

Oracle Designer 10g Standards & Guidelines



Ministry of Community and Rural Development

Ministry of Tourism, Culture and the Arts

(CD/TCA)

Revision History

This section lists the various versions or releases of the document.

Date	Version	Description	Author
2002-NOV-01	0.1	Initial Draft (Analysis Phase only)	Systems By Design Inc.
2002-NOV-06	0.2	Preliminary feedback from MD	Systems By Design Inc
2002-NOV-12	0.3	Incorporated Design Phase section	Systems By Design Inc
2002-NOV-14	0.4	Incorporated Build, Maintenance sections	Systems By Design Inc
2002-NOV-15	0.5	Corrected formatting errors, included appendices, labeled 'DRAFT', removed in-line comments and highlighting	Systems By Design Inc
2002-NOV-19	0.6	Incorporated Analysis Phase comments (up to 4.2.3) from MD and RG	Systems By Design Inc
2002-NOV-23	0.7	Incorporated remainder of Analysis Phase comments from MD and RG	Systems By Design Inc
2002-NOV-24	0.8	Incorporated Design/Build Phase comments from MD and RG	Systems By Design Inc
2002-NOV-24	0.9	Added 'Req?' column in property tables to replace 'mandatory/optional' text	Systems By Design Inc
2002-NOV-25	1.0	In-line comments removed, in preparation for DRAFT release	Systems By Design Inc
2003-JAN-12	1.1	Edits as per Dec 17, 2002 Meeting	Systems By Design Inc
2003-FEB-12	1.2	Minor Edits	Systems By Design Inc
2003-FEB-26	1.3	Addition of "SQL Tuning" section (5.2.20)	Systems By Design Inc
2003-MAY-26	1.4	Addition of 2.0 Security Access	M Dixon
2003-JUN-11	1.5	Addition of Legend to Appendix A	M Dixon
2003-JUN-18	1.6	Addition of Appendix D	Systems By Design Inc.
2003-JUL-29	1.7	Addition of Appendix E	Systems By Design Inc.
2004-MAR-12	1.8	Incorporated comments from RG and BS	Systems By Design Inc.
2006-FEB-01	1.9	Update to reflect change to Ministry name(s) and upgrade to Oracle Designer 10g.	M. Bird and R. Gretchen
2006-JUL-26	1.10	Updated Section 6.2.7 and created Section 7.2.2.1 (System Level Roles) and Section 7.2.2.1 (Application Level Roles) for Oracle 10g Application Security Policy for shared 10g DB environment Added section 6.2.16.1 PL/SQL Best Practices and section 6.2.20.4 Embedding of SQL in PL/SQL Code, and added Section 7.2.6 PL/SQL modules.	R. Gretchen
2006-SEPT-21	1.11	Added new grant to APP_SCHEMA	K. Warnes
2010-MAY-19	1.12	Update Ministry names, change "ACIM" references to "OCIO", and update DAF urls. Update to Section 5.2.1 to highlight standard that ERD must be in 3NF.	M. Bird

Table of Contents

1	Introduction	1
1.1	Target Audience	1
1.2	Purpose.....	1
1.3	Assumptions.....	1
2	Security Access Policy	2
3	Definitions	3
3.1	Guideline.....	3
3.2	Standard	3
3.3	Related Definitions.....	3
4	Application System Properties	4
4.1	Application Naming	4
4.2	Electronic Delivery of the Application	5
5	Analysis Phase.....	6
5.1	Overall Guidelines	6
5.1.1	Referencing Objects in Text Descriptions	6
5.1.2	Structured Notes	6
5.2	Entity Relationship Modeling	6
5.2.1	Objectives	7
5.2.2	Deliverables.....	7
5.2.3	Entities.....	14
5.2.4	Attributes	17
5.2.5	Standard Entity Enhancements	19
5.2.6	Relationships	19
5.2.7	Domains.....	20
5.3	Functional Modeling	22
5.3.1	Objectives	22
5.3.2	Deliverables	22
5.3.3	Functions	24
5.4	Business Processing Modelling.....	25
5.4.1	Objectives	26
5.4.2	Deliverables.....	26
5.5	Business Areas	27
5.5.1	Deliverables.....	27
5.5.2	Business Units	27
5.6	Business Rules Modelling.....	28
6	Design Phase	34
6.1	Overall Guidelines	34
6.1.1	Referencing Objects in Text Descriptions	34
6.1.2	Keeping logical data model current.....	34
6.1.3	Electronic Delivery of the Application	34
6.2	Database Design.....	34
6.2.1	Objectives	35
6.2.2	Deliverables.....	35
6.2.3	Object Naming Conventions.....	37
6.2.4	Database Design Transformer	38
6.2.5	Standard Table Enhancements.....	40
6.2.6	Journal Tables.....	41
6.2.7	Databases.....	42
6.2.8	Tablespaces.....	44
6.2.9	Datafiles.....	44
6.2.10	Tables	44
6.2.11	Columns.....	46
6.2.12	Views.....	49
6.2.13	Sequences	50

6.2.14	Constraints	50
6.2.15	Indexes.....	54
6.2.16	PL/SQL Definitions.....	55
6.2.17	Storage Definitions	60
6.2.18	Synonyms	60
6.2.19	Database Object Grants	60
6.2.20	SQL Statement Tuning	62
6.3	Module Design.....	62
6.3.1	Objectives	64
6.3.2	Deliverables.....	64
6.3.3	Module Naming Conventions.....	65
7	Build Phase.....	66
7.1	Overall Guidelines	66
7.1.1	Referencing Objects in Text Descriptions	66
7.1.2	Keeping logical data model current.....	66
7.1.3	Documenting Post-Generation Changes.....	66
7.2	Implementation of Database Objects	67
7.2.1	Users	68
7.2.2	System and Application Roles.....	68
7.2.3	Table Implementations	70
7.2.4	Sequence Implementations	71
7.2.5	User Object Index Storages	71
7.2.6	PL/SQL Modules.....	71
7.3	Updating Bound Columns in Modules.....	72
7.4	Preferences	72
7.4.1	Objectives	73
7.5	Code Tables.....	73
7.6	Designer Generated Reference Codes - REF_CODES	73
8	Maintenance Phase	74
8.1	Overall Guidelines	74
8.1.1	Synchronizing Table Definitions.....	75
8.1.2	Synchronizing View Definitions	75
8.1.3	Synchronizing Domain Definitions	75
8.1.4	Synchronizing Display Information / Comments / Help Text	76
8.1.5	Synchronizing Entities with Tables	77
8.1.6	Synchronizing Module Definitions.....	78
8.1.7	Electronic Delivery of the Application	80
9	Designer Generation	81
9.1	Generate Database from Server Model	81
9.1.1	Post-Generation Changes.....	85
9.1.2	Reconcile Report	85
9.1.3	Capture Design of Server Model	85
9.1.4	Capture Design of Supporting Tables.....	86
10	Repository Extensions.....	88
11	Summary	89
12	Appendices.....	90
12.1	Appendix A – Summary of Deliverables.....	91
12.2	Appendix B – Glossary of Terms	93
12.3	Appendix C – Standard Approved Abbreviations	94
12.3.1	Mandatory Abbreviations	94
12.3.2	Preferred Abbreviations.....	94
12.4	Appendix D – Developer Guidelines.....	96
12.5	Appendix E – Developer SCM Guidelines.....	97

Table of Figures

Figure 1: Analysis Phase QA Checklist.....	8
Figure 2 - Master ERD Example composed of colour coded Subject Areas entities with the Primary UIDs	9
Figure 3 - Subject Area ERD Example – no external Subject Area entities	10
Figure 4 - Subject Area ERD Example – with external “Enterprise Participant” Subject Area entity	11
Figure 5 - Subject Area ERD Example - with external “Enterprise Participant” and “Contract” Subject Areas entities	11
Figure 6: Function Hierarchy Diagram.....	23
Figure 7: Business Rules Hierarchy	29
Figure 8: Tuple Rules Hierarchy	32
Figure 9: Design Phase QA Checklist	36
Figure 10: DDT Settings - Database.....	38
Figure 11: DDT Settings - Keys	39
Figure 12: DDT Settings - Other	40
Figure 13: Journal Table.....	41
Figure 14: Database Package.....	58
Figure 15: Role Security.....	61
Figure 16: Column Properties.....	63
Figure 17: DB Object Implementation	67
Figure 18: Generator Options	74
Figure 19: Reference Code Table Scope	74
Figure 20: Recreate Domain.....	76
Figure 21: Database Design Transformer Options	77
Figure 22: Entity Retrofit	78
Figure 23: Capture Forms.....	79
Figure 24: Generate Server Model Options.....	82
Figure 25: Database Generator Options	83
Figure 26: Generate Server Model Objects	84
Figure 27: DDL Generation.....	84
Figure 28: Reconcile Report.....	85
Figure 29: Capture Server Model	86
Figure 30: Journal Tables reverse engineered	87

1 Introduction

The intent of this document is to describe the guidelines and standards to be followed when designing and developing Oracle Designer applications at the Ministry of Community and Rural Development and the Ministry of Tourism, Culture and the Arts. This document is not intended to be an 'all inclusive' guide on the use of Oracle products.

Originally the Designer Standards for the Ministry of Sustainable Resource Management, this document has been taken, with permission from that Ministry, and modified to suit our unique requirements.

As with any standards document, this document will evolve over time. It is fully expected that each and every development effort will contribute to the evolution of this document.

1.1 Target Audience

This document is directed at those who will be designing, developing and maintaining Oracle application systems for the Ministry of Community and Rural Development and the Ministry of Tourism, Culture and the Arts. This includes external contractors, consultants, and business partners, as well as ministry employees (Data Administrator, Database Administrator, Business Analyst and Application Analysts).

1.2 Purpose

Oracle's Designer product provides a central repository for the storage of information about an application throughout its entire life cycle. Utilizing this repository provides for consistent application design and development within the Ministry of Community and Rural Development and the Ministry of Tourism, Culture and the Arts.

This document outlines the standards which must be followed when building application systems using Oracle's Designer tool set.

1.3 Assumptions

As it is not the intent of this document to be an 'all inclusive' guide on the use of Oracle products, it is assumed that the audience has a working knowledge of Oracle's Designer and Developer product set, and relational databases.

Throughout the remainder of the document, the Ministry of Community and Rural Development and the Ministry of Tourism, Culture and the Arts shall be referred to as "The Ministry".

The current release of Designer 10g used by the Ministry is Designer 10g Release 2 (10.1.2.0.2). All developers using Designer 10g to access the ministry repositories must have this release installed for compatibility. Release levels will change over time as Oracle support obsolescence occurs. There will be ample notification to developers prior to any upgrade of the ministries Designer 10g repository versions.

2 Security Access Policy

Audience: ISB Staff, External Contractors, Clients

The Data and Database Administration Group within the Ministry ISB is ultimately responsible for the management, data integrity and security of the ministry's Designer 10g Systems Configuration Management (SCM) repository. Due to the inherent complexities and risks associated with managing metadata within a SCM environment such as Designer 10g, the ISB will restrict "Create/Update/Delete" access only to specific external development resources and select ISB staff. "Read only" access may be provided to other individuals in the ministry if deemed necessary on a case-to-case basis. This policy will be firmly enforced by the ISB.

3 Definitions

3.1 *Guideline*

A guideline is a method or custom, which through common usage has become an accepted method of work. A guideline is not enforced, and is not a standard.

3.2 *Standard*

A standard is a specific statement of the rules and constraints governing the naming, contents, and operations of software. Some statements are in bold, to emphasize standards that have been overlooked in the past.

Unless otherwise noted, every statement in this document is a standard.

3.3 *Related Definitions*

Other relevant definitions can be found at:

4 Application System Properties

The Application System Properties sheet must be filled in for each project. The 'Req?' column refers to the fact that the property must be completed, either due to Designer rules or Ministry Standards, or both.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> specifies the application short name (e.g. LGIS) defined at application creation 	Y
Version	<ul style="list-style-type: none"> all properties in this group are maintained automatically through the versioning function 	Y
Title	<ul style="list-style-type: none"> used as the application title on screens and reports that are generated (e.g. LGIS System) 	Y
Authority	<ul style="list-style-type: none"> describes the person, business or organization responsible for this application should be name of the application custodian 	Y
Owner	<ul style="list-style-type: none"> set automatically; should be the application owner (e.g. LGIS) 	Y
Datawarehouse?	<ul style="list-style-type: none"> identifies this application as one that feeds the Ministry data warehouse 	Y
Documentation		
Priorities	<ul style="list-style-type: none"> recommended describes the priorities on this application 	N
Constraints	<ul style="list-style-type: none"> recommended describes any constraints on this application; not interfaces to other corporate systems 	N
Comment	<ul style="list-style-type: none"> describes any general comments about the application as a whole 	N
Summary	<ul style="list-style-type: none"> contains a summary of the purpose of the application system 	N
Objectives	<ul style="list-style-type: none"> describes the objectives of the application system within the context of the business unit: e.g. <i>LGIS is intended to replace multiple legacy applications (LGDETAX and manual spreadsheets); in addition, enhancements will be needed in response to new business requirements and a changing service model.</i> 	Y
Description	<ul style="list-style-type: none"> contains a brief overall description of the application system 	Y
Notes	<ul style="list-style-type: none"> contains a change history of the application may contain extraneous notes on the application system 	Y

4.1 Application Naming

Applications must be named as a 3-4 character short name or acronym that is unique within the business area or corporation. The expanded name should be recorded in the Title property.

This Application Name will be automatically prefixed to all 'physical' database objects such as tables, views, packages, sequences and roles (see Database Design Transformer). Functions and procedures that are not encapsulated in packages should also be prefixed with this name.

The intent of requiring the prefixing of the Application Name on all objects is to reduce the possibility of namespace collisions. For example, the LGIS application uses LGIS as its short name. Therefore, the SCHEDULE entity becomes the LGIS_SCHEDULES table.

Approval to use a new application acronym must be obtained from the Corporate Data Administrator or Corporate Database Administrator to ensure that there are no duplicate names.

4.2 *Electronic Delivery of the Application*

All development is done directly against the Ministry Repository, and all vendors must perform a specific number of steps at the end of each life-cycle phase (e.g. Analysis Phase). For details of this standard process, please refer to Section 8.4 (Promotion Management Procedures) of the Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

It is assumed that the DWS (Development and Web Services) participants have been involved iteratively throughout the project, for Quality Assurance purposes.

For a complete overview of the Ministry standard Promotion Model, see the Ministry's Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

5 Analysis Phase

5.1 Overall Guidelines

This section presents some overall guidelines to assist in requirements analysis within the Oracle Designer environment.

5.1.1 Referencing Objects in Text Descriptions

Whenever the name of another ENTITY, ATTRIBUTE (or any other object) is used within a textual description, it should be capitalized for easier reading (and reference).

For example, if LICENCE is an entity, then the following description should be used for the LICENCE_TYPE entity:

"This entity identifies the types of LICENCES that are available to the polling system"

5.1.2 Structured Notes

Issues, decisions and notes should be recorded in the Notes property of the relevant object (e.g. entity, attribute, function, table, column, and module). The suggested format is to prefix the text with indicators of

1. What phase of development (Analysis, Design, or Build)
2. What type of note (Question, Point, Answer, or Decision)
3. Date that the issue was raised, or resolved
4. Initials of the analyst who raised this issue

An example is:

```
A? 1998-01-18 GW There may be an opportunity to share this
                    functionality with 'Record contact information
                    about a new permittee/PMP'er/Certificate holder.
```

The notation is:

```
<Phase><Note Type> <Date> <Initials> <Note Text>
<Phase>          is one of A, D, B    (Analysis, Design, or Build)
<Note Type>      is one of ?, !, A, D (Question, Point, Answer, or Decision)
<Date>           is in the format YYYY-MM-DD
```

For examples, see [Functions Notes](#) or [Entity Notes](#).

5.2 Entity Relationship Modeling

Entity Relationship Modeling involves identifying the things of importance in an organization (entities), the properties of those things (attributes) and how they are related to one another (relationships).

The Repository Object Navigator (RON) and Entity Relationship Diagrammer (ERD) tools are used to model entities, their attributes, relationship to other entities, and unique identifiers. They are also used to identify domains, allowable values, and unique identifier components associated with attributes.

It is the intention of Entity Relationship Modelling to produce a data model of the business requirements, not the physical implementation.

5.2.1 Objectives

The objectives of the Entity Relationship Modeling process are:

- To provide an accurate model of the information needs of the organization, which will act as a framework for the development of new or enhanced systems.
- To document the business requirements for data, the specific business rules and relationships that apply to that data.
- To provide a model independent of any data storage and access method, to allow objective decisions to be made about implementation techniques and coexistence with existing systems.
- To provide a blueprint for data storage which ensures data integrity and reduces data redundancy.

It is a Ministry standard that, by the end of the Analysis Phase, the Entity Relationship Model is in Third Normal form (e.g. no non-UID attribute can be dependent upon another non-UID attribute).

Related standards on data modelling are available on the Government of British Columbia's Office of the Chief Information Officer (OCIO) Data Administration Forum (DAF) website (<http://www.cio.gov.bc.ca/cio/standards/daf.page?>).

5.2.2 Deliverables

Although Designer can produce numerous reports and diagrams, only the following set of reports and diagrams are required deliverables for Entity Relationship Modeling. This does not preclude the use of the various analytical and quality assurance reports during the design, development, and review of the components of an application.

The logical data model document to be presented for sign-off will contain the following diagrams and reports:

- [Entity Relationship Diagrams](#)
- [System Glossary Report](#)
- [Entity Definition Report](#)
- [Entities and Their Attributes Report](#)
- [Entity Completeness Checks Report](#)
- [Domain Definition Report](#)
- [Attributes In a Domain Report](#)

The logical data model forms part of the Business Requirements Document.

