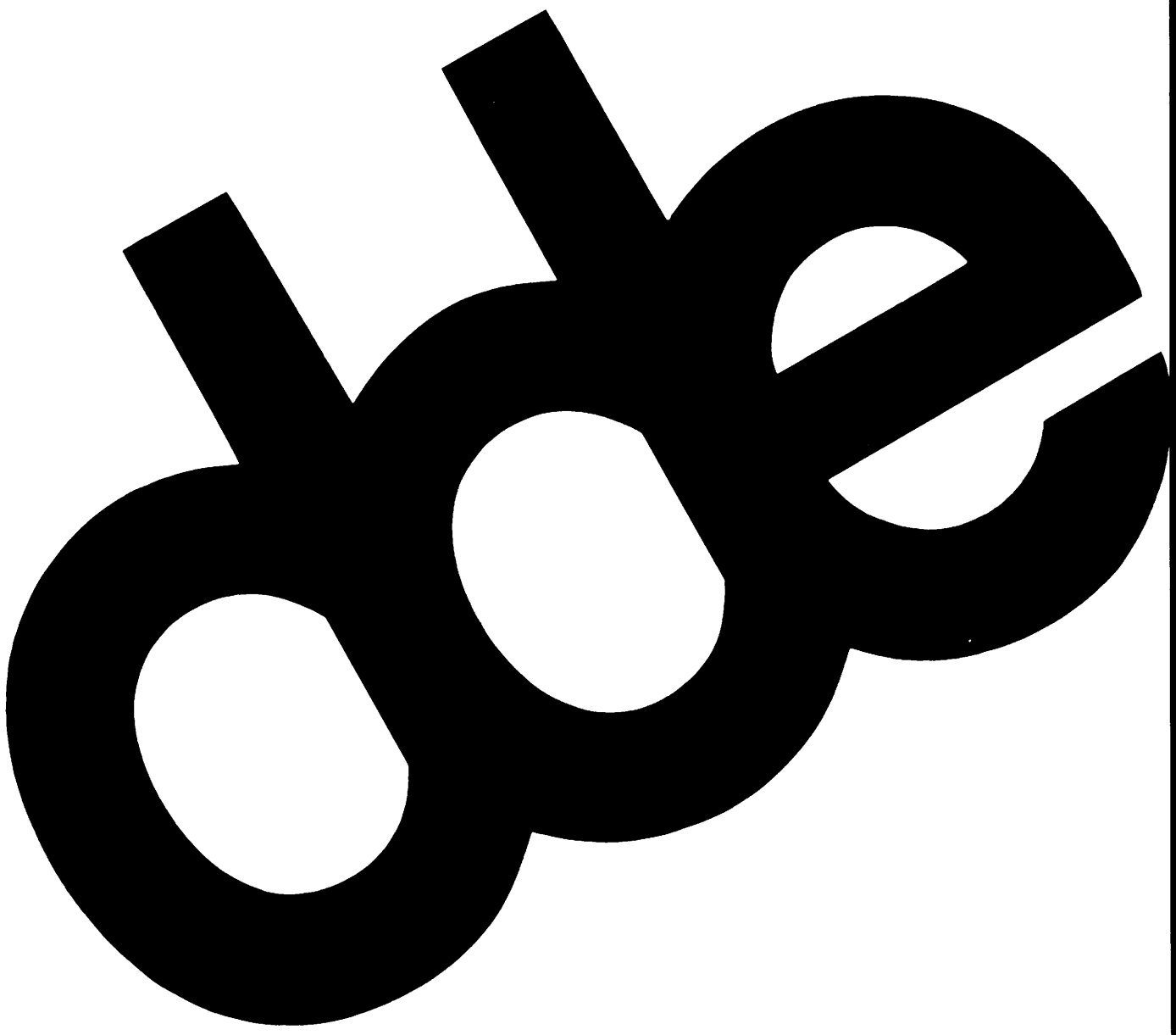
Designer 2000 Usage Hints



24 Maj 96

Designer/2000 Usage Hints

The present paper is written as a contribution to the debate about which ingredients are applicable to a development environment in order to develop good and robust bug fixed programs.

All factual information relates to release 1.1 of Designer/2000, unless otherwise stated.

Version 1.1.1 of this documnet includes a little hint on retrofitting and maintaining domain information. And comments on table and column definitions. Also more information on Domains is given.

Version 1.1.2 of this documnet includes a more constraints to be observed when aiming for later use of Oracle replication. Also a hint on Default where clause settings are presented. And a hint on how to have the same function in more locations in a Function Hierarchy Diagram. And some hints on how to keen an project oriented overview of what goes on in the repository.

Version 1.1.3 of this document includes more sql-statements on how help information may be extracted from the repository.

Version 1.1.4 of this document includes a little hint on line feets in the column help_text property - and how to detect them. And some remarks on *100% generation.*

Version 1.1.5 of this document includes information on how objects from one application system may be shared by another.

Version 1.1.6 of this document includes information on how *load* and *unload* may be used as a source for version controle. And a hint on reversing database objects from a remote database.

---

1. Objective

The objective is to make as good use of the features in Designer/2000 as possible, and prevent developers from falling into the same traps as others did before them.

2. Installation Advice

2.1 NLS_LANG

Make sure the NLS_LANG environment is set to a language where error messages do exist, and to a character set where 8-bit characters are permited. Such as:

NLS_LANG=english_England.WE8ISO8859P1; export NLS_LANG

Otherwise you may also get problems with generated code for decimal seperators in decimal numbers.

2.2 Access to SQL*Plus from the RON

If the SQL*Plus selection in the *Tools* menu in the RON should be possible, you would have to copy the line *EXECUTE_SQL=PLUS31* from the *[PLUS31]* to the *[Oracle]* section in the **c:\windows\oracle.ini** file.

3. Techniques

3.1 Domains and Constraints

3.1.1 Using Domains The term *domain* can be regarded as a kind of user-defined abstract type definition very useful indeed in analyze as well as in design.

A domain can be a set of business validation rules, format constraints and other properties that apply to a group of attributes. For example, a list of valid values, a legal range, or any combination of these.

Also at the design level domains are useful, as columns originated from a certain attribute, has the same domain, which allow the designer to enter further design information for a large group of columns as well, such as detailed validation rules, mappings to actual database datatypes, default values. etc.

Assume in the analyse session of a finance system that a lot of attributes should actually hold amount as well as weight information. By specifying

all amount attributes to belong to a *Amount* domain, and all weight attributes to belong to a *Weight*, will right away allow for a more central description of what we in this system think is essential about say amounts. Also we may check if only amount based attributes are compared to each other.

We may also specify that certain action should be taken on every amount based attribute where ever it exists, such as revision tracking, security rules, etc.

At the design level we may choose a common database datatype for all amount based columns, which could be different for all weight based columns. And we could centrally choose a distinct display format mask for all amount based fields in our applications, which could be different for all weight based fields.

Using Domain specifications right from the analyze level will thus for the basement for yet another way to increase quality and robustness in the application at hand.

Observe that many systems today allow for specifications of not only domains, but also for super and sub-domains, which should not be mistaken as super and sub-entities.

In Designer/2000 a number of limitation on domains exists:

3.1.1.1 Only one level of Super Domains Assume we have a super domain called *DOM_DATE_TIMES* (dates including time in general), then it is perfectly legal to define a new domain called *DOM_ONLY_DATE*, with *DOM_DATE_TIMES* as its super_domain. But when you try to create a domain *DOM_DATE_WITH_DEFAULT* with *DOM_ONLY_DATE* as its super_domain you get error CDA-01006, telling that:

Domain *DOM_DATE_WITH_DEFAULT*: Super-type DOM_ONLY_DATE is itself a sub-type of a domain.

And the explanation suggests that there are loops in our Domain hierarchy, which is not true.