A checklist is available to confirm that the Analysis Phase is complete and that the repository is ready for the Design Phase. This checklist is used in conjunction with the deliverables stated above:

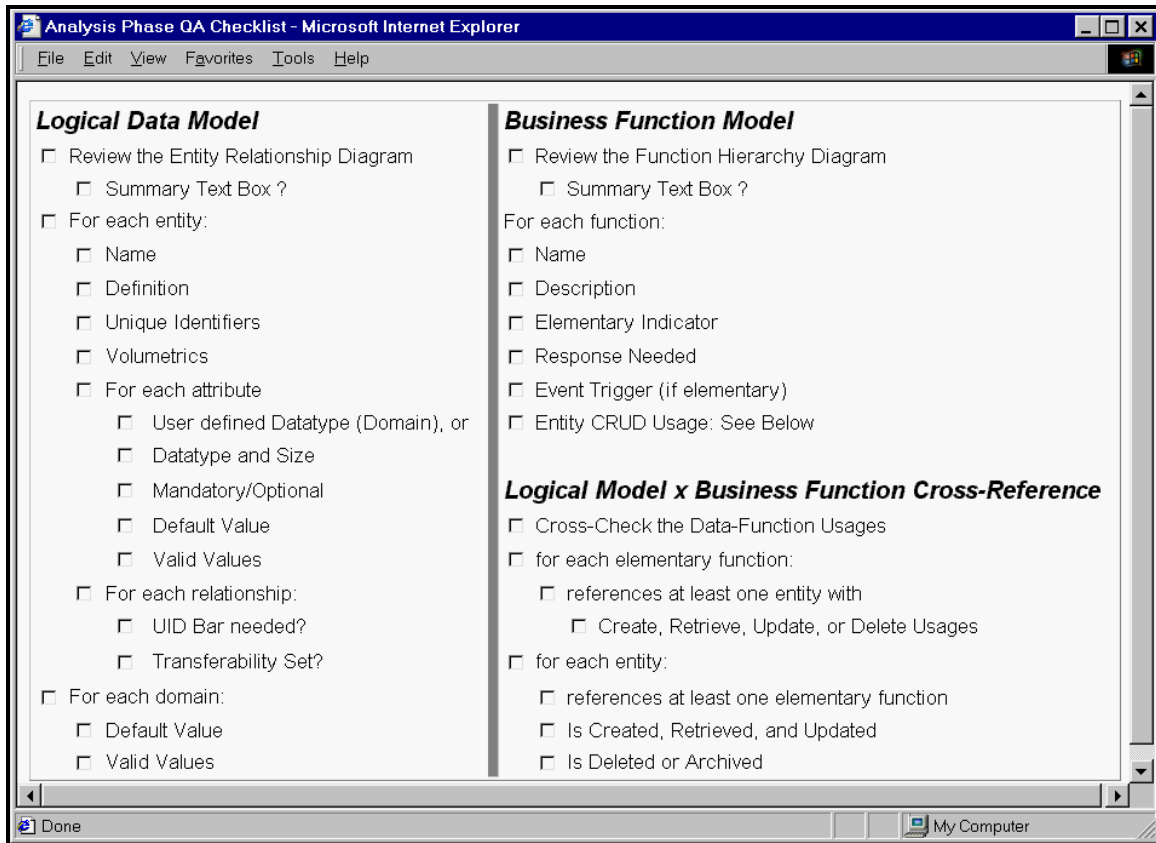


Figure 1: Analysis Phase QA Checklist

Note that Ministry Quality Assurance reviews will reference the Data Modelling standards found on the Government of British Columbia's Office of the Chief Information Officer (OCIO) Data Administration Forum (DAF) website (<http://www.cio.gov.bc.ca/cio/standards/daf.page?>).

5.2.2.1 Entity Relationship Diagrams (ERD)

Entity Relationship Diagrams (ERD's) showing all of the application entities and relationships must be provided.

Master ERD

The Master ERD provides context to a system by presenting a total view of all system entities and their relationships. To facilitate readability and ease of printing, the detailed entity information is presented in Subject Area ERDs.

The Master ERD must contain the following:

- entities with only the Primary UID attributes (if possible). The entities from specific Subject Areas must be colour coded to indicate their origin. The colour code for entities from each Subject Area must be consistent among all diagrams within a system. This requirement provides an effective visual communication of each Subject Area in context of the system.
- a legend describing the colour code for each set of Subject Area entities
- all of the relationships with their descriptions.

Diagram: ERD CONTRACT MANAGEMENT - MASTER
 Title: Contract Management System
 Created: 08 February 2006 14:31:28
 Modified: 15 February 2006 15:06:26
 Author: T. Dixon
 Application System: CLIENT

Subject Area	Colour Code
Enterprise Participant	Yellow
Contract	Green
Contract Payment	Cyan

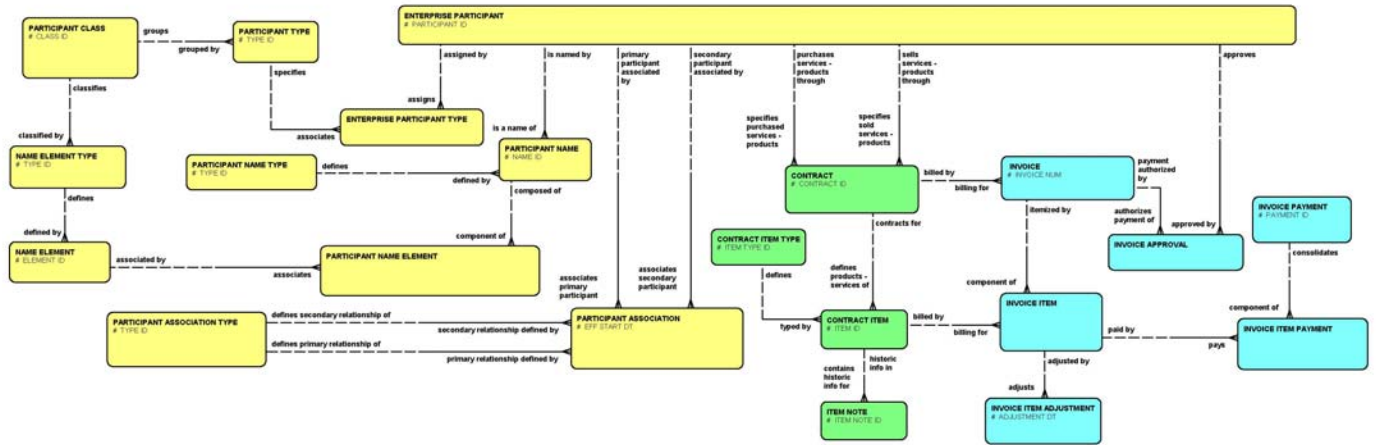


Figure 2 - Master ERD Example composed of colour coded Subject Areas entities with the Primary UIDs

Subject Area ERDs

The Subject Area ERDs provide the required detailed information of all the entities in the system pertaining to a specific business function (e.g. contract payments).

To facilitate readability and ease of printing, a Subject Area ERD must not exceed 15 entities. If there is a business requirement to exceed this maximum, it must first be reviewed and approved by the Ministry DA.

The Subject Area ERD must contain the following:

- entities with all of the attributes including Primary UIDs and Mandatory and Optional attribute indicator symbols
- entities depicted in the diagram must be white in colour. Key entities from external Subject Areas, (which are included to provide context to the Subject Area diagram), must be colour coded to indicate their origin. The colour code for these key entities must be consistent among all diagrams within the system.
- a legend describing the colour code for each set of external Subject Area entities
- all of the relationships with their descriptions.

Diagram : ERD ENTERPRISE PARTICIPANT SUB-SET - SA
 Title : Contract Management System
 Created : 08 February 2006 14:27:15
 Modified : 08 February 2006 14:27:15
 Author : T. Dixon
 Application System : CLIENT

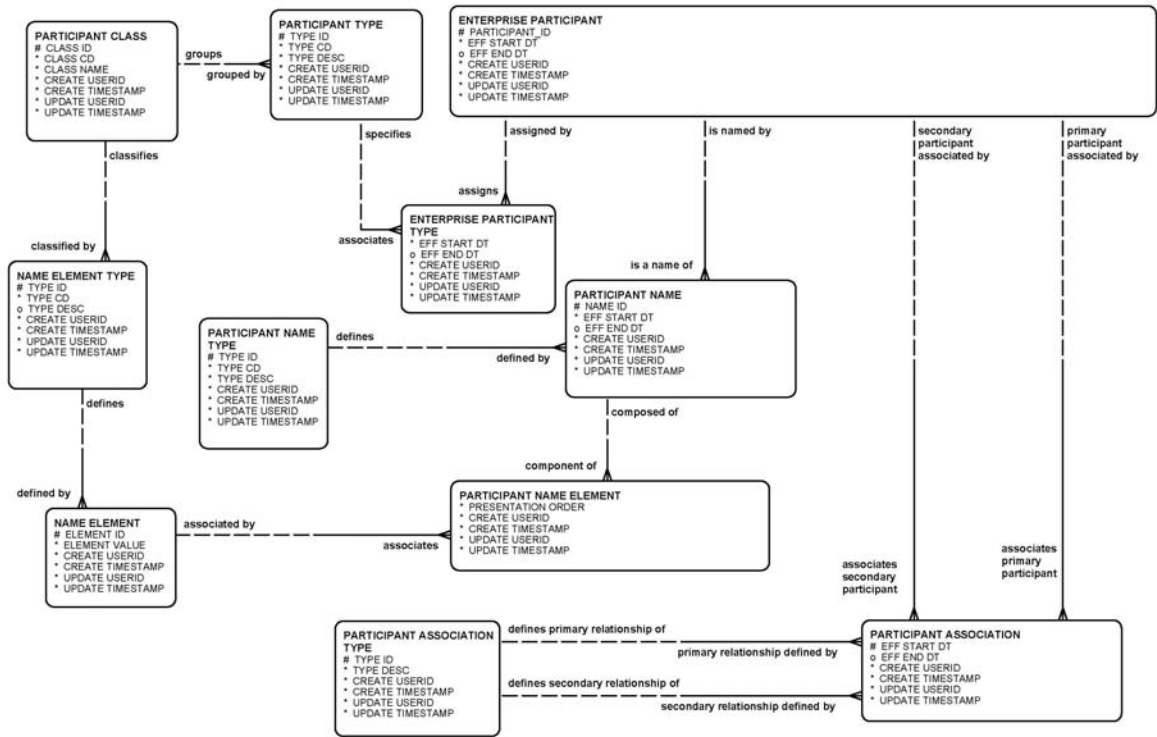


Figure 3 - Subject Area ERD Example – no external Subject Area entities

Diagram : ERD CONTRACT - SA
 Title : Contract Management System
 Created : 08 February 2006 14:27:53
 Modified : 08 February 2006 14:32:55
 Author : T. Dixon
 Application System : CLIENT

External Entities	
Subject Area	Colour Code
Enterprise Participant	

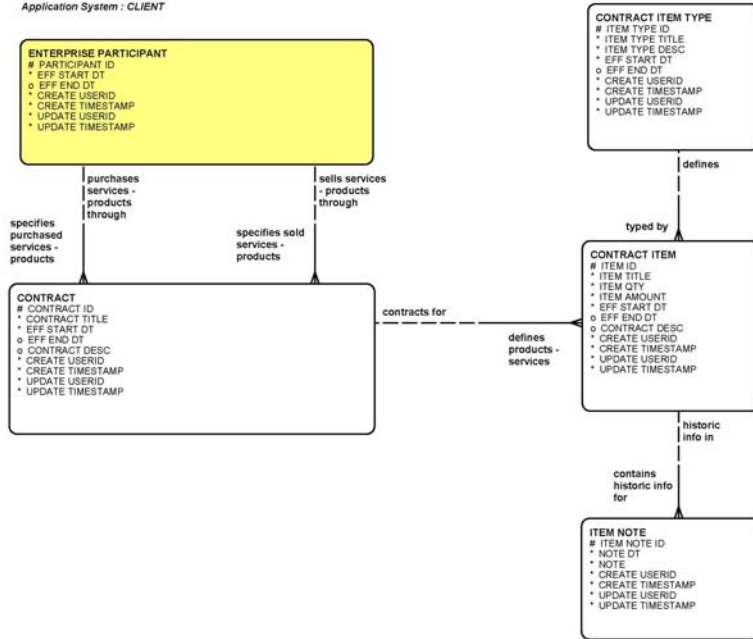


Figure 4 - Subject Area ERD Example – with external “Enterprise Participant” Subject Area entity

Diagram : ERD CONTRACT PAYMENT - SA
 Title : Contract Management System
 Created : 08 February 2006 14:26:32
 Modified : 09 February 2006 14:46:12
 Author : T. Dixon
 Application System : CLIENT

External Entities	
Subject Area	Colour Code
Enterprise Participant	
Contract	

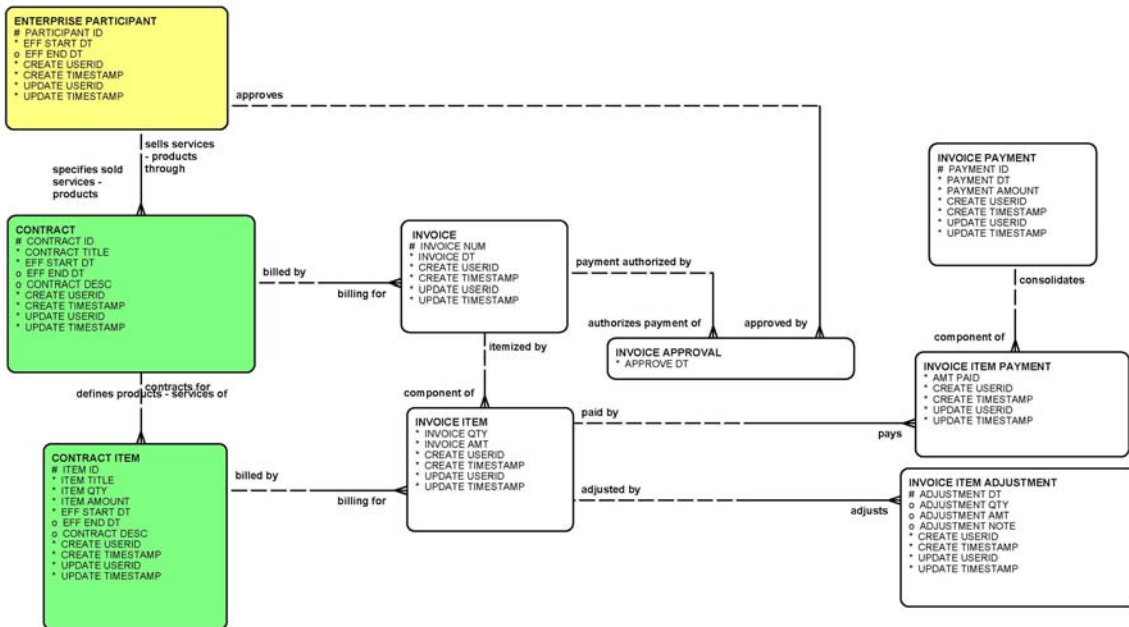


Figure 5 - Subject Area ERD Example - with external “Enterprise Participant” and “Contract” Subject Areas entities

ERD Naming Conventions

The ERD naming conventions are as follows:

- the term “**ERD**” must precede each diagram name
- if it’s a Subject Area ERD, then “- **SA**” must follow each name (e.g. **ERD Contract – SA**)
- if it’s a Master ERD, then “- **Master**” must follow each name (e.g. **ERD Contract Management – Master**)

Diagramming Style

Each ERD must contain the Diagram Summary Information displayed without borders. The Diagram Summary Information must contain the following information as recorded in the Repository:

- the diagram name,
- title (which could be the Application System name if the Container name is not explicit e.g. “Contract Management System” ERD in the “CLIENT” container),
- date and time the diagram was created,
- date and time the diagram was last modified,
- the author,
- the application system name (i.e. CONTAINER name).

A consistent diagramming style should be used throughout the ERD; a recommended style is to diagram master (Independent) entities above the detail (Dependent) entities they are related to. When utilizing this style, all relationships are drawn as lines with the *many* end of *one-to-many* relationships appearing at the bottom of the relationship line and to the right. Using a consistent style improves the readability of the diagram and makes it much easier to identify potential problems in the model.

ERD Visual Check List

A visual check of the ERD would include the following items:

- Diagram Summary Information including:
 - Diagram Name in the format defined above
 - Title,
 - Date & Time Created,
 - Date & Time Last Modified,
 - Author,
 - Application System
- Entity boxes line up, and relationship lines are mainly straight and horizontal or vertical (*many* end at bottom or right of relationship line)
- All text is unambiguous - jargon and abbreviations have been avoided
- The relationship names are easy to read. This implies that the names are:
 - horizontally orientated
 - on opposite sides of the lines next to the entity to which they refer such that they may be read in a clockwise fashion
 - not overlapping.
- If colour is used to enhance the readability of an ERD, a legend describing the colour code for each set of subject area entities must be included in the diagram. The legend may be created using a tool such as Microsoft Excel and then inserting it as an object in the ERD.
- The diagram is presentable, with legible elements and no crossing lines where possible
- The diagram reflects the business accurately as validated by business users
- The diagram can be effectively used to describe data to all interested participants.

5.2.2.2 System Glossary Report

- Entity names should be singular
- Entity names should be meaningful and the use of abbreviations should be kept to a minimum. A typical entity name is a noun
- A standard list of abbreviations can be found in [Appendix C – Standard Approved Abbreviations](#), and should be used wherever possible
- All entities must have a clear business description. The description must explain what the data is to non-application personnel (e.g. Data Administration)
- Descriptions for abstract entities should contain concrete examples
- All references to other objects should be capitalized.

5.2.2.3 Entity Definition Report

- Entity names must be singular
- Entity names must be meaningful and the use of abbreviations should be kept to a minimum. A typical entity name is a noun
- A standard list of abbreviations can be found in [Appendix C – Standard Approved Abbreviations](#), and should be used wherever possible
- All entities must have a clear business description. The description must explain what the data is to non-application and non-technical personnel

Note: It is a Ministry standard that the business area expert(s) (i.e. client representative, business analyst, and data administrator) review and approve these descriptions

- Descriptions for abstract entities should contain concrete examples.
- All references to other objects should be capitalized.
- Where applicable, use should be made of Oracle's support of sub-type entities and domains.
- All super-type entities must have a unique identifier.
- All sub-type entities must have at least one relationship or attribute different from their super-type.
- All sub-type entities must be mutually exclusive
- *Many-to-many* relationships must be resolved with an intermediate entity.
- Relationship names must be meaningful and both sides of a relationship must be named. It is helpful to consider the relationship name in the context of a sentence as follows:

```
EACH ENTITY1 MUST BE/MAY BE relationship ONE AND ONLY ONE/ONE OR  
MORE ENTITY2
```

For example:

```
EACH STUDENT MUST be enrolled in ONE OR MORE CLASSES  
EACH CLASS MAY BE comprised of ONE OR MORE STUDENTS
```

Note: The Entity Model Reference Report has a 'Relationships' section where you may check the relationship wording.

5.2.2.4 Entities & Their Attributes Report

Any attribute where the attribute name does not effectively describe the nature of the attribute must have an associated note. An example would be an attribute name that would exceed the 30-character limit, if fully descriptive.

Note: It is a Ministry standard that the business area expert(s) (i.e. client representative, business analyst, and data administrator) review and approve these attributes and associated elements.

5.2.2.5 Entity Completeness Checks

Any entity that appears on this report should have justifications documented in the Entity Notes. An example is an intersection entity, which has no attributes. The checks are:

- No Attributes
- No Description
- No Unique Identifiers
- With No Relationships
- Not Used by any Functions

5.2.2.6 Domain Definition Report

All attributes should be placed under a domain. This Report lists all the domains and their descriptions. There are currently no Ministry standard domains, so application-specific ones may be defined. Domains must be reviewed and approved by ministry Data Administrator

Note: The Ministry is currently reviewing its COMMON set of domains.

5.2.2.7 Attributes in a Domain Report

There are benefits to creating and using application-specific domains wherever an attribute is used in more than one entity. If this approach is taken, it is easier to ensure that datatype mismatches between entities/tables are avoided, and that any changes to the datatype can be made at the domain level, and then flushed throughout the application.

It is therefore a Ministry standard that extensive use of application-specific domains be used for every attribute/column.

5.2.3 Entities

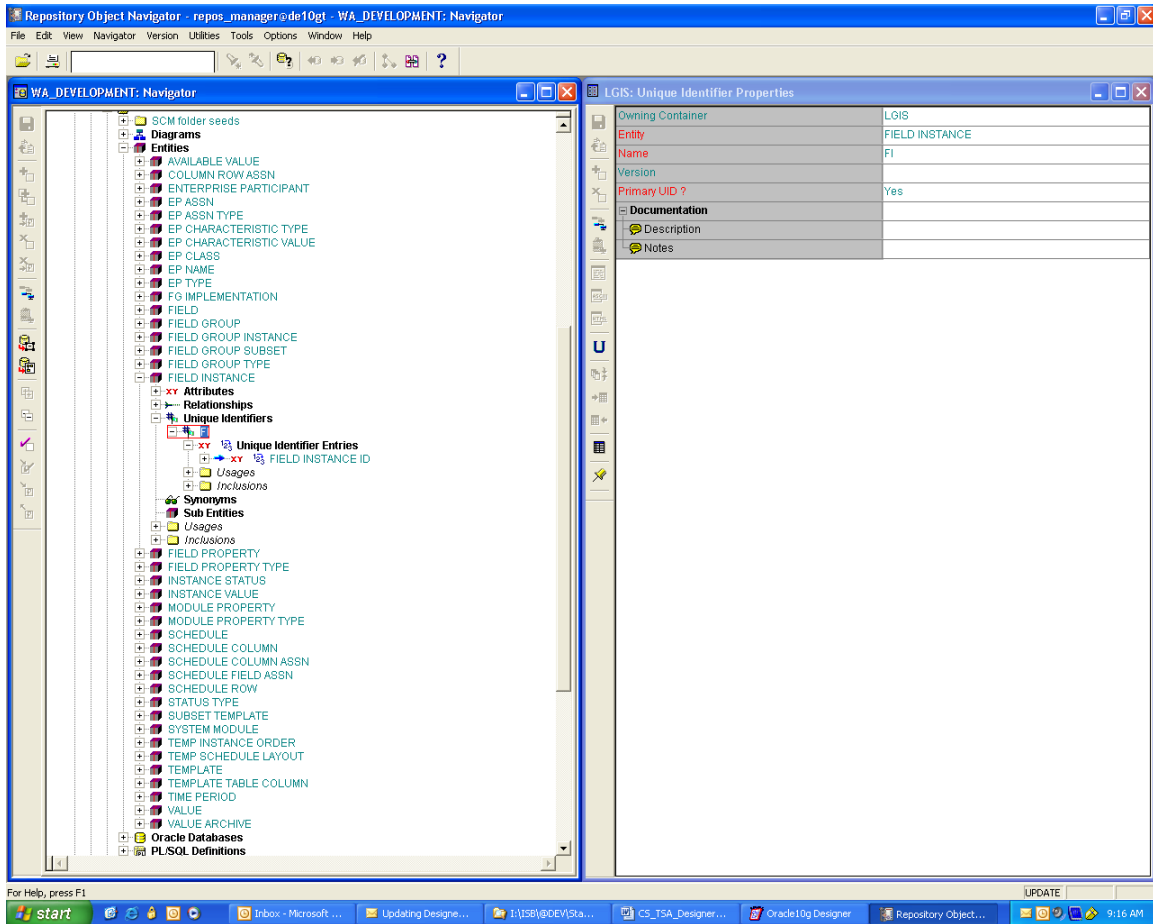
An entity is a thing of significance about which information needs to be known or held.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> • Should uniquely identify the entity in a manner that is easily understood by the business people • should be made up of one to three real words, with no ambiguity in meaning • abbreviations should be avoided unless they are obvious • when the entity name is only one word, it must not be an Oracle Reserved Word • use a singular noun and any modifiers • contains only alphabetic characters and spaces • does not contain hyphens or underscores <p>Note:</p> <ul style="list-style-type: none"> • cross-reference entity names must be suffixed with _XREF • code lookup entities must be suffixed with _CDS • see OCIO's Data Administration Forum (DAF) website (http://www.cio.gov.bc.ca/cio/standards/daf.page?) 	Y
Short Name	<ul style="list-style-type: none"> • an entity identifier, up to 10 characters in length • when the entity short name is only one word, it must not be an Oracle 	Y

Property	Rule	Req?
	<p>Reserved Word</p> <p>Note: The <i>Entity Short Name</i> becomes the default <i>Table: Short Name</i> when the <u>Database Design Transformer</u> is used.</p>	
Plural	<ul style="list-style-type: none"> should follow the standards for the Entity Name (no underscores, hyphens, etc) there are cases where the name is already plural (e.g. SHEEP), so this must be corrected manually <p>Note: The <i>Entity Plural</i> becomes the default <i>Table: Name</i> when the <u>Database Design Transformer</u> is used.</p>	Y
Volumes	<ul style="list-style-type: none"> These fields are the initial estimates for the quantity of data, and provide estimates for the annual growth in information. These data will eventually be used in the sizing estimate algorithm. <p>Note: It is important that some thought, effort, and calculation be put into these estimates by actively soliciting this information from the business experts.</p>	
Initial	<ul style="list-style-type: none"> expected number of records in the entity (table) when the system first goes into production 	Y
Maximum	<ul style="list-style-type: none"> expected number of records in the entity (table) at the end of the third year 	Y
Average	<ul style="list-style-type: none"> average number of expected records in the entity (table) at the end of the third year 	Y
Annual Growth Rate	<ul style="list-style-type: none"> expected % growth rate for the entity (table) per year 	Y
Documentation		
Description	<ul style="list-style-type: none"> must explain what the entity is to non-application personnel descriptions for abstract entities should contain concrete examples can be further categorized into Definition, Example and Miscellaneous, such as: <p>DEFINITION A type of role that a party may play in the context of a permit, PMP, license or certificate.</p> <p>EXAMPLE Examples of role types are 'Located Within' (i.e. Region or regions that the PMA, License or Certificate are physically located or affect in the case of items that cross regional boundaries), 'Administering Region' (i.e. the region actually responsible for a permit. PMP, License or certificate), 'Consultative' and 'Contact'.</p> <p>MISCELLANEOUS Note that Administering Region may include HQ, as HQ staff may perform the hands-on administration of a permit, PMP, license or certificate.</p> <p>This entity is used to hold the valid list of Approval Roles that are available for use in CRISP. This list is only to be modified by the Application Manager, and then, only when adding new functionality to the system.</p>	Y

Property	Rule	Req?
	<p>Recently, we've added 'Approved Training Agency', which will now share the duties of issuing Certificates with 'Administering Region' .</p> <p>Note: The <u>Database Design Transformer</u> copies this information into the User / Help Text and Description fields.</p>	
Notes	<ul style="list-style-type: none"> • should contain any notes about this entity • structured analysis or design comments should be placed here, for example: <p>OUTSTANDING ISSUES =====</p> <p>A! 1998-01-21 GW GENERAL LOCATION and all information specifically related to time/place details have been deleted (R. Adams, 1998-01-19) from EXAMINATION entity and the entire entity has been deleted.</p> <p>IMPLEMENTATION NOTES =====</p> <p>MISCELLANEOUS =====</p> <p>Note: The <u>Database Design Transformer</u> copies this information into the Notes field.</p>	N

Although unique identifiers are usually entered via the Entity Relationship Diagrammer, they also show up in the RON, under the following sub-node.



Node	Rule	Req?
Unique Identifiers	<ul style="list-style-type: none"> must be comprised of attribute(s) and/or relationship(s) that are defined for the entity this is to ensure business uniqueness 	Y
Primary UID?	<ul style="list-style-type: none"> indicates that this UID is the primary key 	Y

Note: UID's can be either a business key (with meaning, such as Name) or a system (a surrogate meaningless value, such as a number). When using System UID's, the underlying business keys are still recorded in Designer as Secondary UID's.

5.2.4 Attributes

An attribute is a thing of significance that serves to classify, quantify, qualify, identify or express the state of an entity.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> should be made up of one to three real words when an attribute name is only one word, it must not be an Oracle Reserved Word should be singular and contain no hyphens or underscores if generating Forms modules, then the length should be 22 characters or less, due to a Forms Generator bug 	Y

Property	Rule	Req?
	Note: The <u>Database Design Transformer</u> uses this to generate the default Name and Prompt for the column.	
Sequence in Entity	<ul style="list-style-type: none"> will become the sequence of the columns in the table (see <u>Database Design Transformer</u>) optional attributes should be after mandatory ones long character attributes should be next large binary attributes (e.g. video, sound) should be next audit attributes (create_userid, create_timestamp, etc.) must be last 	Y
Domain	<ul style="list-style-type: none"> name of the domain (if a domain is used) 	
Definition	<ul style="list-style-type: none"> if a domain is not used, then the following fields must be completed to define the datatype if a domain is used, then these fields will be populated automatically 	
Format		Y
Average Length		Y
Maximum Length		Y
Decimal Places	<ul style="list-style-type: none"> mandatory for numeric attributes 	N
Optional?	<ul style="list-style-type: none"> NO means that the attribute is required (Not null) YES means that the attribute allows null values 	Y
Units	<ul style="list-style-type: none"> Used for documentation purposes only Defined the unit of measure for the attribute (metres, kilograms, ppm) 	N
Default	<ul style="list-style-type: none"> Should not be used if an attribute is optional Must be the same datatype as the attribute <p>Note: Use of defaults must be examined carefully, as default values may lead the inexperienced user to enter erroneous data</p>	N
Derivation	<ul style="list-style-type: none"> algorithm or expression how the attribute's value is derived further explanation of this derivation must be qualified in the Attribute Notes property 	N
Volumes	<ul style="list-style-type: none"> These fields contain estimates for the quantity of data, and will eventually be used in the sizing estimate algorithm <p><i>It is the Ministry standard to enter appropriate values for these fields</i></p>	
Percent Used - Initial	<ul style="list-style-type: none"> should be specified for documentation purposes if an attribute is not optional, then will be set automatically at 100% 	Y
Percent Used - Average	<ul style="list-style-type: none"> if an attribute is not optional, then will be set automatically at 100% specifies the average percent of the attributes that contain values 	Y
Documentation		
Comment	<ul style="list-style-type: none"> For example: Indicator that a stay is currently on this approval <p>Note: The Database Design Transformer uses this to generate the <i>Display: Hint</i> and <i>Comments</i> for the column.</p>	Y
Description	<ul style="list-style-type: none"> must be described from the user's perspective and in plain English provide examples where possible 	Y

Property	Rule	Req?
	<ul style="list-style-type: none"> should further expand upon the attribute comment, for example: A 'Stay' is an order issued by the Environmental Appeal Board, a temporarily suspending the approval, pending a decision on an appeal. This stay can affect only a certain portion on the approval. <p>Note: The Database Design Transformer uses this field to populate the Text: Description and Text: User Help Text fields for the column.</p>	
Notes	<ul style="list-style-type: none"> any additional notes for the attribute 	N

Although Allowable Values are usually entered via the Entity Relationship Diagrammer, they also show up in the RON, under the Allowable Values sub-node:

- where possible, defining allowable values should be done in a domain rather than explicitly in an attribute
- if this is not possible (i.e. the Database Design Transformer creates this list for discriminator columns in super-type implementations), allowable values can be defined for each attribute in this group

5.2.5 Standard Entity Enhancements

It is the Ministry standard that the following audit attributes be added to all entities:

```
CREATE_USERID      not null  varchar2(30)
CREATE_TIMESTAMP   not null  date
UPDATE_USERID      not null  varchar2(30)
UPDATE_TIMESTAMP   not null  date
```

When implemented as columns in the table, these attributes allow a degree of simple security tracking, but can also be useful in tracing down problems.

See Standard Table Enhancements for more implementation details.

5.2.6 Relationships

A Relationship represents any significant way in which two entities can be associated.

Property	Rule	Req?
From		
Relationship Name	<ul style="list-style-type: none"> relationship names must be meaningful both sides of a relationship must be named "catch all" phrases (related to, associated with) should be avoided in favour of more descriptive names <p>Note: It is often helpful to consider the relationship name in the context of a sentence as follows:</p> <p>EACH entity1 MUST BE/MAY BE relationship ONE AND ONLY ONE/ONE OR MORE entity2</p> <p>for example: EACH student MUST BE enrolled in ONE OR MORE classes, or EACH class MAY BE comprised of ONE OR MORE students</p>	Y
Minimum	<ul style="list-style-type: none"> defines optionality of the relationship 	Y

Property	Rule	Req?
Cardinality	0: MAY BE 1: MUST BE	
Maximum Cardinality	<ul style="list-style-type: none"> mandatory in the sense that it must be considered defines the degree of the relationship 1: ONE AND ONLY ONE null: ONE OR MORE	Y
Transferable	<ul style="list-style-type: none"> mandatory in the sense that it must be considered by default, a relationship is transferable, which means the end can be disconnected from the current instance and reconnected to another instance 	Y
To		
Relationship Name	<ul style="list-style-type: none"> relationship names must be meaningful both sides of a relationship must be named "catch all" phrases (related to, associated with) should be avoided in favour of more descriptive names Note: It is often helpful to consider the relationship name in the context of a sentence as follows: EACH entity1 MUST BE/MAY BE relationship ONE AND ONLY ONE/ONE OR MORE entity2 for example: EACH student MUST BE enrolled in ONE OR MORE classes, or EACH class MAY BE comprised of ONE OR MORE students	Y
Minimum Cardinality	<ul style="list-style-type: none"> defines optionality of the relationship 0: MAY BE 1: MUST BE	Y
Maximum Cardinality	<ul style="list-style-type: none"> mandatory in the sense that it must be considered defines the degree of the relationship 1: ONE AND ONLY ONE null: ONE OR MORE	Y
Transferable	<ul style="list-style-type: none"> mandatory in the sense that it must be considered 	Y

Note: One to one relationships should be carefully reviewed; they may actually be sub-types, perhaps with different names or attributes or relationships.

Note: Relationships that are optional at both ends should also be carefully reviewed; they are nearly always a modelling error.

5.2.7 Domains

A domain categorizes the nature of the data represented by a group of attributes, and indicates the general purpose of those attributes. The use of domains can save time and apply a desirably high degree of standardization across attribute definitions, and subsequently, column names.

Domains are also used to implement lists and ranges of valid values. The use of domains to implement lists of values should only be considered when the list of allowable values is static (e.g. days of week, months of the year, yes/no indicators). Domains are created in the Repository Object Navigator, the Server Model Diagrammer or the ER Diagrammer (choose Edit Elements/Domain from the menu).

Only include attributes in a domain when the values that they represent all have the same business meaning. Where applicable, domains must also represent the same units of measure.

Domains must be defined for each application, and reviewed by the Data Administrator.

It is the Ministry standard to place all attributes under domains.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> should be made up of one to three real words when the domain name is only one word, it must not be an Oracle Reserved Word should be singular should be meaningful; abbreviations should be avoided unless obvious 	Y
Attributes in Domain	<ul style="list-style-type: none"> These fields define the datatypes to be used for attributes 	
Format		Y
Ave Att Length		Y
Max Att Length		Y
Att Decimal Places	<ul style="list-style-type: none"> mandatory for numeric datatypes 	N
Unit of Measure	<ul style="list-style-type: none"> applicable to Domain Attributes only 	N
Columns in Domain	<ul style="list-style-type: none"> These fields define the datatypes to be used for columns 	
Datatype		Y
Ave Col Length		Y
Max Col Length		Y
Col Decimal Places	<ul style="list-style-type: none"> mandatory for numeric datatypes 	N
Dynamic List?	<ul style="list-style-type: none"> if selected, will cause the LOV to be implemented as a table lookup (<APPL>_REF_CODES) <p>Note: This is only if the column's Display Datatype is Poplist or Text</p>	N
Documentation		
Comment	<ul style="list-style-type: none"> should contain a simple description of the domain 	Y
Description	<ul style="list-style-type: none"> describes the domain 	Y
Notes	<ul style="list-style-type: none"> contains any additional information about the domain 	N

If the domain is enumerated, then the values are listed under the Allowable Values sub-node.