3.1.1.2 Inherit information from a super Domain Assume we are using domains to specify detailed information of subclases of attributes and columns.

It appers to us that sub domains are not very usefull from a generator respect.

Assume we have domain *A* as a number with max length 1 and allowed values 1, 3 and 5, as well as a subdomain *B* with no extra settings at all.

Now there is no easy way to let domain *B* inherit all information from domain *A*.

And when we have a column of domain *B*, no information from domain *A* is automatical inherited to the column, which can be seen using the DDL generator.

3.1.1.3 More than 1000 Domains I you do have more than 1000 domains in an application, and when the RON tries to load them the warning 21057 appear (Only the first 1000 domains have been loaded).

This can be delt with in the RON:

Highlight the *Domains* node in the RON but do not expand it. Press the F7 key (Hopefully) A dialog window appears.

This dialog applies a filter to the objects being returned It requires a WHERE clause fragment (against SDD_ELEMENTS table) in the form: *el_name like 'XX%'* or *el_name > '<letter>'* to get a section of the list.

There are however no way of getting arround the limit when you have to select a proper domain for an attribute or column - besides programming a little PL/SQL script using the Designer/2000 API.

3.1.2 Constraints The term Constrains is a rather broad one. It makes sense in the analyse level. Choosing an identifier for an entity constrains the tuppels of that entity. Forming a relationship between entities also constrains tuppels of the involved entities as a certain membership is foreseen.

One could state that analyzing a system is in fact very much about putting appropriate constraints on a set of entities.

A number of other kinds of constraints do exists. The advantage however of the two mentioned previously is that they may be graphicly represented. But if a certain constraint does document a certain business rule or desired behaviour, we still advise you to describe that constrain.

Other kinds of constraints are:

- Domain constraints documenting a certain rule of validation or action on all attributes of a given domain. As an example one could have a social

security number domain with a domain constraint on how to validate these numbers.

- Intra tupple constraints documenting a certain rule of validation or action on a set of attributes within each tupple of a certain entity. As an example start dates are often required to be before end dates within a given tupple.
- Inter tupple constraints documenting a certain rule of validation or action on a set of attributes across all tuppels of a certain entity. As an example one could state that no more than 10 accounts per customer are permitted. In fact alternate key specifications is a sort of inter tupple constraints documenting that a certain attribute or set of attributes must be distinct across an entity.
- Transition constraints documenting if a certain changes is legal. As an example one could state that the salary of a given employee cannot be raised or lowered with more than 10%.
- Cross database constraints documenting other rules of validation or action across the entire set of entities. Mutual exclusive relations are specific examples of such constraints.

Most tools do not support all these different kinds of constraints, but we advise the kind of constraint to be documented anyhow along with the constraint itself.

3.2 Analyze Object Dictionary

It is possible from the repository of Designer/2000 view all objects from the analyze level using the view *d2k_object_dict*. This view may for some of the objects look like this:

```
create view d2k_object_dict as
select a.as_name||':'||a.as_version application,
  at.name name, 'ATTRIBUTE' object_type,
  'in entity '||e.name||', of domain '||nvl(d.name,'null')||' - '||at.notes note
from sdd_application_systems a, ci_attributes at,
    ci_entities e, ci_domains d
where e.application_system_owned_by = a.as_ref
  and at.entity_reference = e.id
  and at.domain_reference = d.id (+)
union
select a.as_name||':'||a.as_version application,
  d.name name, 'DOMAIN' object_type, d.description note
from sdd_application_systems a, ci_domains d
where d.application_system_owned_by = a.as_ref
union
select a.as_name||':'||a.as_version application,
  e.name name, 'ENTITY' object_type, 'Entity' note
from sdd_application_systems a, ci_entities e
where e.application_system_owned_by = a.as_ref
union
select a.as_name||':'||a.as_version application,
  e.short_name name, 'ENTITY' object_type,
  'Entity - short name of entity '||e.name note
from sdd_application_systems a, ci_entities e
where e.application_system_owned_by = a.as_ref
union
select a.as_name||':'||a.as_version application,
  e1.name||' '||r.name||' '||e2.name name, 'RELATIONSHIP' object_type,
  r.remark||' - min,max: '||to_char(r.minimum_cardinality)||','||
    nvl(to_char(r.maximum_cardinality),'many') note
from sdd_application_systems a, ci_relationship_ends r,
    ci_entities e1, ci_entities e2
where e1.application_system_owned_by = a.as_ref
  and r.from_entity_reference = e1.id
  and r.to_entity_reference = e2.id
```

Located in the *$DEVENV/D2K/cre_d2k.sql* script.

4. Usage Advice

The following list of advice is by no means complete, and skilled developers will be able to find cases, where the advice is not very applicable. Still, I suggest the advice should be discussed among Oracle Forms developers.

4.1 A to-do List

In order to keep related information in the Designer/2000 repository consistant, a number of different actions have to be performed. Assume that a repository have been populated with ER information from the top, and that design information has been derived, Functional information entered and that Modules have been created.

If now changes are made to the ER-datamodel, a number of actions must be considered, some of them are:

- Run some quality check reports *quality - Entity with no ...* and *Datamodel - Quality checking of Relationships* to make sure that the ER-datamodel is consistant.
- Assuming that some domains did change, the two utilities *Update Attributes in Domain* and *Update Columns in Domain* would have to be executed. Later in this document it is shown how this may be done in batch.
- Now run the *Database Design Wizard* with options agreed uppon in the project.
- Consolidate Table and Column information from Entity and Attribute information. (Posibly with a Dedicated Data Wizard).
- Run some *Impact Analysis* reports to see possible impact to the rest of the system.
- Run the *Generate Reference Tables -> Update Help Tables* utility in order to populate help information from the tables and columns.
- Run the *Generate Reference Tables -> Update Code Control Tables* utility.
- Run the *Generate Reference Tables -> Update Reference Code Tables* utility.
- Run some quality check reports *System Design - ... Quality Control* and *Database Design - Invalid Database Object Quality Control* to make sure that the database related objects are consistant.
- Check that no reserved word is used by any database object. See the section *Naming Convension -> Reserved Words* later in this document.
- Run the *Generate SQL DDL* utility for all database objects.
- Before these objects are applied to the database, you may need to change some of them. Some of the tables may in fact be Views, ...
- Apply these objects to the database.
- Create diagnose scripts capable of performing a validation of the customer site information against the Designer/2000 repository

information. Either created manually or automatic uding the diagnose creation scripts mentioned later on in this document.

Assuming also that you have new functional definitions to implement:

- Flip the *Candidate* flag at the screen of report modules.
- Refine the modules through the MSD and MDD. Use the *Generate Default Links* utility to connect DTU's the way foreseen by the ER-model.
- Run the *Update Help Tables* utility in order to populate help information from the modules.
- Make sure all nessesary and agreed preferences are set to agreed values.
- Consolidate DTU and DCU's with table and column definitions using reports from the repository. (Posibly with a Dedicated Module Wizard).
- Run some *Impact Analysis* reports to see possible impact to the rest of the system.
- Make sure the templates are as rich as possible and linked to the correct modules.
- Run the *[Re]Generate Module* utility for all modules in question.
- Do any presentational and navigational hand crafting in the generated modules.
- Check the generated Modules.

4.2 In general

4.2.1 Error Codes All Designer/2000 error codes are stored in files in the dictionary **oracle_home\repadm10**, which are useful when more help is needed.

These files are in use if you from the *generate* screen click on an *error*, and then on the *more* button.

4.2.2 Command line Possibilities Almost no utility or report has a documented way to be started from a command line. But some can be run from the command line some cannot. All the utilities are Windows based so you must run them from a Windows command line.

Designer/2000 related programs from *orawin\bin* are: (Some of the executables are not pressent in release 1.1, and some are not pressent in release 1.2)

awd20	Data flow Diagrammer
awe20	Entity-Relation Diagrammer
awf20	Function Hierarchy Diagrammer
awm20	Matrix Diagrammer
bpmod10	Process Modeller
cddi55	Oracle Server Generator
cddl55	Oracle Server Generator
cdrep10	Oracle Report Generator
cdrf55	Oracle Server Generator: Reverse Engineer
cdrk55	Oracle Server Generator: Reconcile
cfct45	Update Code Control Table Utility
cfrv45	Update Reference Code Table Utility
cgen45	Forms Generator
cghp45	Help Table Utility
cgor25	Report Generator
ckld10	RON: Load
cknsetdg	RON: Diagnoses
ckrau60	Repository Admin Utility
ckron10	Repository Object Navigator
ckrpt10	Repository Reports
cksoe10	RON: Oracle Exchange
ckul10	RON: Unload
ckver10	RON: New Version

cvvbg10 Visual Basic Generator

dwm10 Module Structure Diagrammer

dws10 Data Diagrammer

iwgpn10 Preference Navigator

iwmdd10 Module Logic Navigator

iwmln10 Module Data Diagrammer

4.3 Analyze Level

Please note, that information such af format masks, prompt and uppercase information is regarded as design information and is as such not relevant at the analyze level. You may however store this information in some of the user defined property items if nessesary.

4.3.1 Establishing the Function Side Assume the ER-datamodel is already pressent in the repository. Then the Function Hierarchy should be created, in order to show the overall relations between the different functions and menues. Also the functions should be told which entities thay may operate on, and do not forget to tell which attributes of the actrual entities should be included. This can be done using the Function Hierarchy Diagrammer (FHD).

Note that it is possible to have the same function in more locations in the same FHD! This is done by letting all the copy functions be *common*, referencing to the actual master function. The copy functions will then inherit the actual description text, but will still have theit own labels.

Now the *Application Design Wizard* may transform leaf-node functions into proper modules at the design level. But in order to work with the generated modules, you would have to enter the modules through the RON and change the *candidate* property to*null*. Note that the Wizard cannot generate the modules if the modules already exists!

Now a Module Structure may be generated using the available Modules using *Include Network* from the *Module Structure Diagrammer* (MSD). If a new module is created instead of the old one MSD may ask you to *Consolidate Network* - In some occations though the modules disappear from the diagram and have to be included again with *Include Network.*

And finally the Module Data Information may be entered through the *Module Data Diagrammer* (MDD), which is where much of the module design takes place. Note that the Entity information entered through FHD, is here transformed to table information, and that possible relations from the ER-model will be included running the *Generate Default Links* Utility. I do not know how to make the read-only modules appear as Lookup block automaticly, and the utility appears to include no columns in the lookup module? I also do not know how to get the circular relationships from the ER-model over as links here - or if it make sense?

Also note, that if you try to *Consolidate* a Module Data Diagram, because a module has been deleted and recreated, the MDD will notify you that the module is deleted, but will then loop with an *Internal Application Error*, (as of version 1.1 on my windows PC).

The module format will usually be filled automatic. The *Lable* and *Control Block* values are used by reports. The *LOV* and *Matrix* values are used by forms. And the *Network, Hierarchy* and *Master/Detail* all mean the same thing. The *Matrix* value is used to inform the system that a master / master / detail system is at hand. Normally only the *LOV* value needs to be set in order to tell that an alternative LOV form is to be called instead of the generic forms LOV feature.

4.3.2 Save often in FHD Unfortunately the Function Hierarchy Diagrammer (FHD) is not very stable (version 1.1 - under Win 3.11), so be sure to save whenever a new entity or attribute has been entered.

4.4 The Database Design Wizard

This rutine will try to move Entity Relation Information from the Analyze level down to Table and Column Information at the Design Level. The following information is relevant for version 1.1. For version 1.2 you would need a patch, in order to be able to get more than 100 entities in the main selection list (bug number 328197).

4.4.1 Tables The Wizard may deal with super- and sub entities, but cannot combine two entities in a one to one relation as one table. Select the sub-entities, if you want them to go into seperate tables.

Foreign Key attributes are inserted for each relation where this table is a child.

- *Display Title* is set to *Initcap(table_name)* with spaces for underscores.
- *Create ?* is set to *Yes*.
- *Description Text* is copied from the Entity Description Text.
- *Notes Text* is copied from the Entity Notes Text.
- *User Help Text* is copied from the Entity Description Text.

Database Type, Database Name, Tablespace, Table prefix and others may get options in a DDW run.

4.4.2 Columns

- *Uppercase* is set to *False*.
- *Prompt* is set to *Initcap(column_name)*.
- *Display Length* is set to the maximum length from the attribute, except if the datatype is Varchar2, in which case the display length is lowered to a maximum of 70 characters. The display length for a Number field is set to one more than the maximum length to account for a sign, plus one if decimals are allowed - and to 39 if no maximum length is specified. The display length for a Date field is not set at all.
- *Description Text* is copied from the Attribute Description Text.
- *Notes Text* is copied from the Attribute Notes Text.
- *User Help Text* is copied from the Attribute Description Text.
- *Datatype* is in some cases changed. *Video* and *Photographic* are changed to *Long Raw*. *Time* and *Timestamp* are changed to *Date*. *Char* and *Varchar* are changed to *Varchar2*. *Text* is changed to *Long*, and *Money* is changed to *Number*.

  This is also documented by the select statement:

```
select ref_code analysis, ref_meaning design
from ref_values
where ref_domain = 'DDW_DATA_TYPE';
```

4.4.3 Primary Key A Primary Key Constraint is generated corresponding to a primary Unique Identifier.

- *Validate In* is by default set to *SERVER*, but may be changed using an DDW option.

- Some diciplin is needed to create a view for all tables, and to remember to update a view if a column is added - or deleted from a table.
- Most of the Modules should then use Views rather than tables for their DTU's.
- The Database relation Diagrams does certainly not get more easy to understand.

So you may want to hide the fact that some (most) of the tables known to the Designer/2000 repository are in fact views. But of cause then you would have be responsible for changing (and managing) the DDL creation scripts. And you would not be able to use the Retrofit utility of tables directly.

4.5.2 Creating a View Object You may create Views through the RON, but it might be esier through the DD, like this:

1. Enter DD, and inside a diagram, select the icon: *Add a view definition to the current diagram*, place the new object on the diagram, and give the new view a name.
2. Double click on the new view to enter the view definition sheets.
3. Make sure that the check *Select Text* is not selected at page *View*.
4. At page *Base Relations*, choose the tables, views or snapshots to base the new view uppon. Note that the sequence number is mandatory, and that it is difficult to change the sequence once given.
5. At page *Base Cols*, point at the columns from the selected tables to have in the select part of the view. Note that the sequence number is mandatory, and that it is difficult to change the sequence once given. Also note that a column does not have to be based on a *Base Column*, but would have to have an *expression* instead, (scroll to the right).
6. Insert other information at the other pages as appropriate.
7. At page *View* select the *Text* button and choose a *Where/Validation Condition*. Begin the text with *where* followed by the total where clause. Note that join clauses have to be entered manually, as this information is unfortunately not taken for the relation description.
8. Save by selecting *Ok*, and check the specified view through the RON. Also choose a database for the view or set *All Databases* to *true*.

9. Generate the DDL for the view using the DDL generator.

4.5.3 After running the ADW After running the Applcation Design Wizard (ADW), creating default modules from FHD, you should (from the RON) change the *candidate ?* flag from *True* to *null* in order to work with the modules in the Module tooles (MDD and MSD).

4.5.4 LOV List of values (LOV) defined in a domain (Allowable Values) is well handled in the module by using an appropriate field type such as a check box, combo box or others.

If a LOV is not defined in a domain, it may be entered in the MDD as a *look up table* (instead of a *base table*. This is a possible way to define extra columns to be included in the look up table list, as well as possibly display some descriptor columns in the form. A *where* and *order by* clause may also be entered for the list of values.