Property	Rule	Req?
Value	<ul style="list-style-type: none"> valid value for the attribute/column in this domain or the lowest allowable value when implementing a range of values 	Y
High Value	<ul style="list-style-type: none"> maximum allowable value when implementing a range of values 	N
Abbreviation	<ul style="list-style-type: none"> mandatory for entries representing a valid value in a list of values 	N
Meaning	<ul style="list-style-type: none"> mandatory for entries representing a valid value in a list of values 	N
Display Sequence	<ul style="list-style-type: none"> mandatory for entries representing a valid value in a list of values determines the order the values are displayed in the list 	N
Documentation		
Description	<ul style="list-style-type: none"> contains a description of the allowable value/range 	N
Notes	<ul style="list-style-type: none"> contains any additional information about the allowable value/range 	N

Changes to a domain can only be propagated to the associated attributes and columns by using the *Update Attributes in a Domain* and *Update Columns in a Domain* utilities. These can be accessed from the Utilities menu of the Repository Object Navigator.

Note: For enumerated values, a lookup entity (with the valid values stored as data) may be more appropriate if the valid values are subject to change; for example, city names or product codes. If the valid values are relatively static, then a domain is more appropriate; for example, gender or compass direction.

5.3 Functional Modeling

Functional Modeling is used to describe what an organization does. Simply put, Functional Modeling involves identifying what a business does (functions), what triggers those activities (events), and which things of significance (entities) or properties of those things (attributes) are acted upon by the functions.

The Functional Model should be geared towards what the organization is attempting to achieve in terms of its objectives, and should be easily understood by non-application personnel.

The Repository Object Navigator (RON), and Functional Hierarchy Diagrammer components are used to model the hierarchy of business functions.

5.3.1 Objectives

The objectives of the Functional Modeling process are:

- To provide an accurate model of the functional needs of the organization, which will act as a framework for the development of new or enhanced systems.
- To provide a model that is independent of any mechanism or process method, and allows for objective decisions to be made about implementation techniques and coexistence with existing systems.

5.3.2 Deliverables

The Functional Model documentation to be presented for sign-off will contain the following diagrams and reports:

- Functional Hierarchy Diagram
- Function to Entity Matrix Diagram (CRUD Matrix)
- Function Definition Report

5.3.2.1 Functional Hierarchy Diagram

A Functional Hierarchy Diagram is a hierarchy of business functions describing what the organization does not how it does it. It does not include mechanisms, examples, organization, responsibilities or roles.

- All Function Hierarchy Diagrams must include the Summary Information in the upper left hand corner of the diagram
- The diagram name must be prefixed with "FHD"
- There must be one "complete" Functional Hierarchy Diagram per application system
- A function definition should be a single sentence that begins with a verb followed by the entity (or entities) affected
- All entity names should be capitalized when used within definitions and descriptions

Diagram Name : FHD - SYSTEM, CRISP MODEL
Title : High Level System View
Date last modified : 07 June 1999 Time last modified : 17:19:39
Author : GW

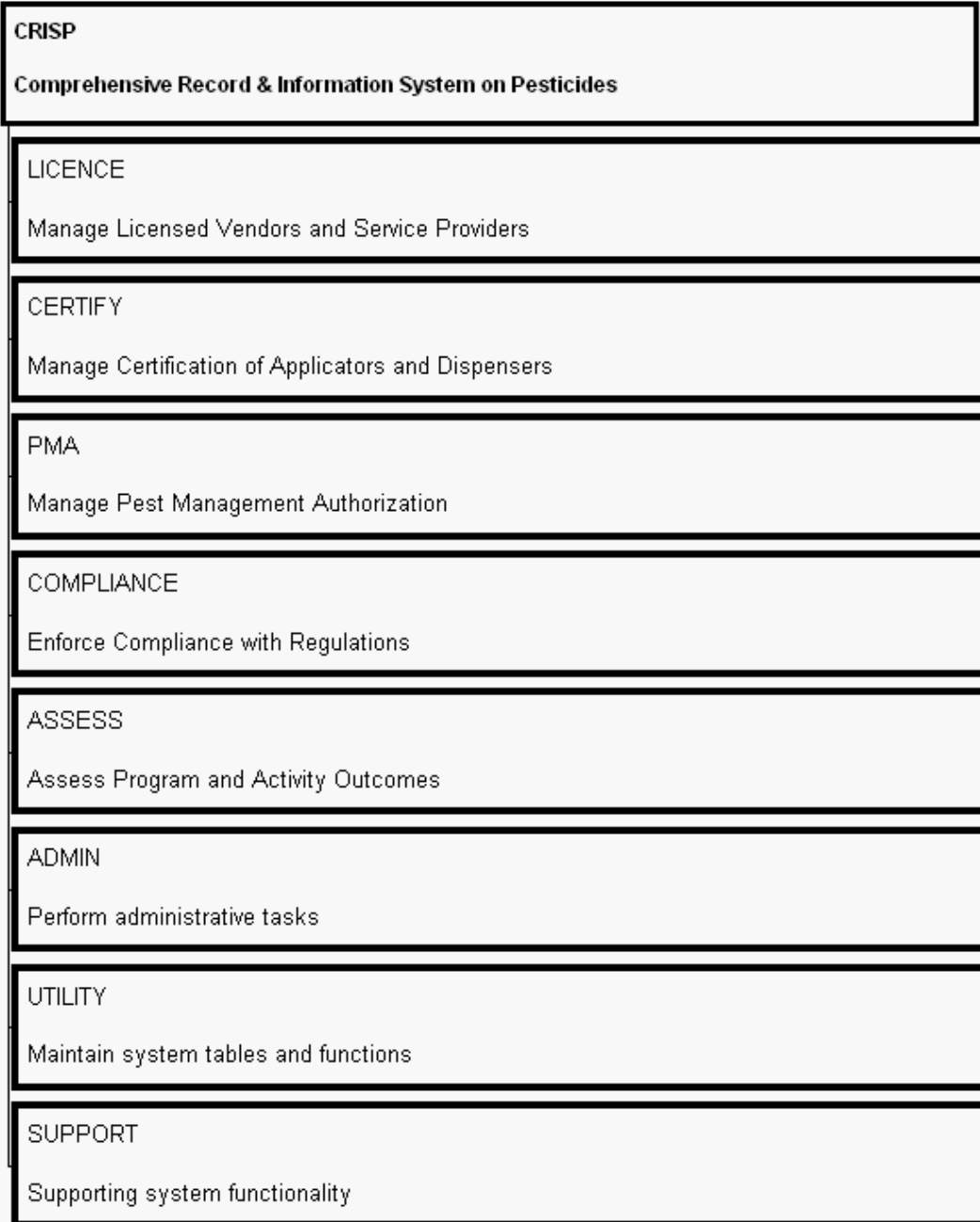


Figure 6: Function Hierarchy Diagram

5.3.2.2 Function to Entity Matrix (CRUD Matrix)

The CRUD Matrix is used to check for completeness and correctness for the function (and data) model. Completeness is assured if the following is true:

- All elementary functions must refer to at least one entity
- All entities must have a usage by at least one function
- Each entity must have a created, read, updated and deleted usage

5.3.2.3 Function Detail Report

Any specific processing requirements for a function should be documented and available for review on this report.

5.3.3 Functions

A function is something that an organization does or needs to do, irrespective of how it does it. An elementary function is a lowest implementable function, and results in a form module, reports module, etceteras.

Property	Rule	Req?
Label	<ul style="list-style-type: none"> • must be unique within the application 	Y
Short Definition	<ul style="list-style-type: none"> • function descriptions must start with a verb acting upon an object (or objects) 	Y
Elementary?	<ul style="list-style-type: none"> • must be considered • elementary functions act upon data and are the only functions to have associations with entities • an elementary function must be associated with at least one entity 	N
Frequency		
Frequency	<ul style="list-style-type: none"> • mandatory for elementary functions 	N
Frequency Unit	<ul style="list-style-type: none"> • mandatory for elementary functions 	N
Response Needed	<ul style="list-style-type: none"> • mandatory for elementary functions 	N
Documentation		
Description	<ul style="list-style-type: none"> • must be clear and concise • must be more detailed than the Short Definition • references to entities must be CAPITALIZED • can be further categorized into Goal, Description, Example of Typical Use, and Miscellaneous, for example: <p>GOAL Record contact information for a license applicant.</p> <p>DESCRIPTION This function enables the user to enter contact information, such as name, address and telephone number, into the system. The user can also update existing information for a licensee.</p> <p>When entering new contact information, the user should be sure that the licensee is truly unknown</p>	Y

Property	Rule	Req?
	<p>to us (in any role, including permittee, PMP holder, etc.). Since a licensee may be an organization or person, the user should first declare how the applicant should be recorded (i.e. last name and first name, or legal registered business name).</p> <p>EXAMPLE OF TYPICAL USE This function is used when entering the initial license application from a new applicant, or when updating the contact information for a known licensee.</p> <p>MISCELLANEOUS When updating the licensee's name, it will be necessary to keep the old name (since the original license was issued in this name) in addition to recording the new name (for future reference).</p>	
Notes	<ul style="list-style-type: none"> should contain any notes about this function structured analysis or design comments should be placed here, for example: <p>OUTSTANDING ISSUES =====</p> <p>A? 1998-01-27 GW There may be an opportunity to share this Functionality with 'Record contact information about a new permittee / PMP'er / Certificate holder.</p> <p>IMPLEMENTATION NOTES =====</p>	Y

Although data usages are usually entered via the Function Hierarchy Diagrammer, they also show up in the RON, under the following sub-nodes.

Usages Node	Rule	Req?
Using Entities	<ul style="list-style-type: none"> One or more entities must be associated with each elementary function. How the function acts upon the entity must be recorded (e.g. create, update, delete) 	Y
Using Attributes	<ul style="list-style-type: none"> One or more attributes must be associated with each entity within each elementary function. How the function acts upon the attribute must be recorded (e.g. create, update, delete...) optional, and only for elementary functions 	N

5.4 Business Processing Modelling

The Process Model is intended to describe the business, and requirements for a system to capture information about the business, and events which occur in it. The focus, therefore, in Process Models will

be on the information used in, and produced by, the process and how that information will be handled within the process.

Formal Process Models produced early in the development and reviewed by users avoid repeated changes to the system as business processes are determined by trial and error during the design phase. The Repository Object Navigator (RON), and Process Modeler components are used to model the business processes.

5.4.1 Objectives

The objectives of Process Modeling are:

- To provide the basis for confirming and understanding business activity.
- To provide a means to define business requirements before system is designed and built
- To provide a framework for analyzing and improving current business processes and for business re-design.
- To document the results of the analysis of business information requirements
- To verify the data model
- To analyze organizational responsibilities

The analysis should model the following key ideas:

- Business Activities
- Hierarchical Organization and Decomposition
- Data Objects
- Data Usage (i.e., association matrices)
- Activation, Sequencing, and Termination
- Governing Rules and Conditions

High-level guidelines for modelling business processes include:

- Process Model Diagrams will use boxes for processes and arrows to indicate the flow
- The process boxes in the models will be labeled in a numeric sequence indicating the hierarchy (e.g. 1,2,3 at the highest level, 1.1, 1.2, 1.3 at the next level, 1.1.1, 1.1.2, 1.1.3 at the next level and so on)
- Diagrams should not have a wide variety of different diagram shapes (e.g. use only the basic Process box, Event arrow, Input arrow, Output arrow, Decision diamond, and Data Store rectangle)
- Diagrams should have no more than ten processes; if a diagram requires more, then link it to another diagram which continues the process
- The process modelling exercise should ensure that the "as is" and "to be" processes are documented separately.
- Items identified during the development of the initial or "analytic" model should be included in the text of the final process modelling documentation so as not to lose the knowledge gained. These could be such items as responsibilities for processes, interactions between processes and specific systems, gaps between the "as is" and the "to be" models, and issues that hamper the "as is" model".
- It is recommended, but not mandatory, to use Oracle Designer to produce Business Process Models; alternatives include Microsoft Visio and/or Microsoft Word.

Note that these alternative documents must then be checked in and held in the Repository.

5.4.2 Deliverables

The Process Model documentation to be presented for sign-off will contain the following diagrams and reports:

- Process Model Diagram(s)
- Process Definition Report (Function Definition Report if Designer is used)

5.5 Business Areas

During the Analysis phase, it is important to record the involved organizational departments (e.g. Information Services Branch), and the relevant roles that a user may play (e.g. child care worker). Both are recorded in Designer as Business Units.

5.5.1 Deliverables

The Business Area documentation to be presented for sign-off will contain the following diagrams and reports:

- Business Unit Definition Report
- Business Function to Business Unit Matrix

5.5.2 Business Units

Using the Repository Object Navigator, Business Units (also known as User Groups) may be recorded. Business Units define the departments, divisions, and organizational units that are within the scope of the application. Roles are also documented here, as a business unit may be a role that the person, or persons, plays in the use of the application.

Property	Rule	Req?
Short Name	<ul style="list-style-type: none"> • must be unique within the application • abbreviation for the business unit (e.g. "App Mgr" for Application Manager) 	Y
Name	<ul style="list-style-type: none"> • full name of the business unit or role (e.g. Conservation Office) 	Y
Parent	<ul style="list-style-type: none"> • the parent business unit (e.g. Federal Government could be the parent of Environment Canada) 	
Documentation		
Comment	<ul style="list-style-type: none"> • should contain a simple description of the business unit 	Y
Description	<ul style="list-style-type: none"> • describes the business unit 	Y
Notes	<ul style="list-style-type: none"> • contains any additional information about the business unit 	N
BPR		
Role	<ul style="list-style-type: none"> • identifies that this business unit is a role, to be implemented as a database role during the Design Phase 	Y

Although function usages can be entered via the Function Hierarchy Diagrammer, they also show up in the RON, under the Performing Functions sub-node. The actual usages are mandatory, but the frequency and response information is optional.

Property	Rule	Req?
Function	<ul style="list-style-type: none"> • reference to the business function for which this business unit has access • chosen from a picklist 	Y
Frequency	<ul style="list-style-type: none"> • frequency of access 	N
Frequency Unit	<ul style="list-style-type: none"> • unit of measure for the frequency 	N
Response Needed	<ul style="list-style-type: none"> • type of response; translates to on-line versus batch implementation during the Design phase 	N

Although data usages can be entered via the Entity Relationship Diagrammer, they also show up in the RON, under the Using Entities sub-node. The actual usages are mandatory, but the frequency and response information is optional.

Property	Rule	Req?
Entity	<ul style="list-style-type: none"> reference to the entity for which this business unit has access chosen from a picklist 	Y
Initial Volume	<ul style="list-style-type: none"> expected number of records that this business unit contributes to the entity (table) when the system first goes into production 	N
Maximum Volume	<ul style="list-style-type: none"> expected number of records that this business unit contributes to the entity (table) at the end of the third year 	N
Average Volume	<ul style="list-style-type: none"> average number of expected records that this business unit contributes to the entity (table) at the end of the third year 	N
Annual Growth	<ul style="list-style-type: none"> expected % annual growth rate that this business unit contributes to the entity (table) 	N

These business units form the basis for defining authorization rules, such as:

- the hierarchy of user roles (parent property of the business unit)
- the function access rules (which Performing Functions are used)
- the vertical data access rules (which Using Entities are linked)

The Business Function to Business Unit matrix provides the documentation for security and network constraints; it shows which Business Units perform which Business Functions.

5.6 Business Rules Modelling

A business rule is:

- a restriction that applies to the state of the data or to the change of data, or
- an automatic action that takes place after a change in data

It is recommended that the modelling of business rules be treated as an integral part of analysis. Most of these rules can be documented in the data model (e.g. format, domain rules), but many others (e.g. change event rules, update rules, delete rules) have no pre-defined property in Designer. These rules are often hidden in the descriptions and notes text of the functions and entities.

Another method is to organize the rules into classes, and record them in a separate 'Business Rules' function hierarchy. This 'Business Rules' hierarchy is completely separate and distinct from the 'Business Functions Hierarchy' usually created during Analysis.