4.5.5 Help Specifying Help on all tree levels (Module, DTU/Block/Table , DCU/Field/Column) gets generated nicely in the generated forms. But do remember to run the *Generate Reference Tables -> Update Help Tables* Utility.

The actual values of the help_text property for columns would be visible through the *ci_columns* view instead of the more general *cdi_text* table.

Be sure not to enter any line feets for this property, as you may get the following errors if you later on UNLOAD and LOAD the information (seen in release 1.2)

CDB-00182: Unfinished token in line ...
CDB-00194: Invalid M command syntax on line ...
CDB-00174: Unknown command type on line ...

You may actually check if any column does have this weakness, running the following select against the repository:

```
select t.name||'.'||c.name
from ci_columns c, ci_table_definitions t
where t.application_system_owned_by = &&APP_ID
 and c.table_reference = t.id
 and c.help_text like '%'||chr(10)||'%';
```

Please also note that the column *prompt* property and the module detail column usage (DCU) *hint_text* and *prompt* property should not include line feets either.

4.5.6 One Form Module calling an other If one Forms based Module should call another forms based module, the generator will move the actual values to global variables in the calling form. The called form may now inspect these values. The code for this in the called form will be generated by the generator if the first block is a query only block and is based on the same table as used in the calling form.

And if the Windows Layout Preference *WINUSR* for the called form is set to *N*, then the called form will act as a pop-up uppon the calling form.

It is required however that the table used by the module from the calling form does have a primary key.

The following SQL-statement will reive a list of DTU's with the actual table used as well as information on Module and kind of DTU, with up to two lockups from a base block:

```
select m.name module_name,
    nvl(nvl(dtu.block_name, dtu_prev1.block_name), dtu_prev2.block_name) block_name,
    t.name table_name, dtu.usage_type
from ci_modules m, ci_module_detail_table_usages dtu,
   ci_table_definitions t, ci_module_detail_table_usages dtu_prev1,
   ci_module_detail_table_usages dtu_prev2
where m.id = dtu.module_reference
 and dtu.table_reference = t.id
 and dtu.mti_reference1 = dtu_prev1.id (+)
 and dtu_prev1.mti_reference1 = dtu_prev2.id (+)
order by m.name, dtu.usage_sequence;
```

4.5.7 Self-referencing Tables If you have a self-referencing table, it is possible to generate a server-side constraint to check that every row in that table has a primary row. This works also fine for the very first legal row (as it obviously gets stored in the database and afterwards gets checked). But if you only asks for a client check, the first row will not get committed to the database if the foreign key is a single column because a *WHEN-VALIDATE-ITEM* trigger is fired and will fail. If more than one columns composes the foreign key a *WHEN-VALIDATE-RECORD* trigger will fire after a *POST*, and will succede.

Anyway the very last row can hardly be deleted using server side constraints.

4.5.8 Reducing Pagesize Be carefull reducing the pagesize of a forms based modul, as there may not be room for buttons from the template, and the generator may fail. (version 1.1).

4.5.9 Including Buttons in the generated Forms To have the generater create a button, you must create a second DCU for one of your columns and set the display datatype to Button. Then, under the Column Text tab, choose the PL/SQL Block type and enter the following command:

COMPLEX: *your_procedure_name*

Then, you will need to create a procedure called *your_procedure_name* either in your template, in an attached library of the template or in the database as a stored procedure.

Designer/2000 will then create a button in your generated form with a WHEN-BUTTON-PRESSED trigger that will call your_procedure_name. The name of the button will come from the prompt of the second DCU.

4.5.10 Including other Control Items Items not based on database columns can also be included in a module. To make Designer/2000 create a check-box you may enter MDD, add a dummy column to the DTU by inserting one of the existing columns again. In *Detailed Usages for Screen Module, Column Text* select the added column set the type and select the type: *DDerivation Expression* and enter *COMPLEX: function_name(current column parameters).* Remember to set the *ColumnDisplay DisplayDatatype* to *Checkbox*, as well as the prompt.

Note that if only a simple expression is used you must use one or more of the actual columns from the block.

We have seen cases in version 1.1 where it was impossible to change a dummy character field to *date.*

BTW: functions should be called with actual DCU names without any prefix, which is a problem if the scope includes more columns of the same name as the following little example shows:

Assume we have a Module over the good old EMP and DEPT tables, DEPT being a look-up. Also assume that we want to have a derived field in the EMP block where we need to pass the *deptno* value as parameter to a complex function.

COMPLEX:tst_deptno(deptno)

Designer/2000 now tells us that deptno is defined more than once, so we change to:

COMPLEX:tst_deptno(emp.deptno)

Now generating the forms module an error occures in a generated procedure:

```
/* CGF$GET_EMP_DRV_EMPNO */
PROCEDURE CGFD$GET_EMP_DRV_EMPNO(
  P_DRV_EMPNO IN OUT NUMBER   /* Item Being derived */
 ,P_DEPTNO   IN    NUMBER) IS /* Item value        */
/* This derives the value of a base table item based on the */
/* values in other base table items.                        */
BEGIN
P_DRV_EMPNO :=
 tst_deptno(emp.P_DEPTNO);
END;
```

Obviously no P_DEPTNO column exists in the emp block!

Verified by Oracle to be a bug in version 1.1 and 1.2 (bug no: 271523).

4.5.11 Including Blocks not based on an Application Table Any block that isn't based on a table (or that is based on a table that is not part of your application (i.e. a preferences table)), must be placed in the template form. The generator will only generate blocks based on tables in your module, but it will happily add any other blocks in the template form to the generated form.

4.5.12 Using Presentational Preferences If the page, canvas and view dimensions are not stated in the table usage in a module the generator will use general, application or module specific preferences for these matters. But if you later on change these preferences, and the old values have already been inserted in the modules, the preference values will not be used by those modules. You would have to clear the properties in question in order to take advantage of the preference values.

4.5.13 Enter a Query Block Some preferences will controle what should happen when a query block is enteres.Should the query be executed automatic, or wait for the user to specify a query, or should the user be able to decide at runtime. Including the *CG$QI* item in the template, will be used by the forms generator to create a toggle between the two query possibilities.

4.5.14 Preferences for Dates and Timestamps Under *Layout_Text_Item* you may set the preferences *TXTDDF* for the actual format of the Date DCU's, and *TXTDDL* for the actual display length of the Date DCU's.

Note however the generating modules when *TXTDDL* is set will let the tool update the repository withe the used display length velues. And as actual

values in DCU's has priority over preferences changing *TXTDDL* for a later generation will only have effect if the display length gets clered.

Likewise for Timestamps the preferences *TXTDTF* and *TXTDTL* exists.

4.5.15 Stretching Fields Sometimes the actual fontsize does not fit the actual display length of a field, forcing the user to scroll the fields horizontally in order to view the whole contents.

Under *Layout_Text_Item* you may set the preferences *TXTSCA* for asking the generators to scale Date, Number and Uppercase DCU's up or down at your desire.

4.6 Implementation Level

Being from a country where national letters exists, you often have to include a proper Characterset clause to the create database statement. But this issue cannot be specified to the Server Generator Utility as this property does not to exist. (version 1.1)

This means that all databases generated using the Server Generator default to US7ASCII. If this is a problem, you could edit the .db file generated before running it in SQLDBA / Server Manager.

4.6.1 Windows sizes in generated Forms Modules The size of the MDI window of the generated form may be controled. Simply add the following to the *WHEN-NEW-FORM-INSTANCE* trigger of you template form:

```
-- this will set the MDI window to a 6.4 inches X 3.7 inches (or whatever units you use)
set_window_property(forms_mdi_window, window_size, 6.4, 3.7);

-- this will maximize the MDI window as if the user clicked on the maximize button
set_window_property(forms_mdi_window, window_state, maximize);
```

4.6.2 Implemented Datatypes Note that if *integer* or *smallint* datatype is selected for a column, no explicit *maximum length* information can be used as *integer(n)* and *smallint(n)* are illegal constructs in the generated DDL. So if the maximum length information should be propagatyed to the database the datatype *number* should be used instead.

sound, video, long vargraphic and *image* are changed to *long raw. vargraphic* and *graphic* are changed to *raw. long varchar* and *text* are changed to *long*.

BTW: Do not use the datatype *Binary Integer* or the column will simply be missing in the generated DDL create statement, use *number* or *integer* instead. (Release 1.2, bug: 277600)

4.6.3 Use Comments If comments are given for table and column elements, this information is used by the DDL generator and transformed info a number of appropriate *comment on* commands.

4.7 Extra Detailed Information

The purpose for this should be to show a small number of rows (and columns) of a table in the first block with scroll allowed. And in the second block show more details of the currect row from the first block. This may be achieved through Designer/2000 in two different ways:

Design a multirecord block and then for the last columns try to display them in a singlerecord overflow area below. In *Detailed Usage for Screen Module* in Mdd Select *Overflow Style* as *Overflow Area Below*. It is hard though to control which columns will actually get to the overflow area.

The other way is allocate two different blocks from the very same table. The first one a multirecord block, and the other one a singlerecord block. In MDD draw the two blocks based on the same table without any relation between, as Designer/2000 is not capable of handling this directly. Generate the modules and in Forms the relation between the two blocks may be established to do the nessesary syncronisation.

4.8 Reverse Engineer

When doing *Reverse Engineer* from already created tables and constraints be careful to inspect the table foreign keys property *Mandatory ?*. It is NOT filled out by the reverse engineer utility and is infact mandatory (null not part of the list values). True or False must be entered to prevent errors when creating modules. Version 1.1.

Remember that if you are to reverse engineer database objects from an other oracle instance, be shure to change the *Object Owner* value from your current Designer/2000 account to the owner on the object in the remote database. Also note that if you fail to have the right privileges, try again as the manager of the repository.

Unfortunately the retrofit utility will not reverse engineers named check constraints (not the *SYS_%* named ones). So remember to check if all your

constraints do have proper names!

If you have tables with explicit names for the *not null* constraint, the constraint name does get retrofitted into the repository as a check constraint with the correct name stating that the column is not null. If you choose to generate a DDL statement for creating this table, the generated DDL statement will unfortunately not include this constraint. Version 1.2.

Also note that the required *Mandatory* property of the Foreign Key Constraint object is not given.

Now you may retrofit the table definition from the design level back up to entities at the analize level, but note that any foreign keys existing at the design level will not be pressent at the analyze level. Also note that if you want to create domains to enhance the datamodel, you should do this at the design level before retrofitting, at this information is kept at the analyze level for the attributes as well.

4.9 Runtime Dependences

Moving the generated forms and database changed to the custom platform is not enough, some PL/SQL libraries have to be shipped as well. Two of them are *ofg4tel.plx* and *ofg4bsl.plx* in the **cgenf45/admin** directory. Also *ofg4mnl.plx* is needed if menu5 modules are generated.

You may also need to export and ship the *cg_ref_codes* table, as the generated forms may lookup in this table.

4.10 Default Where Clauses

In Designer/2000 if requesting some fields in a block to have lookup facilities, the Oracle Forms generator will create a pre-query trigger to change the select statement. Now the forms developer does not have very good ways to change the default whereclause as such, since the default is dictated from the case tool. The generated pre-query trigger looks like this.

```
/* CGFK$DEFAULT_WHERE */
/* Add restrictive query on FK items if value(s) entered in non-table */
/*  items                                          */
/* and/or if there are items which may contain nulls as valid values. */
/*   Code to                                       */
/* restrict the query is added to the default where clause of the    */
/*  block.                                         */
BEGIN
 DECLARE
  block_id Block := find_block('block');
  sub_where VARCHAR2(512);
  def_where VARCHAR2(512) :=
   'the actual default where clause from the Designer/2000';

  FUNCTION add_and(
   P_WHERE IN VARCHAR2
  ) RETURN VARCHAR2 IS
  BEGIN
   IF (nvl( length(P_WHERE), 0) != 0) THEN
    RETURN( P_WHERE || ' AND ');
   ELSE
    RETURN( P_WHERE );
   END IF;
  END;

 BEGIN

  sub_where := NULL;
  IF (:field IS NOT NULL) THEN
   sub_where := add_and( sub_where ) || '(column LIKE ''' ||
     :field || ''')';
  END IF;
  IF (sub_where IS NOT NULL) THEN
   def_where := add_and( def_where ) || '((column) IN ' || '(SELECT
     column ' || 'FROM table WHERE ' || sub_where || '))';
  END IF;

  set_block_property(block_id, DEFAULT_WHERE, def_where);

 END;
END;
```

5. Naming Convension

5.1 Reserved Words

Check that the entity (or table) names does not collide with reserved words. Assume you have a table of all known reserved words in different systems in the table *reserve* with the column *r* as the word ans the column *used_id* as a system identifier. The a check could be done like this:

```
select a.as_name||':'||a.as_version application, e.name entity, r.used_in
from sdd_application_systems a, ci_entities e, reserve r
 where e.application_system_owned_by = a.as_ref
   and lower(e.name) like r.r
order by a.as_name, a.as_version, e.name, r.used_in
```

The same kind of check should be made for attributes.

```
select a.as_name||':'||a.as_version application,
  e.name||'.'||at.name attribute, r.used_in
from sdd_application_systems a, ci_entities e, ci_attributes at, reserve r
 where e.application_system_owned_by = a.as_ref
   and at.entity_reference = e.id
   and lower(at.name) like r.r
order by a.as_name, a.as_version, e.name, at.name, r.used_in
```

Actually all the database objects should be checked likewise.

5.2 Illegal Names

As the forms generator will prefix colum names with *P_*, Attributes or columns should not take this form or the forms generator may fail. Query the repository for potential attributes having these names:

```
select a.as_name||':'||a.as_version application,
    e.name||'.'||a.name attribute
from ci_entities e, ci_attributes a, sdd_application_systems a
where a.entity_reference = e.id
 and e.application_system_owned_by = a.as_ref
 and a.name like 'P%' and substr(a.name,2,1) = '_'
group by a.as_name, a.as_version, e.name, a.name
```

Or for columns:

```
select a.as_name||':'||a.as_version application,
    t.name||'.'||c.name column
from ci_table_definitions t, ci_columns c, sdd_application_systems a
where c.table_reference = t.id
 and t.application_system_owned_by = a.as_ref
 and c.name like 'P%' and substr(c.name,2,1) = '_'
group by a.as_name, a.as_version, t.name, c.name
```

6. Using the API

All changes to the repository is done by a documented set of API rutines, which is in fact PL/SQL functions and procedures in a package installed in the repository account in the rdbms. The function of this API layer is to preserve and guarantee that data integrity of the repository. There is however (at least) one exception in version 1.1 and 1.2, which is the maintenance of long text bodies in the **cdi_text** table, where proper API's

are not documented yet.

6.1 User Defined Properties

It is possible in a repository to define a concrete usage for up to 10 extra User defined Properties for each element type covering all applications in that repository. Only note, that once a certain property has been created and published, no changes to that property may be given again in the lifetime of that repository!

6.2 Get the Application System Number

Querying from the underlying repository tables, you would need to get the actual application system number. You may get this number with the following select if you know the name, and just want to work with the latest version:

```
select id, version, version_date
from ci_application_systems
where name = upper(&APP_NAME)
  and version = (select max(version) from ci_application_systems
          where name = upper(&APP_NAME));
```

6.3 Hidden Properties

Not all the columns from the background tables are vissible through the RON, as an example:

The element definition of *relationship_ends* show that a field exists with the name 'REMARK' varchar2(240) - but this property is hidden in the Repository Object Navigator.