An example of such a 'Business Rules' hierarchy is the following:

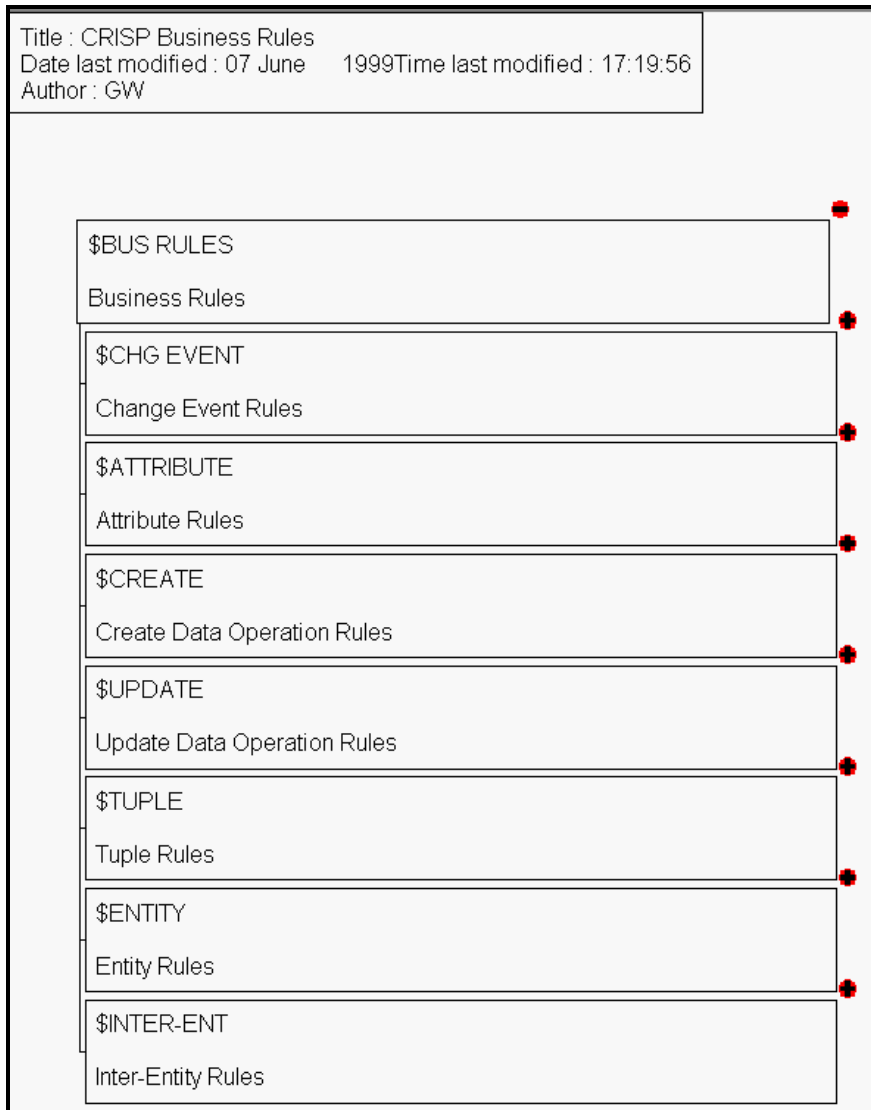


Figure 7: Business Rules Hierarchy

There are several advantages to this method:

1. Event modelling can be used to explicitly describe the event that triggers the rule
2. Function to Entity and matrices can be created to show the entire life-cycle of an entity
3. During the Design and Build phases, the implemented rules (e.g. stored procedures, modules, or database triggers) can be cross-checked using the 'Implementing Business Functions' usage of the module; this would facilitate completeness checking of the rules
4. The business rule is documented in one and only one location, and its subsequent implementation can be easily traced to this business rule.

Note: Advantage #3 does not apply for rules implemented as Check Constraints; although the comment property of the check constraint could refer to the source business rule.

There are four main classes of business rules:

- static data rules
- data operation rules
- change event rules
- authorization rules

Static data rules are rules that always apply. Every state of the data has to adhere to all the data rules. Static data rules are tightly integrated with entity relationship modeling. Some of these rules are implicitly created during entity relationship modeling and can be graphically represented in an entity relationship model.

Data operation rules are dynamic, which means that they tell something about valid state transitions but nothing about the state itself.

Change event rules define automatic actions to be taken after the state of the data has changed.

Authorization rules define which business unit, person or group of people is able to perform a function or manipulate (a set of) data.

There are sub-classes under these four main classes, and are shown in the following table:

RULE			Recording Method	Function Prefix	Tool Access
Class	Sub-Class	Type			
Static Data Rules	Attribute Rules	Format	Attribute Format / Function Description		ER Diagrammer
		Optionality	Attribute Required		ER Diagrammer
		Allowable Values	Domain object		ER Diagrammer
		Other Attribute	Function Description	\$TUP	FHD ¹
	Tuple Rules	Tuple	Function Description	\$TUP	FHD
	Entity Rules	Unique ID	Unique Identifier		ERD
		Other Entity	Function Description	\$ENT	FHD
	Inter-Entity Rules	Referential Integrity	Relationship		ER Diagrammer
		Relationship Cardinality	Relationship property		ER Diagrammer
		Restricted Relationship	Function Description	\$RER	FHD
Data Operation Rules	Create Rules	Create Rules	Function	\$CRE	FHD
	Update Rules	Transferable Relationship	Transferable Property		ERD
		Attribute Transition	Function Description	\$UPD	FHD
		Other Update	Function Description	\$UPD	FHD
	Delete Rules	Relationship	Function Description	\$DEL	FHD
Other Delete		Function Description	\$DEL	FHD	
Change Event Rules		Change Event	Function Description	\$CEV	FHD
Authorization Rules		Function Access	Business Unit to Function matrix		Matrix Diagrammer
		Vertical Data	Business Unit to Entity/Attribute matrix		Matrix Diagrammer
		Horizontal Data	Model as Entity		ER Diagrammer

¹ Function Hierarchy Diagrammer

The shaded types are those rules which can be recorded in the data model, or through the appropriate matrices; these rules do not belong in the 'Business Rules Hierarchy'. The tool FHD refers to the Function Hierarchy Diagrammer used to maintain the 'Business Rules Hierarchy'.

Rule Sub-Class	Definition
Other Attribute	All other allowable value rules for an attribute. In most cases the attribute value depends on the value of a constant.
Tuple Rules	Tuple rules define attribute allowable values which depend on the value of other attribute(s) of the same entity occurrence (tuple).
Other Entity Rules	Other entity rules consist of all allowable value rules which you can define within an entity.
Restricted Relationship Rules	A restricted relationship rule restricts the set of entity occurrences to which an entity relationship can refer.
Attribute Transition Rules	Transition rules define attribute allowable values that depend only on the previous value of the same attribute.
Change Event Rules	<p>A change event rule defines an automated, derived (secondary) action that takes place after a data state change. The change event defines which change of data triggers the automated action. This can be one, or a combination, of the following events:</p> <ul style="list-style-type: none"> • creation or deletion of an entity • insert or update of an attribute value • creation, transfer, or deletion of an entity relationship <p>The automated action triggered is most likely another data manipulation action (which in turn can trigger another change event rule). However, it could be anything else, for example sending an email or printing a report.</p>

The rules which are not shaded are recorded in the appropriate Designer property, and do not require explanation. The shaded rules may require some examples.

Examples of these are:

Rule Type	Example
Other Attribute	"Employee Salary must be a multiple of 1000."
Tuple	"Employee end date must be later than employee hire date."
Other Entity	"No more than 20 departments are allowed."
Restricted Relationship	"An employee can only be managed by an employee with job 'MANAGER'".
Create Rules	"An order item can not be added to an order that is already approved."
Attribute Transition	<p>"Allowed transitions for marital status of employee are:"</p> <ul style="list-style-type: none"> • unmarried → married • married → divorced • divorced → married • married → widow • widow → married
Other Update	"The price of a product cannot be updated if the status of the article is 'obsolete'."
Relationship	<p>"When a department is deleted all employees working for that department should be deleted as well. "(cascade)</p> <p>"A department cannot be deleted when employees exist who are working for that department. "(restricted)</p> <p>This rule maps to the choice of 'On Delete' Cascade rules for new foreign keys</p>

Rule Type	Example
	in the Database Design Transformer (Keys tab of the Settings dialog). The only other choice is 'Nullify' (affected employees no longer have a parent department', which is not recommended.
Other Delete	"A client may only be deleted when there are no outstanding orders."
Change Event	"When the end date of a customer of a telephone company is set, create an occurrence in Work Order of type 'disconnect customer' so the maintenance crew will perform the disconnection."

The following is an example of Tuple rules in the 'Business Rules' hierarchy;

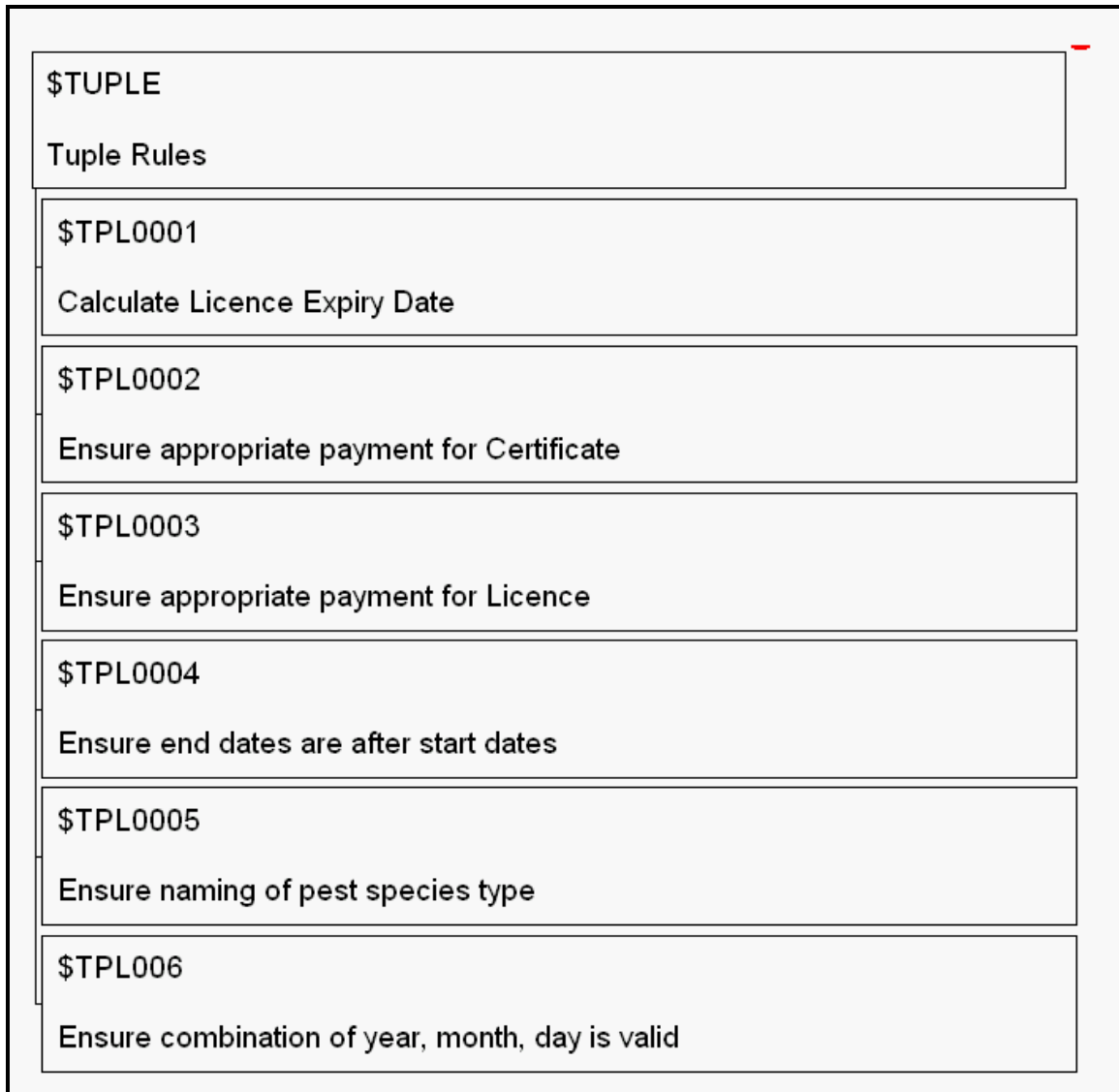


Figure 8: Tuple Rules Hierarchy

The function label holds the rule name, the short definition is a succinct phrase, and the description hold the actual rule.

For example:

Label	Short Definition	Description
\$IER0021	Calculate Certification Expiry Date	IF 60 <= EXAM_RESULT <= 74 THEN EXPIRY_DATE = Issue Date + 1 Calendar Year ELSIF 75 <= EXAM_RESULT <= 100 THEN EXPIRY_DATE = Issue Date + 5 Calendar Year Note that this will be a DB function, so that changes to this logic may be easily distributed.
\$SENT0004	Ensure unique names for a party	For each party, ensure that no duplicate names are recorded. This could be accomplished by a unique index on the appropriate combination of attributes and relationships; including the EFFECTIVE DATE attribute to allow the re-use of names (i.e. by changing to a previously used name).

This 'Business Rules Hierarchy' is not a Ministry standard, but is a suggested method of modelling business rules. The main objective for explicitly modeling rules this way is to allow for a more structured approach to actually implementing the business rules.

6 Design Phase

6.1 Overall Guidelines

This section presents some overall guidelines to assist in the Design Phase within the Oracle Designer environment.

6.1.1 Referencing Objects in Text Descriptions

Whenever the name of another TABLE, COLUMN (or any other object) is used within a textual description, it should be capitalized for easier reading (and reference).

For example, if ADAMS_MODEMS is a table, then the following description should be used for the ADAMS_MODEM_TYPES table:

"This table identifies the types of ADAMS_MODEMS that are available to the polling system"

Note: This may make maintenance of this text difficult, as changes in table names would necessitate updates to the descriptive text. Therefore, this is a recommended guideline, and not a Ministry standard.

6.1.2 Keeping logical data model current

In the Design Phase, there may be corrections and/or additions to the data requirements (i.e. new column). Aside from de-normalization or other issues specific to physical implementation, all such changes must be re-documented in the logical data model, either via the *Table to Entity Retrofit Utility* or via manual update of entities, attributes and relationships.

See [Synchronizing Entities with Tables](#) for further information on the *Table to Entity Retrofit Utility*.

6.1.3 Electronic Delivery of the Application

All development is done directly against the Ministry Repository, so no explicit delivery is required. However, all vendors must perform a specific number of steps. For details of this standard process, please refer to Section 8.4 (Promotion Management Procedures) of the Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

For a complete overview of the Ministry standard Promotion Model, see the Ministry's Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

6.2 Database Design

The Database Design (Physical Data Modeling) process involves the conversion of entities, attributes, relationships, and other logical constructs to their physical database counterparts. Specifically, entities are mapped to their corresponding table definitions, attributes to their corresponding column definitions, relationships to their corresponding foreign key definitions, and so on.

The process of converting analysis data into tables and foreign key constraints is automated by the *Database Design Transformer* (DDT). The *Database Design Transformer* creates and maintains database designs based upon entity, attribute and relationship information previously recorded in the Designer

Repository. The *Database Design Transformer* creates tables to record instances of each entity, columns to store the attributes, and constraints to implement the relationships between entities. It also creates constraints to enforce any unique identifiers that have been defined, and indexes to support foreign keys. The *Database Design Transformer* generated database design is stored in the Designer Repository. This model can subsequently be used by the Server Generator to generate the SQL DDL statements required to create the associated database objects.

The Server Model Diagrammer is a graphical tool for modeling logical database schema designs. The database objects within the schema can be represented graphically on one or more data diagrams. These diagrams depict the relationships between tables, views and snapshots recorded in the Designer repository. After a first cut database design is completed using the Database Design Transformer, the Server Model Diagrammer can be utilized to refine the database design.

De-normalization and the addition of columns to support special processing logic may be done as required. This must, however, be fully documented in the description for the column and must be done with DBA approval.

6.2.1 Objectives

The objectives of the Database Design process are:

- To ensure that all entities, attributes, and relationships that are to be physically implemented have corresponding database objects.
- To ensure that the transition from the logical model to the physical database design is documented.
- To provide an accurate model of the database requirements of the organization. This model can subsequently be used by the Server Generator to generate the statements required to create the associated database objects.

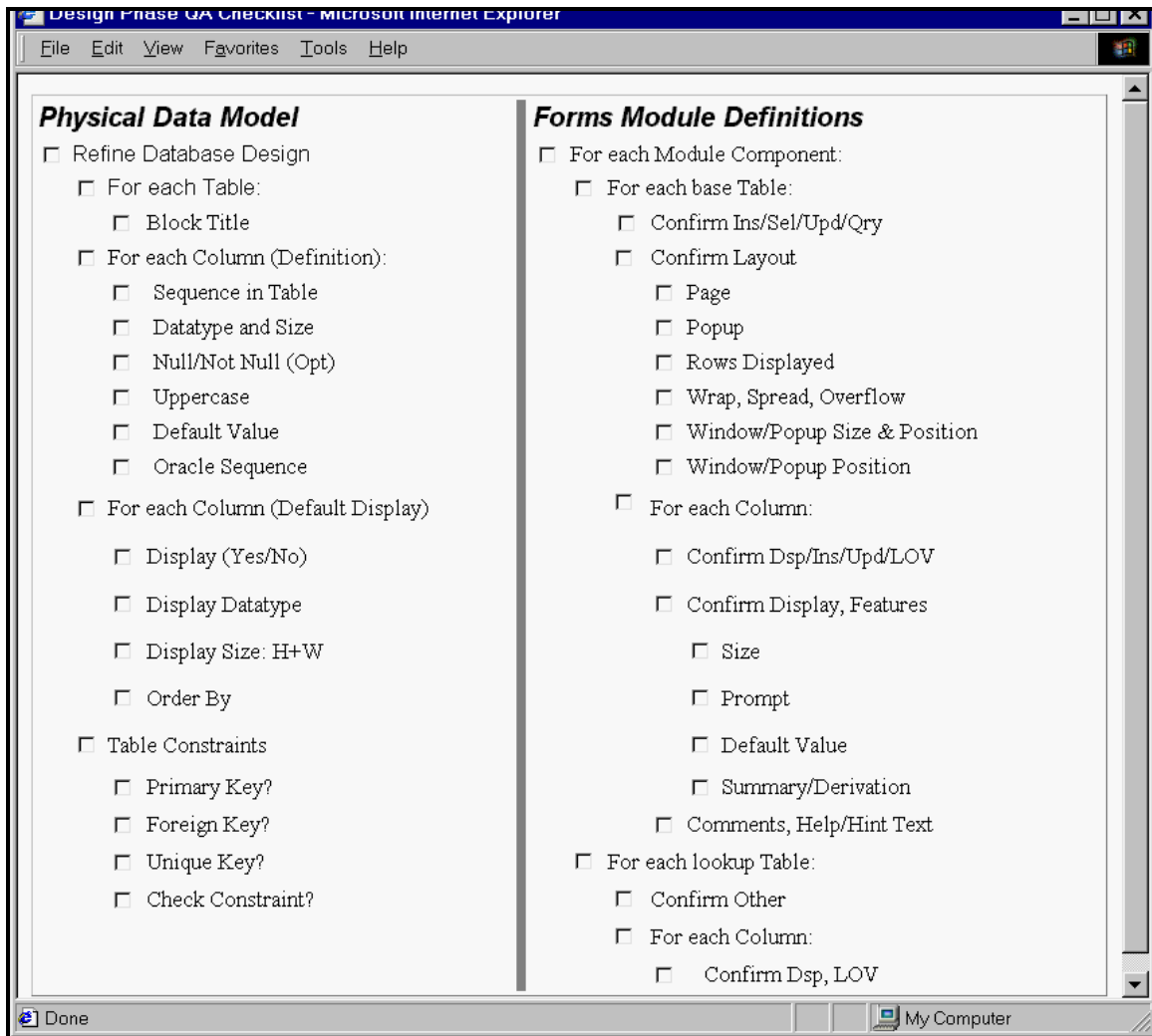
6.2.2 Deliverables

The Database Design document to be presented for sign-off will contain the following diagrams and reports:

- Proposed Database Design
- Entity to Table Implementation
- Table Definition Report
- Server Model Diagram
- Database Table and Index Size Estimates
- Role Definition Report

A checklist is available to confirm that the Database Design task is complete and that the repository is ready for the Build Phase. This checklist is used in conjunction with the deliverables stated above.

6.2.2.1 Design Phase QA Checklist



The screenshot shows a web browser window titled "Design Phase QA Checklist - Microsoft Internet Explorer". The browser's menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The main content area is split into two columns:

- Physical Data Model**
 - Refine Database Design
 - For each Table:
 - Block Title
 - For each Column (Definition):
 - Sequence in Table
 - Datatype and Size
 - Null/Not Null (Opt)
 - Uppercase
 - Default Value
 - Oracle Sequence
 - For each Column (Default Display)
 - Display (Yes/No)
 - Display Datatype
 - Display Size: H+W
 - Order By
 - Table Constraints
 - Primary Key?
 - Foreign Key?
 - Unique Key?
 - Check Constraint?
- Forms Module Definitions**
 - For each Module Component:
 - For each base Table:
 - Confirm Ins/Sel/Upd/Qry
 - Confirm Layout
 - Page
 - Popup
 - Rows Displayed
 - Wrap, Spread, Overflow
 - Window/Popup Size & Position
 - Window/Popup Position
 - For each Column:
 - Confirm Dsp/Ins/Upd/LOV
 - Confirm Display, Features
 - Size
 - Prompt
 - Default Value
 - Summary/Derivation
 - Comments, Help/Hint Text
 - For each lookup Table:
 - Confirm Other
 - For each Column:
 - Confirm Dsp, LOV

Figure 9: Design Phase QA Checklist

6.2.2.2 Proposed Database Design

This is not a standard Designer report, but is a Word document describing major sub-type implementations and denormalization rationales.

6.2.2.3 Entity to Table Report

- All entities that are to be physically implemented must have a corresponding table.
- The transition from logical entities to physical tables is documented.
- Supertype & subtype mapping must be addressed.

6.2.2.4 Table Definition Report

- All tables must have a primary key.

- Any special tables (e.g. tables with no relationships, tables implemented for physical reasons only, etc.) should be well documented in the 'Comments'.
- All columns should be understandable (either by its name or by the comments or by the use of an example in the comments about that column) to a non-application person.

6.2.2.5 Server Model Diagram

One or more diagrams should be made to show the relationships between the tables

6.2.2.6 Database Table and Index Size Estimates

As this report tables the volume estimates for all tables and columns (inherited from their corresponding entities and attributes during the table generation process with the Database Design Transformer) and calculates approximate sizes for the tablespaces.

This report is vitally important; it is the Ministry standard that these sizing estimates be performed.

6.2.2.7 Role Definition Report

This report shows the database roles in the application. All security should be enforced at the server, using role-based security.

6.2.3 Object Naming Conventions

The Ministry's conventions for naming database objects (including tables, views, columns, indexes, sequences, roles, packages and functions, etceteras) follow those basic naming conventions imposed by Oracle:

- object names should be maximum of thirty (30) characters long with these exceptions:
 - names of databases are limited to 8 characters
 - names of database links can be as long as 128 characters
- should not contain quotation marks
- are in upper-case
- can only contain alphanumeric characters from the database character set and the characters _, \$, and #. The use of \$ and # is strongly discouraged. Names of database links can also contain periods (.) and at-signs (@)
- should contain underscores (_) for visual clarity
- must begin with a letter
- must not duplicate an ORACLE reserved word
- should not contain the word DUAL (e.g. DUAL is the name of a dummy table frequently accessed by Oracle tools such as SQL*Plus and Forms)
- must not duplicate the name of another database object
- should use nouns, rather than verbs
- should be as descriptive and as short as possible
- should use standard abbreviations when required (see [Appendix C – Standard Approved Abbreviations](#))
- should not be ambiguous
- In addition, it is a Ministry standard that the Application Name (acronym such as LGIS) be prefixed to all 'physical' database objects such as tables, views, packages, sequences and roles.

Note: Procedures and functions defined within a package do not need this prefix, as the package itself will be prefixed with the application name.

6.2.4 Database Design Transformer

The Database Design Transformer can be used to easily convert the logical model into a physical implementation. Bear in mind, however, that no automated process is without problems; it is the developers' responsibility to ensure that the way that all objects are built is correct and will satisfy the business needs.

The following *Run Options* should be used:

- the first time running the DDT, the Create flags are set; subsequent runs will have the Modify flags set to allow modifications to existing objects

The following *Settings - Database* should be used:

- the Database should be set to a database that has been defined
- the Database User should be set to a application schema user that has been defined
- the Tables - Tablespace should be set to the tables tablespace name
- the Index - Tablespace should be set to the index tablespace name
- Commit frequency for changes allows the user to determine when/if the results of the design session will be saved:
 - *After each phase* is the most efficient and will allow some work to be saved even if later steps fail to process
 - At end of run will allow the rollback of the entire session if an error occurs
 - *Don't commit* allows the user to perform a trial run to see what objects would be built, but without saving anything

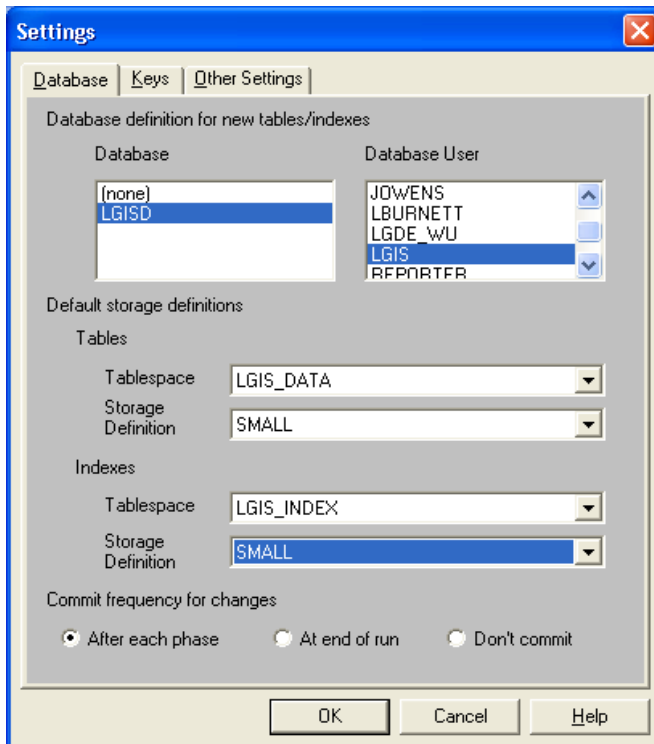


Figure 10: DDT Settings - Database

The following *Settings - Keys* should be used:

- the Surrogate Keys - Create surrogate keys for all new tables option will create a primary key if a table does not have one.

Most of the time, the developers should ensure that all entities have UID's defined before using the DDT, so this option should not be used.

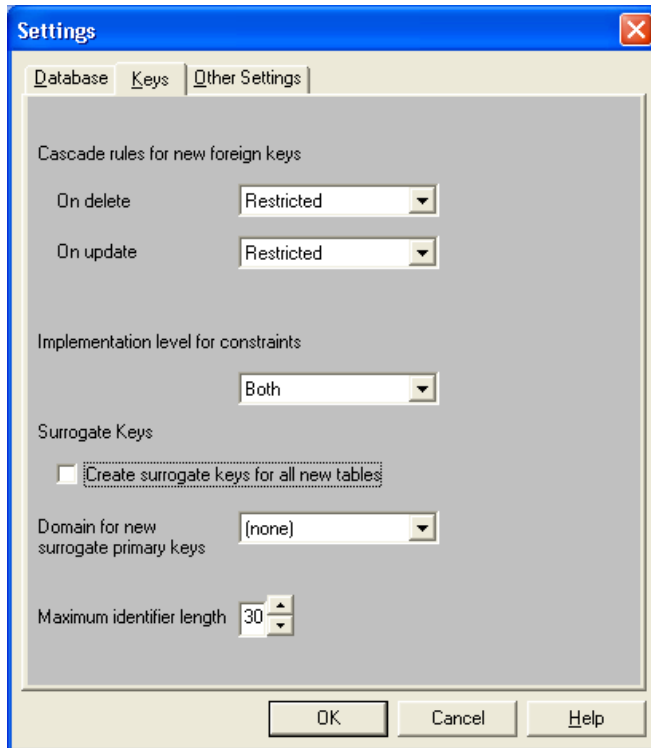


Figure 11: DDT Settings - Keys

Note: Neither 'Nullify' nor 'Delete' is permitted as a cascade rule. This applies to 'On Delete' as well as 'On Update'.

The following Settings - Other Settings should be used:

- the Elements that you want prefixes generated for (Columns) option should NOT be checked
- the Elements that you want prefixes generated for (Foreign Key Columns) option should be checked
- the Elements that you want prefixes generated for (Surrogate Key Columns) option should be checked
- the Table Prefix should be set to the Application Short Name followed by an underscore (e.g. LGIS_)

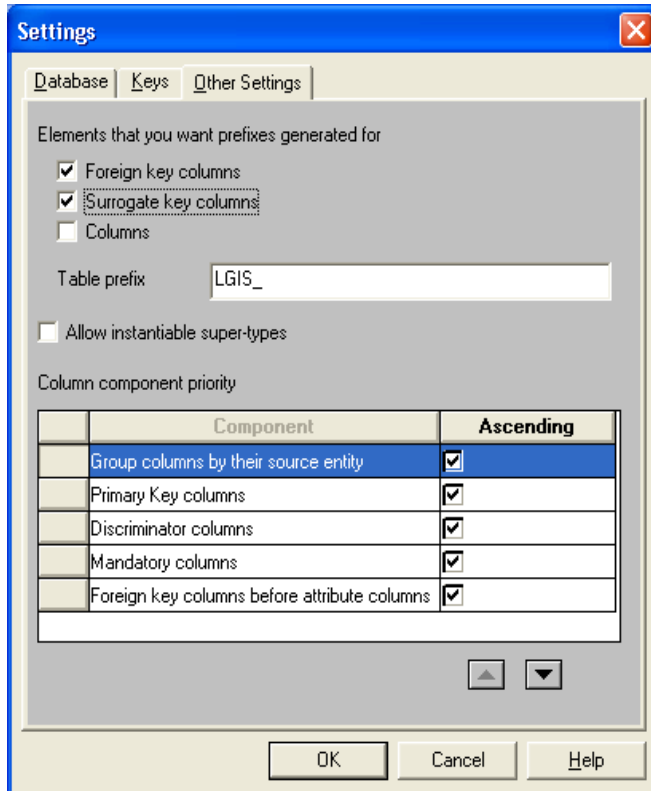


Figure 12: DDT Settings - Other

6.2.5 Standard Table Enhancements

Tables which record transactions, or go through changes in state due to a business process, must have these audit columns. For all other tables, it is still mandatory to include these columns; this is especially important for data warehouse replication to determine when data values have changed. They allow a degree of simple security tracking, but can also be useful in tracing down problems.

As per the Standard Entity Enhancements section, these columns will be automatically generated from the source attributes.

It is the Ministry standard that the following audit columns be included in all tables:

```
CREATE_USERID      not null  varchar2(30)
CREATE_TIMESTAMP   not null  date
UPDATE_USERID      not null  varchar2(30)
UPDATE_TIMESTAMP   not null  date
```

Database triggers should be created on the tables to fill these columns. The following code can be used as an example for the trigger functionality necessary:

```
CREATE OR REPLACE TRIGGER LGIS_IV_BR_IUD_TRG
BEFORE INSERT OR UPDATE
ON LGIS_INSTANCE_VALUES
FOR EACH ROW
DECLARE
BEGIN
  if inserting then
    :new.create_userid:= user;
```

```

:new.create_timestamp:= sysdate;
:new.update_userid := user;
:new.update_timestamp := sysdate;

if :new.identifier is null then
  select prt_staffs_seq.nextval into :new.identifier from dual;
end if;
elsif updating then
:new.update_userid := user;
:new.update_timestamp := sysdate;
end if;
END;

```

It is the Ministry standard that all audit columns be populated in this fashion (at the Server instead of the client).

It is the Ministry standard that all surrogate key columns be defaulted, if null, in this fashion (at the Server instead of the client).

6.2.6 Journal Tables

The four audit columns (described in the previous section) provide only a basic historical audit. In some cases, a table may require a more sophisticated mechanism in order to keep a complete history of the changes to some tables. This is done by holding each updated copy of a row in a separate Journal table, with user and timestamp information added

A journal table is a database table that is used to record details about each row that is inserted, updated or deleted in the associated table. This is specified in the Table Definition under the Journal property:

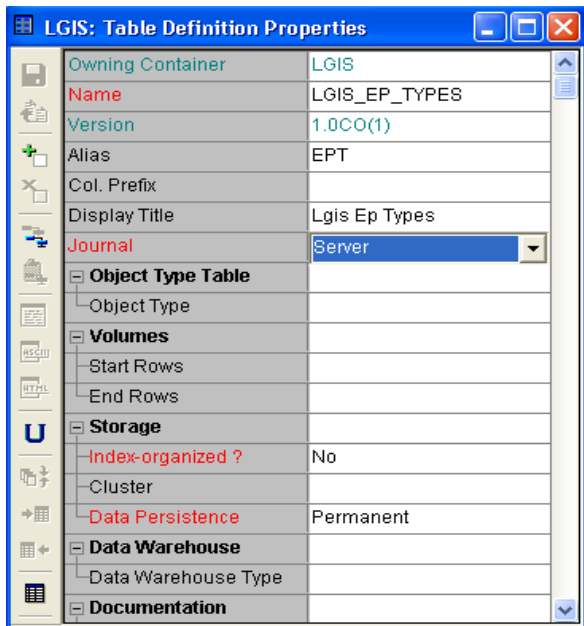


Figure 13: Journal Table

The name of the generated journal table is <table_name>_JN. The journal table is a duplicate of the base table but has six additional columns, prefixed by JN_, to maintain transaction information. These columns are described below:

Journal Column Name	Type of information recorded
JN_OPERATION	Type of transaction performed: INSERT, UPDATE or DELETE.
JN_ORACLEUSER	Name of the Oracle user who performed the transaction.
JN_DATETIME	Date and time the transaction was performed.
JN_APPLN	Name of the application in which the transaction was performed.
JN_NOTES	Notes associated with the transaction.
JN_SESSION	Identifying number of the auditing session for that user.

The journal columns can be maintained in one of the following ways, specified when you create the journal table:

- via generated Table API triggers
- via client calls to generated Table API procedures
- via client side code

The Ministry standard is to use database triggers to maintain journal tables, and does not support the use of any client-side journaling code.

Note: The table being journalled must have a non-updateable primary key, so that each row in the journal table can be traced back to the original row in the source table.

Note: There is no means of recording a storage clause against a journal table. However, by reverse engineering these tables into the repository, you may record this information against the resulting table definitions.

6.2.7 Databases

An Oracle Database and appropriate application level users and roles will need to be defined before running the Database Design Transformer or any of the module or DDL generators.