6.4 Entities in an ERD

It is possible to select a list of entities used in different ER-Diagrams in different application systems.

```
select a.as_name||':'||a.as_version application,
  d.name diagram_name, e.name entity_name
from ci_diagrams d, sdd_application_systems a, ci_diagram_element_usages eu,
    ci_entities e
where d.application_system_owned_by = a.as_ref
and eu.diagram_reference = d.id
and eu.cielement_reference = e.id
and diagram_type = 'ERD'
order by a.as_name, a.as_version, diagram_type
```

6.5 Using the API Elements

Assume you have defined a variable *dom* as *ciodomain.data*. Also assume you know the id of a domain where some properties should be changed, then you would have to get the existing values of the properties for that object loaded into the *dom* variable: *ciodomain.sel(<domain_id>, dom)*.

Now for each property, there are two fields in the *dom* record. Like *dom.v.DESCRIPTION* and *dom.i.DESCRIPTION*. The *dom.v...* hold the actual value, and *dom.i...* is a kind of indicator variable tilling the API procedures if this property should be updated in the repository at all. Update the repository using *ciodomain.upd(<domain_id>, dom)*. followed by the API commit procedure.

This little example should clarify the point:

```
--   'Hello' would be the new value of the Description property.
     dom.v.DESCRIPTION := 'hello';
     dom.i.DESCRIPTION := true;

--   The Description property will not be changed.
     dom.v.DESCRIPTION := 'Hi there';
     dom.i.DESCRIPTION := false;

--   The Description property will be changed to nothing.
     dom.v.DESCRIPTION := null;
     dom.i.DESCRIPTION := true;

--   The Description property will not be changed.
     dom.v.DESCRIPTION := null;
     dom.i.DESCRIPTION := false;
```

6.6 Update Attributes in a Domain

Sometimes it is hard to run the *Update Attributes in a Domain* utility for all changed domains by hand, in this case you may consider using the procedure *ckoupddomain.att* directly. But please note that this is not documented at all. So be careful testing on new releases.

In order for *ckoupddomain.att* to run we must prepare parameters in the *cdi_temp_rpt_tables* table. Assume we want to update attributes from the domain with id 13676 in the application system with id 2, the script could then look like this:

```
declare
  rpt_seq number;
begin
  select cdi_tmp_rpt_seq.nextval into rpt_seq from dual;
  insert into cdi_temp_rpt_tables values
    (rpt_seq, user, 'CKOUPDDOMAIN', 'DOMAIN', sysdate, '13676');
-- More rows with other domain id's.
  rmmes.clear;
  ckoupddomain.att( rpt_seq, 13641 );
  commit;
end;
```

The *Update Columns in a Domain* utolity may be executed likewise using the *ckoupddomain.col* procedure instead.

7. Maintenance

7.1 Creating a new Application

The same Designer/2000 repository may hold many application systems (of more versions.

To make a new application system object in a given repository, simply select *New* from the *File* menu - and give the new application proper properties.

7.2 Exp / Imp of Repository Users

When exporting and importing Repository users, be sure to have the main packages in place:

```
sqlplus sys/<sys_password> @$ORACLE_HOME/rdbms/install/plsql
```

And in order to install the packages from the Designer/2000, make sure the packages **dbms_pipe** and **dbms_lock** are accessible from the repository user (grant execute ... and create synonym).

7.3 Which Preferences is set?

It is possible to select a list of all preferences set in the repository.

```
select sdd.el_name||' ('||sdd.el_type_of||') has pref '||p.preference_name name,
  'Last changed by '||sdd.EL_USER_CHANGED_BY||
  ' the '||to_char(sdd.EL_DATE_CHANGED,'DD-MON-YYYY HH24:MI:SS') changed,
  '>'||p.preference_value||'<' value,
  'Default is >'||d.value||'<' def_value,
  d.description
from ci_user_preferences p , sdd_elements sdd, CDI_DFLT_USER_PREFS d
where p.module_reference = sdd.el_id
  and p.preference_name = d.sname(+)
order by sdd.el_type_of, sdd.el_name, p.preference_name
```

7.4 Number of Extents and Sizes

The table and index objects forming the Designer/2000 repository is created without any storage specifications, which implies that some of the tables may (for large projects) run out of extents or allocate very big extents.

You could now and then run the following little SQL statement to see if some of the tables needs other storage parameters.

```
select segment_name||' ('||segment_type||')', count(*), sum(bytes)
from user_extents
group by segment_name, segment_type
 having sum(bytes) > 100000;
```

The issue may be handled by regular exports and imports, but you could also design a little script to raise the storage parameters for a small number of tables.

Some of the candidates would be: **CDI_TEXT, CK_FLATTEN_ELEMENTS, RM$1, SDD_ELEMENTS, XT_SDDV_ELEMENTS**.

The script could look like the one in **$DEVENV/D2K/alt_tab**

7.5 Parameter Table

Designer/2000 uses a temporary table *cdi_temp_rpt_tables* to hold parameters for designer/2000 utilities. Normally these parameters are deleted as the part of the very last commit from the utility, but if the utility for some reason is aborted, the rows will stay in this table forever.

You may keep an eye on this table as it informs you when a certain utility with what parameters where not terminated succesfully.

And you should perge some of the older entries as this table grows.

7.6 List of Changed Objects

It is often needed in bigger projects to be able to see which objects have in fact been created or changed in a period of time. The following two sql-statements may show this:

```
select el_type_of type, el_name object_name, el_user_created_by created_by,
 to_char( el_date_created, 'DD/MM-YYYY HH24:MI:SS' ) date_created
from sdd_elements
where el_elem_owned_by = application_id
 and el_date_created >= sysdate - number_of_days
order by el_type_of, el_name
```

And likewise for all changed objects in a period:

```
select el_type_of type, el_name object_name, el_user_created_by created_by,
  el_user_changed_bY changed_by,
  to_char( el_date_created, 'DD/MM-YYYY HH24:MI:SS' ) date_created,
  to_char( el_date_changed, 'DD/MM-YYYY HH24:MI:SS' ) date_changed
from sdd_elements
where el_elem_owned_by = application_id
  and el_date_changed >= sysdate - number_of_days
order by el_type_of, el_name
```

7.7 Sharing Objects

It is possible in Designer/2000 to share objects from one application to another. This way one application may maintain a set of primary objects, and state that named applications may use these objects - without being able to modify these objects. Note that it is not possible to tell the system that some of the objects may be public!

The following view may be used to show which objects one application does share with other applications:

```
create view shared_objects as
select n_app.as_name||':'||n_app.as_version target_app,
    n_sys.str_type_of obj_type, o_sys.el_name obj_name,
    o_app.as_name||':'||o_app.as_version source_app
from sdd_structure_elements n_sys, sdd_elements o_sys,
    sdd_application_systems n_app, sdd_application_systems o_app
where n_sys.str_use_of = o_sys.el_id
  and n_sys.str_part_of = n_app.as_ref
  and o_sys.el_elem_owned_by = o_app.as_ref
  and n_sys.str_part_of != o_sys.el_elem_owned_by;
```

In fact the views *ci_app_sys_<element_type>* should be used, but it is esier to select from *sdd_structure_elements* to get information on many kinds of objects at the same time.

Note that if a new version of an application is generated (and the old one is hereby frozen) then any application that previously inherited objects from the old version, will now inherit the same objects from the new version. Even if the old version is *Unfrozen* again. Fair enough!

By the way - if the application is frozen - do not generate modules with the flag *Update Repository* set or you vill get an error - and no modules generated.

7.8 Checkout Objects

It is possible to check out objects from the repository to a flat file (in ascii Designer/2000 loader format), and install and change these objects on an other system, and then to bring them back again.