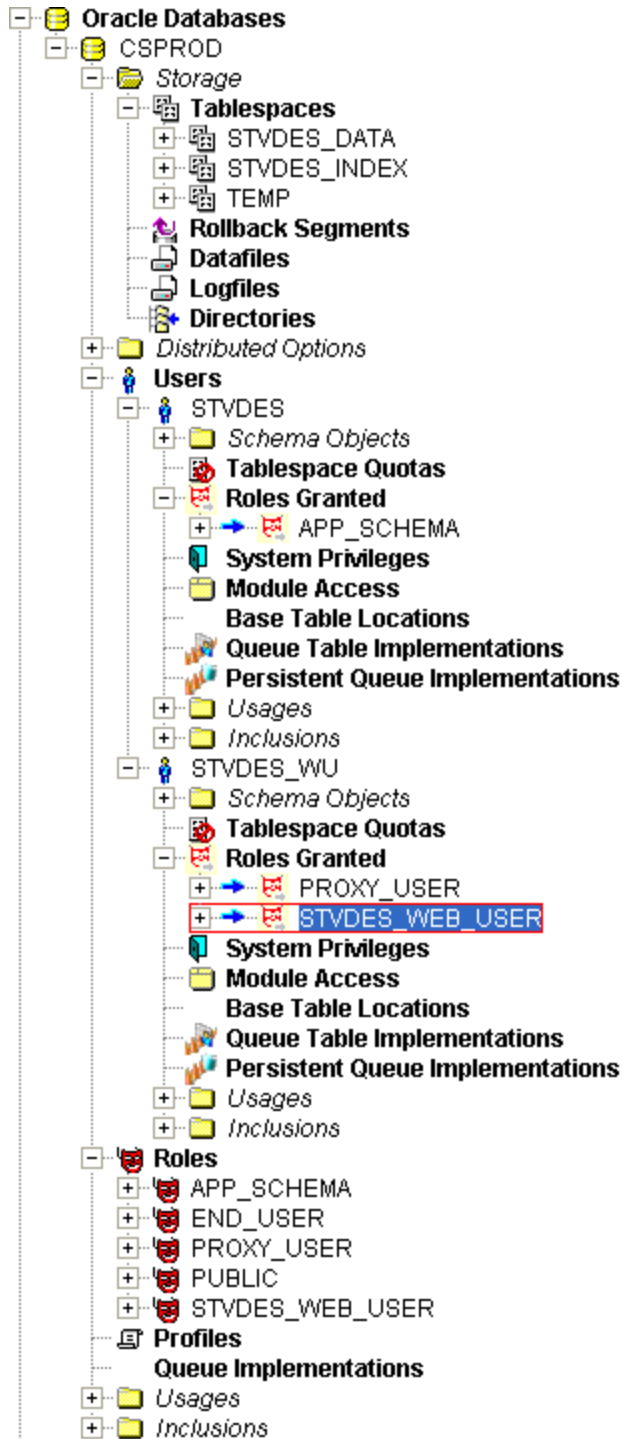
Property	Rule	Req?
Database Name	<ul style="list-style-type: none"> • Should be set to “CSPROD” as a default Target. No SYSTEM level scripts will be generated by the developer using this value, but it is useful metadata for the application to maintain. 	Y
Oracle Version	<ul style="list-style-type: none"> • should be set to Oracle 10g for new development 	Y
Complete ?	<ul style="list-style-type: none"> • must be Yes 	Y

The following screenshot taken from the Designer RON shows the Oracle Database Node and Users and “System” and “Application” level Roles which are mandatory for every application container. It is not necessary to define the underlying system privileges that are granted to the APP_SCHEMA, PROXY_USER or END_USER roles. At this time the ministry does NOT require that end users be modeled in the Designer container (eg. JSMITH). Therefore the END_USER role will not be referenced further in this discussion.

The granting of system privileges to the APP_SCHEMA and PROXY_USER System Roles is the responsibility of the CD/TCA DBA. System privileges allow resource access by the user to Oracle resources and as such are strictly controlled by the DBA group. Database object privileges are granted through separate Application roles and are the responsibility of the developer.

In the following example the “STVDES” user is the schema level user and is granted the System role APP_SCHEMA and the STVDES_WU user is the proxy user and is granted the System role

PROXY_USER and the Application role STVDES_WEB_USER. It is necessary for the developer to create the “Application” level Roles that will enable access to data structures within the schema through object privileges. Please refer to Section 7.2.2. for a more detailed explanation of System and Application level roles.



6.2.8 Tablespaces

Tablespaces should be defined in the Designer tool. In order to define the application schema owner as an Oracle Database User, the system's temporary tablespace will also need to be defined; the Ministry standard name for this tablespace is TEMP.

It is a Ministry standard is to define (at least) two tablespaces, one for tables and the second for indexes.

The tablespace names should be in the form <application_short_name>_DATA and <application_short_name>_INDEX, e.g.:

- LGIS_DATA
- LGIS_INDEX

Property	Rule	Req?
Database	<ul style="list-style-type: none"> • should be set to a database that has been defined 	Y
Name	<ul style="list-style-type: none"> • <name>_DATA for 'tables' tablespace • <name>_INDEX for 'indexes' tablespace • TEMP for temporary tablespace 	Y
Complete ?	<ul style="list-style-type: none"> • set to Yes 	Y
Online ?	<ul style="list-style-type: none"> • set to Yes 	Y
Datafiles Node	<ul style="list-style-type: none"> • This allows datafiles to be defined for tablespaces. These data files must be defined first in the Datafiles section. 	Y
Full Pathname	<ul style="list-style-type: none"> • must be defined in the Datafiles section 	Y
Autoextend ?	<ul style="list-style-type: none"> • must be set to No (unless otherwise approved by the Ministry DBA) 	Y

6.2.9 Datafiles

Data files should be defined for all the _DATA and _INDEX tablespaces described in the Designer tool (e.g.: with *Complete?* =Yes).

The Ministry uses a standard directory convention (based on Oracle's OFA) on all servers for datafiles, always in the format of E:\ORA_DB_FILES\<SID>\DATA_LGIS_01.DBF.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> • must follow the Ministry standard naming convention (based upon OFA), e.g. DATA_LGIS_01.DBF 	Y
Full Pathname	<ul style="list-style-type: none"> • must follow the Ministry standard naming convention (based upon OFA), e.g. . E:\ORA_DB_FILES\LGIS\ 	Y
Reuse ?	<ul style="list-style-type: none"> • must be No 	Y
File Size	<ul style="list-style-type: none"> • should be the initial datafile size (in units defined below) • refer to the <i>Database Table and Index Size Estimates</i> report to get datafile size estimates 	Y
Unit	<ul style="list-style-type: none"> • either KILOBYTES or MEGABYTES 	Y
Autoextend ?	<ul style="list-style-type: none"> • must be No 	Y

6.2.10 Tables

Table definitions can be generated directly from the entity relationship model using the *Database Design Transformer*, or can be created manually using the Server Model Diagrammer or Design Editor.

Note: When using the Server Model Diagrammer, viewing properties via the Property Palette show more information than using the Dialog Palette. Many of the following properties are visible only through the Property Palette.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the table name should conform to the naming standards presented in the Object Naming Conventions section must be prefixed with the Application Short Name and an underscore, e.g.: LGIS_FORMS cross-reference table names must be suffixed with _XREF code tables must be suffixed with _CDS table names should be kept as generated from the associated Entity Plural Name. If a table is defined manually, the table name should be plural 	Y
Alias	<ul style="list-style-type: none"> if a table definition is generated using the DDT, the default is the short name of the corresponding Entity; otherwise an appropriate short name should be entered <p>Note: The table alias is used when generating default index names for the table. It is also used to create block names during Forms generation.</p>	Y
Col Prefix	<ul style="list-style-type: none"> should be left blank 	
Display Title	<ul style="list-style-type: none"> will be used by the module generators to create a default title 	Y
Volumes	Note: These fields are automatically populated from the corresponding fields for the associated entity. These fields are vitally important in systems design to allow for adequate sizing estimates.	
Start Rows	<ul style="list-style-type: none"> estimated number of rows when the table is initially loaded 	Y
End Rows	<ul style="list-style-type: none"> estimated number of rows at the end of 3 years 	Y
Documentation		
Comment	<ul style="list-style-type: none"> becomes the table comment when the table is built <p>Note: It is often overlooked to populate these comments, but they must be filled prior to the Build Phase, in order for the DDL to contain these comment commands.</p>	Y
Description	<ul style="list-style-type: none"> should be described from the users perspective and provide examples where possible 	Y
Notes	<ul style="list-style-type: none"> this should contain any notes about the table 	N
User/Help Text	<ul style="list-style-type: none"> contains the User Help Text associated with the table 	Y
Triggers	<ul style="list-style-type: none"> This section ties the trigger to a specific table. 	
Name	<ul style="list-style-type: none"> the trigger name should conform to the naming standards presented in the Object Naming Conventions section must contain application short name prefix (eg. LGIS) must contain the table alias must be suffixed with _<type>_TRG Ministry standard is: <appl. prefix>_<table_alias>_<B/A><R/S>_<I/U/D>_TRG Note: B/A is Before/After, R/S is Row/Statement, IUD is Insert / Update / Delete. For example, LGIS_IV_BR_IUD_TRG is a trigger on the LGIS_INSTANCE_VALUES table, triggered Before Row upon the operations Insert, Update and Delete 	Y
Purpose	<ul style="list-style-type: none"> describes why this trigger is needed 	Y
PL/SQL Definition	<ul style="list-style-type: none"> PL/SQL definition that holds the code chosen from a picklist 	Y
Complete ?	<ul style="list-style-type: none"> Yes means the Generate Database from Server Model utility will create 	Y

Property	Rule	Req?
	the trigger	
Enabled ?	<ul style="list-style-type: none"> Yes means the trigger will be enabled when it is created with the Generate Database from Server Model utility 	Y
Trigger		
Time	<ul style="list-style-type: none"> controls whether the trigger fires before or after the triggering event 	Y
Level	<ul style="list-style-type: none"> controls whether the trigger is at the row level or the statement level 	Y
Insert ?	<ul style="list-style-type: none"> Yes means the trigger fires on insert 	
Delete ?	<ul style="list-style-type: none"> Yes means the trigger fires on delete 	
Update ?	<ul style="list-style-type: none"> Yes means the trigger fires on update 	
Trigger When Condition	<ul style="list-style-type: none"> optional defines the when-clause for the trigger 	N

6.2.11 Columns

Column definitions for tables are generated directly from attributes of the corresponding entity using the Database Design Transformer or can be manually defined with the Server Model Diagrammer.

Please refer to the [Standard Table Enhancements](#) section for information on suggestions for adding certain columns to all 'primary' tables.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the column name should conform to the naming standards presented in the Object Naming Conventions section column names should not be prefixed names should be singular; the default column name generated using the Database Design Transformer is the name of the corresponding attribute if a system generated primary key is used, the column name should be suffixed by <code>_ID</code> if a Super-type (Single Table) implementation of Sub-types is chosen, specify the name of the discriminator column as: <code><entity-short-name>_TYPE</code>; this is the default name generated by the Database Design Transformer 	Y
Sequence in Table	<ul style="list-style-type: none"> specified the column sequence within the table primary key fields should be first; in the case of a multiple column primary key, the columns should follow the order in the primary key NOT NULL columns must be listed before columns that allow nulls Long VARCHAR2 columns are next LONG column are next audit columns (<code>create_userid</code>, <code>create_timestamp</code>, etc.) must be last 	Y
Complete ?	<ul style="list-style-type: none"> Should be Yes 	Y
Domain	<ul style="list-style-type: none"> Name of the Domain if the column is defined with a domain type 	Y
Scalar	Note: If a domain is not used to define the data type, the properties in the Scalar Group must be entered. If a domain is used for a column, these properties are filled automatically.	
Datatype	<ul style="list-style-type: none"> Should use datatypes explicitly supported by the target database 	Y
Average Length	<ul style="list-style-type: none"> used for sizing estimates 	Y

Property	Rule	Req?
Maximum Length	<ul style="list-style-type: none"> used for sizing estimates 	Y
Decimal Places	<ul style="list-style-type: none"> mandatory for datatypes of NUMBER or DECIMAL 	N
Definition		
Optional ?	<ul style="list-style-type: none"> No will generate a NOT NULL column; Yes will generate a NULL column 	Y
Uppercase ?	<ul style="list-style-type: none"> should be considered for CHAR or VARCHAR2 datatypes used by the Forms generator for a field mask 	N
Default Value	<ul style="list-style-type: none"> Should not be used if an column is optional Must be the same datatype as the column <p>Note: Use of defaults must be examined carefully, as default values may lead the inexperienced user to enter erroneous data</p>	N
Sequence	<ul style="list-style-type: none"> name of a sequence if one is used to populate the column 	N
Volumes	<p>Note: The Volume Group estimates the percentage of columns in the table that will contain values. If a column is NOT NULL (Optional? = No) then these volumes will be 100 (%). These values are important for sizing estimates.</p>	
Initial Volume	<ul style="list-style-type: none"> the percentage of columns that will contain values at initial data load 	Y
Final Volume	<ul style="list-style-type: none"> the average percentage of columns that will contain data when the system is active 	Y
Default Display	<p>Note: The Display Group is used to set properties that will be used as default values when creating Forms, Reports and Web modules based on the table and column. For that reason, it is worthwhile filling these properties in.</p> <p>It is the Ministry standard that these values are filled in prior to creating the Module Components, as these values are read only upon creation of the Table/Column usages</p>	
Display ?	<ul style="list-style-type: none"> if Yes, then this column will be displayed 	N
Display Type	<ul style="list-style-type: none"> mandatory if Display? = Yes usually the same as the column datatype 	N
Alignment	<ul style="list-style-type: none"> mandatory of Display? = Yes 	N
Display Length	<ul style="list-style-type: none"> should be specified if Display? = Yes specifies the display length in characters 	N
Display Height	<ul style="list-style-type: none"> should be specified if Display? = Yes specifies the display height in characters 	N
Display Sequence	<ul style="list-style-type: none"> should be specified if Display? = Yes 	N
Format Mask	<ul style="list-style-type: none"> applicable only if Display? = Yes sets the default display format, e.g. YYYY-MM-DD 	N
Prompt	<ul style="list-style-type: none"> mandatory if Display? = Yes specifies the boilerplate text prompt for the column (note that the generator will append a colon ':' onto the prompt). E.g.: Date Received: 	N
Help		
Hint	<ul style="list-style-type: none"> mandatory where Display? = Yes should contain business terms where application <p>Note: During Forms generation, this field is used to provide hint text to the user and is displayed on the message line of the form. This field is populated with comment text defined for an attribute if the Database Design Transformer was used to generate the column.</p>	N

Property	Rule	Req?
Documentation		
Comment	<ul style="list-style-type: none"> • specifies a column comment <p>Note: It is often overlooked to populate these comments, but they must be filled prior to the Build Phase, in order for the DDL to contain these comment commands.</p>	Y
Description	<ul style="list-style-type: none"> • must be clear and concise • must be meaningful to non-application personnel 	Y
Notes	<ul style="list-style-type: none"> • any column-specific notes should be included here 	N

6.2.12 Views

A view defines a "window" on one or more tables through which the table information may be queried or changed. Views are defined to simplify complex queries and are often created for security purposes. By creating a view, user access may be restricted to a subset of columns in a table, thus protecting sensitive information by controlling data access at the object level.

View definitions can be created as either free-form text, or explicitly identifying each column. Although free-form text is often easier, it does not document the source columns used in the view as clearly. **It is the Ministry standard to explicitly declare the base tables and columns.** The only exceptions are views which cannot be defined declaratively, such as those using set operators (e.g. union, minus)

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the view name should conform to the naming standards presented in the Object Naming Conventions section must be prefixed with the Application Short Name and an underscore and suffixed with “_VW”, e.g.: LGIS_VALUES_DATA_VW the view name should be descriptive, as well as indicate which tables are used within the view 	Y
Alias		Y
Col. Prefix	<ul style="list-style-type: none"> should be left blank 	N
Display Title		Y
SQL		
Free Format Select Text ?	<ul style="list-style-type: none"> should be set to No except where the view includes tables from other databases and/or other applications that are not modeled in the repository if a view is reverse engineered, then this will be Yes <p>Caution: If you change the value of this property from Yes to No, you will lose any text that you have entered in the Select Text property.</p>	N
Select Text	<ul style="list-style-type: none"> if Free Format Select Text? is No, then this text is Read Only and will contain the 'base columns' from the various tables that the view is created from if Free Format Select Text? is Yes, then this text will contain the view definition 	N
Where / Validation Condition	<ul style="list-style-type: none"> applicable only with Free Format Select Text? = No contains the 'where' clauses for the view 	N
Documentation		
Comment		Y
Description	<ul style="list-style-type: none"> contains a description of the view 	Y
Notes	<ul style="list-style-type: none"> should contain any special notes about the view 	N
Base Tables	<ul style="list-style-type: none"> defines the tables on which the view is based 	Y
Columns	<ul style="list-style-type: none"> defines the column names as presented with the view ensure that the sequence numbers for the columns are correct 	Y
Select Column	<ul style="list-style-type: none"> 	
Base Column	<ul style="list-style-type: none"> if the view column is based on a table, then this will be the column_name from that table if the view column is based on a function or an expression, then this will be blank 	N

Select Text	<ul style="list-style-type: none"> if the view column is based on a table, then this will be the table_alias.column_name from that table if the view column is based on a function or an expression, then this will be that function or expression. e.g.: to_char(sysdate,'Month') 	N
-------------	--	---

6.2.13 Sequences

A sequence number generator (often just called a sequence) can be used to automatically create unique integer numbers for primary keys. This primary key is called a surrogate (or artificial) key, and has no meaning in the sense of the Business Requirements.

Sequence number generators improve performance in a multi-user environment by avoiding lock conflicts at the cost of potential gaps in the sequence. These sequence numbers are generated independently of tables. The same sequence may be used for one or more tables, although multiple table usage of a single sequence is not recommended. Sequences may be defined using the Repository Object Navigator. The Database Design Transformer will generate a surrogate key and its associated sequence in situations where a primary key for an entity was not provided.

Sequences can be implemented in one of two ways: as an Oracle sequence, or as a Code Control sequence. An Oracle sequence is faster and simpler, but the Code Control sequence approach is better suited to situations where the values are required to be continuous. Please consult the Oracle technical documentation for a detailed description of these two approaches.

The Ministry standard is to always use Oracle sequences

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the sequence name must conform to the naming standards presented in the Object Naming Conventions section must be prefixed with the table name (which itself is prefixed with the Application Short Name) and an underscore, e.g.: LGIS_LVT_PK_SEQ must be suffixed with _SEQ if multiple sequences are required for a single table, the sequence name should be suffixed with _SEQ# where the '#' increments sequentially 	Y
Code Control?	<ul style="list-style-type: none"> must be Oracle sequence 	Y
Documentation		
Comment		Y
Description	<ul style="list-style-type: none"> contains a description of the sequence 	Y
Notes	<ul style="list-style-type: none"> contains any special notes on the sequence 	N

6.2.14 Constraints

Constraint definitions are generated automatically by the Database Design Transformer from relationships, primary keys, unique identifier entries and attribute allowable value lists. There are four types of constraints defined in Designer:

Type	Description
Primary Key	<ul style="list-style-type: none"> a column or a set of columns in a table that will always be unique within the table. All columns within the primary key must be mandatory. The primary key may be referenced by foreign keys in join tables.
Unique Keys	<ul style="list-style-type: none"> a column or set of columns in a table that will always be unique within the table. Columns within a unique key may be optional (Note: this is a departure from ANSI SQL standards). A table may have zero, one, or many unique key constraints.