Note that these ascii flat files could be the source from a source control system, with a more refined mechanism than just to do versioning on the whole application.

A User Defined Set (UDS) is generated to contain all the primary elements subject to a checkout.

Now from the *Check Out* dialog two different kinds of sets can be choosed from, a source and a working set. If source set is choosed a *.rco* file is created, and the elements in question becomes locked. If working set is choosed a *.rci* file is created, and the elements in question does not become locked.

Checking in the appropriate files again, one has to change the default file extension between *.rco* and *.rci*.

If errors occure checking in the checged objects, an user defined set may always be unlocked again using the *Unlock* dialog. The following script will show which objects from an application has been locked, when and by who:

```
select uds.name set_name, uds.locked_by,
  e.el_name object_name, e.el_type_of object_type,
  to_char(m.date_changed,'DD-MON-YYYY HH24:MI:SS') date_changed
from ci_user_defined_sets uds, ci_set_member_items m, sdd_elements e
where uds.application_system_owned_by = <application_id>
 and uds.locked_by is not null
 and m.user_defined_set_reference = uds.id
 and m.element_reference = e.el_id
order by uds.name, e.el_name;
```

7.9 Using load / unload as source for Version Controle

It is not easy to do version control of the individual objects inside the repository in the database, as relational technology does not support this kind of feature.

It is however possible to extract the actual information per object / object group or for an application by *unload*. The actual information will in fact be stored in an ascii file in Designer/2000 loader format.

This makes it possible to use these files as subject for source controle, as well as it makes it possible to restore information using the *load* feature of Designer/2000.

Before unloading remember to select the actual application / objects.

Before loading, remember first to clear the application area, and then to specify the *Insert* option, or the operation may fail.

8. Documentation

A lot of reports from Designer/2000 are already available for developers to extract proper documentation from the repository. Sometimes it is however nessesary to extract such information from the repository directly, in order to be able to integrate with various office automation systems.

The following SQL-statement will extract all modules, and show the appropriate user help text for that module.

```
select m.name module, t.txt_text text
from ci_modules m, cdi_text t
where m.application_system_owned_by = <application_id>
  and m.id = t.txt_ref(+)
  and (t.txt_type = 'CDHELP' or t.txt_type is null)
order by m.name, t.txt_seq
```

9. Diagnoses

9.1 Data Diagnoses

Assume a system is generated and implemented through Designer/2000, Developer/2000 and the Oracle rdbms, and that the system is shipped to a customer. Also assume that the customer after a while gets strange errors back from the system hard to verify at the developer site.

Then it would be nice if a diagnose scriped was shipped with the system, to check if all requirements specified in the Designer/2000 repository is actually met at the customer site. And even more handy if such a diagnose script could be automatic generated from the repository.

It may f.ex. check if (without using constraints) all foreign keys actually exists in the parent tables.

This is done by selecting the information from the Designer/2000 repository, and then format the result into a proper sequence of SQL-statements.

```
select decode(max(x_cc.sequence_number),min(y_cc.sequence_number),
       'select c.rowid, '''||max(fk.name)||' - FK violations from table '||
       max(ct.name)||''' ','Empty Line') a,
      decode(max(x_cc.sequence_number),min(y_cc.sequence_number),
       'from '||max(ct.name)||' c, '||max(pt.name)||
       ' p ','Empty Line') b,
      decode(max(x_cc.sequence_number),min(y_cc.sequence_number),
       'where ','and ') c,
      'c.'||max(x_cc.name)||' = p.'||max(x_pc.name)||
      '(+) and p.'||max(x_pc.name)||' is null' d,
      decode(max(x_cc.sequence_number),max(y_cc.sequence_number),
      ';','Empty Line') e
from ci_foreign_key_constraints fk, ci_table_definitions ct,
    ci_application_systems a, ci_table_definitions pt,
    ci_key_components x_ckc, ci_columns x_cc, ci_columns x_pc,
    ci_key_components y_ckc, ci_columns y_cc, ci_columns y_pc
where a.name = upper('&APP_NAME')
  and version = (select max(version) from ci_application_systems
           where name = upper('&APP_NAME'))
  and ct.application_system_owned_by = a.id
  and fk.table_reference = ct.id
  and fk.foreign_table_reference = pt.id
  and fk.id = x_ckc.constraint_reference
  and x_ckc.constraint_type = 'FOREIGN'
  and x_ckc.column_reference = x_cc.id
  and x_ckc.foreign_column_reference = x_pc.id
  and fk.id = y_ckc.constraint_reference
  and y_ckc.constraint_type = 'FOREIGN'
  and y_ckc.column_reference = y_cc.id
  and y_ckc.foreign_column_reference = y_pc.id
group by ct.name, fk.name, x_cc.sequence_number
order by ct.name, fk.name, x_cc.sequence_number
```

Executed this way:

```
sqlplus -s $D2KREP_PASS @$DEVENV/D2K/diag APP_NAME | grep -v '^Empty Line$' > script
```

Now the *script* file may look like this:

```
select c.rowid, 'ADVIS_ADVIS_SYS_FK - FK violations from table ADVIS'
from ADVIS c, ADVIS_SYS_TAB p
where
c.ADVIS_SYS_KOD = p.ADVIS_SYS_KOD(+) and p.ADVIS_SYS_KOD is null
and
c.PNR = p.PNR(+) and p.PNR is null
;
select c.rowid, 'ADV_TAB_ADV_GRP_FK - FK violations from table ADVIS_KOD_TAB'
from ADVIS_KOD_TAB c, ADVIS_KOD_GRP p
where
c.ADVIS_GRP = p.ADVIS_GRP(+) and p.ADVIS_GRP is null
;
```

9.2 Data Definition Diagnoses

It is also important that the actual database objects at the costumer site corresponds to the objects represented in the Designer/2000 repository, or the application may fail with errors very hard to find. The following utility will extract information from the Designer/2000 Repository and generate a datakatalog diagnose script to be shipt with the application system and executed at the customer site to verify that no major differences excist.

First the *get_d2k_%* views have to exists:

```
$ sqlplus -s $D2KREP_PASS @$DEVENV/D2K/cre_d2k
```

Then the Data Catalog diagnose script (SQL*Plus) may be generated:

```
$ sqlplus -s $D2KREP_PASS @$DEVENV/D2K/dd_diag APP_NAME > dd_script
```

9.3 Designer/2000 Repository Diagnoses

From the reports of Designer/2000 it is possible to run a number of diagnose reports from the repository, but more information is needed especially in a bigger development project team, where the group need a tool to identify what goes on where in the repository, and to ensure that any special disipline actions have been followed by the developer.

A script extranting such information from the repository has been implemented and produces directly a html document ready to be looked at by a html-browser. the *no_of_days* parameter is used to tell the script for which period in time a list of changed and modified objects is demanded.

```
$ sqlplus -s repository account @$DEVENV/D2K/d2k_di application id no_of_days >html_file
```

10. Cannot do in Designer/2000

You can allmost create all sorts of forms screen using the Oracle Forms Generator, but some issues are hard to cover and could therefor be done at the forms level. But do not forget to aim at the *100% Generation* goal. Not because of the goal itself, but because it gives the project better possibilities to reimplement issues changed late in the design / implementation period, and helps maintaining the code. Some of these issues are:

- Creating blocks side by side on the canvas.
- Controle the window title if the generated module has more windows. They will be called *<module_name>: Window <n>*.

- Select different fonts for items of different datatypes.
- Creating a master / detail module based on the same table can to some extent be done in Designer/2000, but the relation as such should be done in forms in order to get the appropriate triggers.
- Asking forms to disallow record navigation in a given block can only be controled in forms.
- Creating items not connected to database objects with no derivation for accepting user input for enhanced querying and navigation.
- The Forms Zoom facility is not generated by the generator.
- Some forms properties cannot be controled by Designer/2000, like the *Case Insensitive Query* property, and the *enable* field property. The later is nessesary to switch when derived fields are to be horisontally scrolled in order to view the result.

10.1 Tricks for 100% Generation

Remember that *100% Generation* is a goal - not an existent tecknique within Oracle Forms 4.5, but some of the tricks to use, will be polished in later versions of Designer/2000.

Some of these tricks are:

- Put more information into the data definition area of Designer/2000, such as Constraints, Domains, Foreign Keys, ...
- Remember that templats are static in the sence that when a form is generated on a template, the template is copied into the form, and that later changes in the form does not by default change the form.
- You may however externalize code by using program libraries from the template, as the generated forms will then use the same set of libraries, where changes are reflected instantly in the forms. Also note that the use of libraries, helps to keep the use of memory usage in the form as low as possible.
- Implement an event trigger broker, where almose every generated trigger will actually call a library procedure with the trigger name as parameter, allowing for centralized changes if needed.

- Note, that some of the Forms properties not to be managed by Designer/2000 may in fact be automatically set at runtime in a user-written pre-form trigger.
- Also note that Designer/2000 has a way to insert repository information into the gewnerated form, where it may be picked up an used to do same last minute changed at runtime, before the user actually starts using the form.

11. Generating Applications to a Replicated Environment

At the first glanse there should be no problem running the generated Developer/2000 application in a distributed replicated database environment instead of on a single database.

There are however a number of issues to deal with if one wants to forsee some of the challenges moving a generated application on to a replicated environment.

This section does list some of this issues.

11.1 Naming all Constraints

In a replicated environment all replicated objects must have unique names across the entire network - inclusive all constraints. So any database constraint should have a dedicated name and not the local generated one (SYS_nnnnn). This is not a problem to Designer/2000 except for all the *not null* constraints, where the DDL generator of Designer/2000 release 1.2 does not support explicit names. So the generated DDL scripts would have to be modified issuring exact names for *not null* constraints.

11.2 Using Sequence Generators

It is quite common in administrative systems to assign a sequence generator to a primary key of a table, in order to allow an esier change of values otherwise taken to be identifying an object.

Now a sequence generator is local to a database (unless you want to establish a central sequence generator service), so replicating an object from one database to an other may not work well if the other database already uses the actual number for an other object.

It is possible to create the sequence generators in such a way that different numbers are generated on each sute of the system, and it is possible to tell

the replication service that an alternate set of columns should be used for checking if a transported object is already at the target database.

Anyway, the designer should know exactly the purpose of each proposed sequence generator, abd should supply alternate keys for all the tables where the primary key is based on a sequence generator.

11.3 Time Management

Time is a difficult issue in a distributes system, at it is hard to syncronize time across nodes in a WAN. In a replicated environment one would have to accept time information within some tolerance limits.

The designer would have to take care of the actual precision needed in all usage of time information across the entire system, in order to choose a reasonable implementation strategy.

Also allowing nodes to be in a different time zone does influence the implementation, as time information would have to be normalized in order to be used in the database system.

11.4 Delete Operations

In replicated environments one would like to implement a delete operation as a change of the record status from active to deleted or out-dated, and then in a later procedure actually delete all records marked for delete.

Using the Designer/2000 for this issue is not a trivial task, as one should deside the level of information to exist within the Designer/2000 repository. Whether the actual mechanism of how records are deleted in the system should be implemented through Designer/2000 or kept apart in a post design stage is a more political decision. Any way be consistent here.

11.5 Database Limitations

Keep in mind that replication is a relatively new arena, not populated yet by all creatures known from our existing database.

So if you do have a strong fealing that the application you are designing would enter in a replicated environment some day, you should limit the use of *Long* datatypes and clusters.

11.6 Table Names

Observe that when updatable snapshots are used, tables are automatic generated in order to hold the actual values. These tables will take the name *SNAP$_<table_name>*, so do not use table_names which are not unique within the first 24 characters.

11.7 Updatable Snapshots

Updatable Snapshots are indeed very handy for partitioning data among a number of nodes, but bear in mind that changes to an updatable snapshot will be visible even if the change is not yet validated by the master site. And that referential ingetrity rules cannot be enforced directly on the snapshots as they are implemented as views. Implementing the constraints on the tables behind the snapshots cannot be recommented either, as there is no automatic way to be sure that new parent records are inserted before any child records from the snapshot master site.

Thking this into account you should put as much validation as possible on the client side as well.

12. Debugging Designer/2000

From the OCSIG group we have got the following excelent hint:

DES2K Troubleshooting Techniques:

Level Description & Use

0 Normal tool operation; no diagnostic output.

1 SQL Trace - initiate via *alter session set sql_trace = true.*

2 Emit each sql statement as number for finding relevant code.

3 Emit value of bound variables of relevant sql statement.

4 Execution thread (like NARATE=Y).

5 Execution thread displaying actualfunction arguments.

For example, to enable level 3 diagnostics for the process modeler you might have to set the following in oracle.ini

DES2_BPMOD_DIAG_LEVEL=3
DES2_BPMOD_TRACE_FILE=c:\temp\bpr.txt

oracle.ini variables per DES2K component

Process Modeller DES2_BPMOD_DIAG_LEVEL
Entity Realtionship DES2_SYSMOD20_ERD_0DIAG_LEVEL
Function Heirarchy DES2_SYSMOD20_FHD_DIAG_LEVEL
Dataflow DES2_SYSMOD20_DFD_DIAG_LEVEL
Matrix Diagrammer DES2_REPADM10_MD_DIAG_LEVEL
Module Data DES2_SYSDES10_MDD_DIAG_LEVEL
Module Logic DES2_SYSDES10_MLN_DIAG_LEVEL
Preference Navigator DES2_SYSDES10_PN_DIAG_LEVEL
Data Diagrammer DES2_SYSDES10_DD_DIAG_LEVEL
Module Structure DES2_SYSDES10_MSD_DIAG_LEVEL
Forms Generator DES2_CGENF45_DIAG_LEVEL
Reports Generator DES2_CGENR25_DIAG_LEVEL
Repository Object Nav DES2_REPADM10_RON_DIAG_LEVEL
Repository Administer DES2_REPADM10_RAU_DIAG_LEVEL
Repository Reports DES2_REPADM10_REP_DIAG_LEVEL
Repository Utilities DES2_REPADM10_UTL_DIAG_LEVEL
Repos Load/Unload/Check DES2_REPADM10_LUL_DIAG_LEVEL
Visual Basic Generator DES2_VBGEN10_DIAG_LEVEL
C++ Generator DES2_CPPGEN10_DIAG_LEVEL
DB Design Wizard DES2_DATWIZ55_DIAG_LEVEL
Common DLLs DES2_COMALLS10_DIAG_LEVEL

Good luck and please use it with caution.

13. Acronyms

ADW	Application Design Wizard
DCU	Detailed Column Usage
DDW	Database Design Wizard
DD	Database Diagrammer
DFD	DataFlow Diagrammer
DTU	Detailed Table Usage
ERD	Entity Relation Diagrammer
FHD	Function Hierarchy Diagrammer
MD	Matrix Diagrammer
MDD	Module Data Diagrammer
MDI	Multiple Document Interface (Used by MicroSoft)

MIS	Management Information System
MLN	Module Logic Navigator
MSD	Module Structure Diagrammer
OFG	Oracle Forms Generator
ORG	Oracle Reports Generator
OSG	Oracle Server Generator
PN	Preferences Navigator
PM	Process Modeller
RAU	Repository Administration Utility
RON	Repository Object Navigator
RR	Repository Reports
SG	Server Generator
VBG	Visual Basic Generator

CONTENTS

Dansk Data Elektronik A/S
Herlev Hovedgade 199
DK 2730 Herlev
Tel.: (+ 45) 42 84 50 11
Fax: (+ 45) 42 84 52 20