Type	Description
Foreign Keys	<ul style="list-style-type: none"> a column or set of columns which reference a corresponding column or set of columns in another table through primary or unique keys. The columns in a foreign key must have the same relative sequencing of the columns in the associated primary or unique key.
Check Constraints	<ul style="list-style-type: none"> a condition or expression that applies to a table restricting the data that can be entered. Check constraints may be used to enforce such things as: <ul style="list-style-type: none"> enforcing ranges for specific columns inter-column dependencies (e.g. column_a column_b or if column_a is null then column_b must not be null).

6.2.14.1 Primary Key Constraints

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the constraint name should conform to the naming standards presented in Object Naming Conventions the default name, as generated by the Data Design Transformer, should be kept as-is. This ensures that that it: <ul style="list-style-type: none"> contain the table alias suffixed with <code>_PK</code> prefixed with the Application Short Name and an underscore, e.g.: <code>FNI_TREATIES_PK</code> 	Y
Complete ?	<ul style="list-style-type: none"> set to Yes if the constraint should be built by the <i>Generate Database from Server Model</i> utility 	Y
Enable ?	<ul style="list-style-type: none"> set to Yes if the constraint should be automatically enabled when it is built 	Y
Update ?	<ul style="list-style-type: none"> if set to Yes, then it allows the Primary Key to be updated 	N
Validation		
Validate in	<ul style="list-style-type: none"> must be Server or Both 	Y
Error Message	<ul style="list-style-type: none"> suggested, unless using an Error Message table specifies a text message to be hard-coded in a generated module if the constraint fails 	N
Documentation		
Description	<ul style="list-style-type: none"> a brief description of the primary key constraint 	Y
Notes	<ul style="list-style-type: none"> any notes for the constraint 	N
Columns	Note: These entries determine the columns in the Primary Key; ensure that their order is correct.	
Column	<ul style="list-style-type: none"> the column name is specified via a picklist 	Y
Sequence in Key	<ul style="list-style-type: none"> controls the order of the columns within the primary key 	Y

6.2.14.2 Unique Key Constraints

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the constraint name should conform to the naming standards presented in the Object Naming Conventions section the default name, as generated by the Data Design Transformer, should be kept as-is. This ensures that that it: <ul style="list-style-type: none"> contain the table alias suffixed with <code>_UK</code> prefixed with the Application Short Name and an underscore, e.g.: <code>FNI_TREATIES_UK1</code> 	Y

Property	Rule	Req?
Complete ?	<ul style="list-style-type: none"> set to Yes if the constraint should be built by the Generate Database from Server Model utility 	Y
Enable ?	<ul style="list-style-type: none"> set to Yes if the constraint should be automatically enabled when it is built 	Y
Update ?	<ul style="list-style-type: none"> if set to Yes, then it allows the Unique Key to be updated 	
Validation		
Validate in	<ul style="list-style-type: none"> must be Server or Both 	Y
Error Message	<ul style="list-style-type: none"> suggested specifies a text message to be displayed in a generated module if the constraint fails 	N
Documentation		
Description	<ul style="list-style-type: none"> a brief description of the constraint 	Y
Notes	<ul style="list-style-type: none"> any notes specific to the constraint 	N
Columns	<ul style="list-style-type: none"> These entries determine the columns in the Unique Key; ensure that their order is correct. 	
Column	<ul style="list-style-type: none"> the column name is specified via a picklist 	Y
Sequence in Key	<ul style="list-style-type: none"> controls the order of the columns within the unique key 	Y

6.2.14.3 Foreign Key Constraints

Property	Rule	Req?
Join Table	<ul style="list-style-type: none"> name of the table that foreign key constraint references 	Y
Name	<ul style="list-style-type: none"> the default name, as generated by the Data Design Transformer, should be kept as-is. This ensures that that it: <ul style="list-style-type: none"> contain the table alias suffixed with _FK the constraint name should conform to the naming standards presented in the Object Naming Conventions section table aliases should be used (e.g. emp_dept_fk is a foreign key constraint on the employees table) <p>Note: In rare cases, the constraint name must be unique within the first 21 characters, due to a known bug in the Forms Generator. In these cases, it will be necessary to modify the constraint name manually, after the DDT.</p>	Y
Complete ?	<ul style="list-style-type: none"> set to Yes if the constraint should be built by the Generate Database from Server Model utility 	Y
Enable ?	<ul style="list-style-type: none"> set to Yes if the constraint should be automatically enabled when it is built 	Y
Mandatory ?	<ul style="list-style-type: none"> Yes indicates that a value is required in the foreign key 	Y
Transferable ?	<ul style="list-style-type: none"> Yes indicates that the foreign key can be updated (transferable); No indicates the foreign key cannot be updated (non transferable). 	Y
Validation		
Validate in	<ul style="list-style-type: none"> must be Server or Both 	Y
Error Message	<ul style="list-style-type: none"> suggested specifies a text message to be displayed in a generated module if the constraint fails 	N
Cascade Rules		
Delete Rule	<ul style="list-style-type: none"> determines what happens when a row in the join table is deleted Cascades: deletes the foreign key in this table when the related row is 	Y

Property	Rule	Req?
	<p>deleted in the join table</p> <p>Restricted: prevents deletion of a row in the join table when a related row exists in this table</p> <p>Nullifies: updates the foreign key in this table as NULL where a row in the related join table is deleted</p> <p>Defaults: updates the foreign key in this table to the specified default value where a row in the related join table is deleted</p>	
Update Rule	<ul style="list-style-type: none"> must be Restricted or the Generate Database from Server Model utility will not build the constraint 	Y
Joining To		
Primary Key Joined to	<ul style="list-style-type: none"> either this field or Unique Key Joined to is mandatory 	N
Unique Key Joined to	<ul style="list-style-type: none"> either this field or Primary Key Joined to is mandatory 	N
Documentation		
Description	<ul style="list-style-type: none"> a brief description of the foreign key 	N
Notes	<ul style="list-style-type: none"> any notes on the foreign key 	N
Columns	Note: These entries determine the columns in the Foreign Key; ensure that their order is correct.	
Column	<ul style="list-style-type: none"> name of the column in this table 	Y
Sequence in Key	<ul style="list-style-type: none"> controls the order of the columns within the foreign key 	Y
Join Column	<ul style="list-style-type: none"> name of the corresponding column in the join table 	Y

6.2.14.4 Check Constraints

Property	Rule	Req?
Name	<ul style="list-style-type: none"> the constraint name should conform to the naming standards presented in the Object Naming Conventions section must contain the table names or aliases should be suffixed with _CHK if multiple check constraints are defined for a table, then they should be suffixed with _CHK# must be prefixed with the Application Short Name and an underscore, e.g.: FNI_REGIONS_CHK 	Y
Complete ?	<ul style="list-style-type: none"> set to Yes if the constraint should be built by the <i>Generate Database from Server Model</i> utility 	Y
Enable ?	<ul style="list-style-type: none"> set to Yes if the constraint should be automatically enabled when it is built 	Y
Error Message	<ul style="list-style-type: none"> Suggested Specifies a text message to be displayed in a generated module if the constraint fails 	N
Validation		
Validate in	<ul style="list-style-type: none"> must be Server or Both 	Y
Comment	<ul style="list-style-type: none"> describes the check constraint 	Y
Where/ Validation Condition	<ul style="list-style-type: none"> contains the constraint text for the check constraint, e.g.: RECEIVE_DATE <= SYSDATE 	Y
Documentation		

Property	Rule	Req?
Description	<ul style="list-style-type: none"> a brief description of the constraint 	N
Notes	<ul style="list-style-type: none"> any notes on the constraint 	N

6.2.15 Indexes

Indexes are used for two purposes within a relational database management system:

- to provide quick access to rows in a table
- to enforce uniqueness of one or more columns within a table

Applications should be 'tuned' for performance by creating indexes on columns or groups of columns which are frequently queried. Primary key constraints are implemented through unique indexes, as are unique key constraints. Foreign key constraints also generate indexes (non-unique) to enhance performance.

If a table is small, typically less than 2 * block size, it is often more efficient not to index the table. This is because if there is an index, it will take at least one read (of one block) to get it, and then a second read to get the data; if the whole table can be read into memory in two reads, then there is no performance gain through the index.

Care should be taken to remove redundant indexes. For example, if an composite index already exists for columns (col_a, col_b, col_c), then queries on (col_a) and (col_a, col_b) will use this index, so there is no need to define an additional index. However, this index will not be used with queries on (col_a, col_c) or (col_b); in such situations, additional indexes may be required if these are common queries.

As the number of indexes on a table is increased, the insert and update performance usually decreases while the select performance increases.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> must be suffixed with <code>_I</code> <ol style="list-style-type: none"> Foreign Key indexes must be suffixed with <code>_FK_I</code> Unique indexes should be suffixed with <code>_UK_I</code> the index name should conform to the naming standards presented in the Object Naming Conventions section table aliases should be used (e.g. <code>emp_dept_fk_i</code> is a foreign key index from the employees table to the departments table) indexes supporting foreign key constraints should have the same prefix (e.g. index <code>t_mbr_c_thm_fk_i</code> supports the foreign key constraint <code>t_mbr_c_thm_fk</code>) 	Y
Index Type ?	<ul style="list-style-type: none"> either Unique or Not unique 	Y
Complete ?	<ul style="list-style-type: none"> set to Yes if the index should be built by the <i>Generate Database from Server Model</i> utility 	Y
Foreign Key	<ul style="list-style-type: none"> mandatory if this index is for a foreign key relationship 	N
Documentation		
Description	<ul style="list-style-type: none"> a brief description of the index 	N
Notes	<ul style="list-style-type: none"> any notes specific to the index 	N
Columns	Note: This group defines the columns in the index.	
Column	<ul style="list-style-type: none"> name of the column included in the index 	Y
Usage Sequence	<ul style="list-style-type: none"> sequence of the column within the index 	Y

6.2.16 PL/SQL Definitions

6.2.16.1 Best Practices for Coding

The intent of this section is to describe best practices when coding PL/SQL in the Oracle database. This applies only to new applications, and as per Ministry standards, the PL/SQL must be documented in the Designer Repository.

These best practices are simple and practical guidelines for developers. The objective of these best practices is to produce PL/SQL code that is understandable and maintainable.

Much of the content in this section is derived from Steven Feurestein's book "Oracle PL/SQL Programming"² and on-line articles.

6.2.16.1.1 *Follow Ministry Coding Standards*

In addition to the Designer-specific guidelines in this document, PL/SQL Developers should follow the Ministry's "Standardized Coding Practices".

Some of these practices, such as Variable Type Prefixes, are not relevant to PL/SQL but the following do apply:

- Variable Scope and Usage Prefixes (e.g. g_, st_, v_, etc.)
- Variable Name Capitalization (i.e. camelCase) ; although camel_Case (with underscores) are also permissible if this aids readability
- Constants (i.e. all uppercase)
- Comments (comment code blocks that are large or complex)
- Readability (indenting code and using whitespace); there are no explicitly rules to indentation and whitespace (i.e. leading tab characters, or 'four spaces'), so the key best practice here is consistency within the application.

6.2.16.1.2 *Use Packages instead of stand-alone procedures or functions*

Organize all PL/SQL code in well-named packages, which has the following advantages over stand-alone modules:

- breaks the dependency chain in that there are no cascading invalidations when you install a new package body. If you have procedures that call other procedures, then compiling one will invalidate your code.
- supports encapsulation -- allows you to write modular, easy to understand code -- rather than monolithic, hard to read procedures
- increases your namespace measurably. Package names have to be unique in a schema, so

² *Oracle PL/SQL Programming, Third Edition*, by Steven Feurstein with Bill Pribyl, 2002, O'Reilly & Associates, Inc.

you can have many procedures across different packages with the same name without colliding.

- supports overloading
- supports session variables when you need them
- promote overall good coding techniques by logically grouping related code

6.2.16.1.3 Use Anchored Declarations

In retrieving the value of a column, it is possible to declare the variables as a generic numeric or character (i.e. `party_id IN number`). However, it is better to anchor the declaration to the underlying datatype in the column (i.e. `party_id party.id%TYPE`).

This ensures that the variable will be able to hold the value, even if the column's datatype changes in the future.

6.2.16.1.4 Avoid Repetition of SQL Code

Instead of embedding native SQL code everywhere, the SQL statements should be encapsulated into a central PL/SQL function. Typically, there is a central function for every table, or set of tables acting as a common interface (e.g. `PARTY`, `NAME`, `ADDRESS` are normalized tables that often are queried at the same time). This also applies to `SELECT`'s, `INSERT`'s, `UPDATE`'s, and `DELETE`'s.

This "central PL/SQL function" should be a pre-built, pre-tested module that allows it to be 'written once, used often'.

6.2.16.1.5 Avoid excessively long procedures or functions

Use local procedures and functions to hide logic from the 'mainline' portion of the module, breaking up the larger problem into smaller, more manageable problems.

This will result in more small, focused packages. Each of these will have executable sections that are smaller, readable and less than 75 lines from `BEGIN` to `END`.

6.2.16.1.6 Use Bind Variables instead of string literals

Instead of concatenating strings together (i.e. `user = '&User'`) to build a SQL query, it is better to use bind variables (i.e. `user = :user`). This is important in terms of scalability and performance, but it is especially important to help prevent against SQL injection attacks.

For more details on SQL injection, see "*Effective Oracle by Design*" by Thomas Kyte (ISBN number 0072230657).

6.2.16.1.7 Formalize Unit Testing

Unit Testing should incorporate documented test cases, and any bugs discovered should reference this test case. It is recommended that a testing framework be established and used. Two examples are Oracle Unit

Tester (<http://www.ounit.com/>) and Unit Testing Framework for PL/SQL Developers (<http://oracle.oreilly.com/utplsql/>).

6.2.16.2 Function, Packages, Procedures, and Cursors

PL/SQL Definitions are components of the application that are stored in the database. Naming conventions for triggers are described in the appropriate Database Design section.

Names for these:

- should be no more than 30 characters long
- must contain the Application Short Name and an underscore as a prefix, unless the program is inside a package
- must be suffixed with an underscore and a type (unless it is inside a package):
 - _PKG for packages
 - _F for functions
 - _P for procedures
 - _CSR for cursors
- the centre component of the name is a free format descriptive identified that follows general naming conventions where possible (see Object Naming Conventions for database objects)
- example: STVS_STANDARD_PKG- a package of functions.

PL/SQL Definitions can be declarative (e.g. every variable, constant, argument, etceteras is defined and recorded as an object in the repository) or free format (e.g. all code and variable declarations are recorded as part of a multi-line text property in the repository).

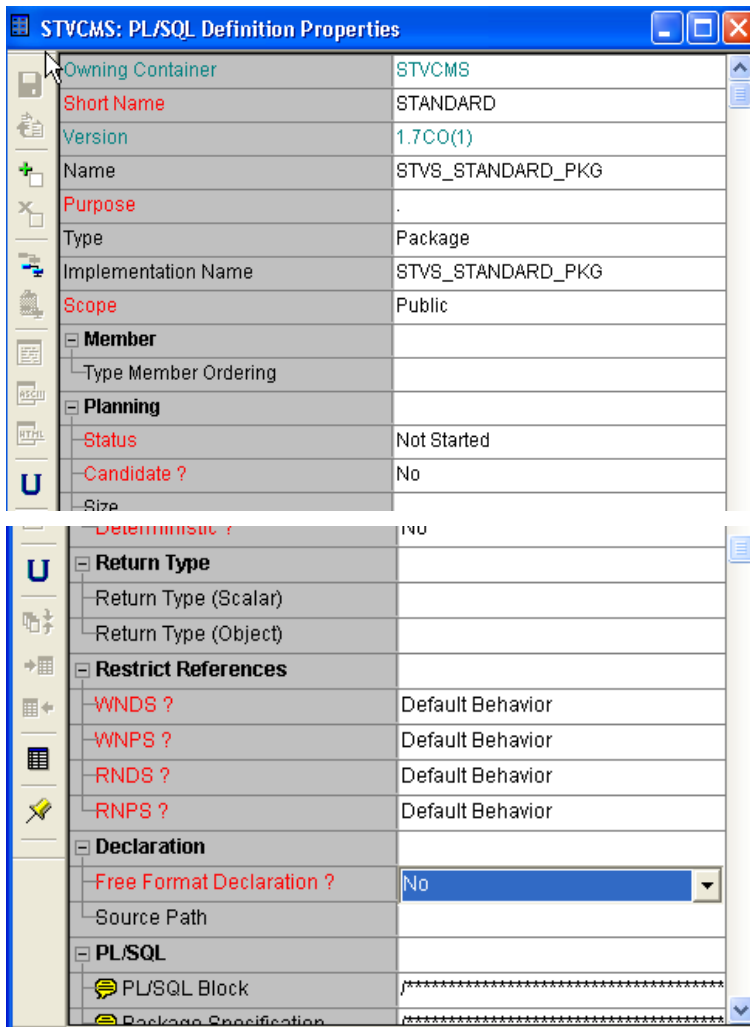


Figure 14: Database Package

It is a Ministry standard to define packages, and its procedures declaratively (e.g. *Free Format Declaration?* = No). Standalone procedures and functions should not be used as all procedures and functions should be encapsulated in Packages.

For other definitions (e.g. standalone procedures, functions), it is recommended to declare them declaratively, although free format definitions will be accepted.

The Ministry will not accept PL/SQL definitions created by pointing to a file on disk (e.g. having a file name in the Source Path property).

When selecting repository objects to generate (Generate Database from Server Model utility), the defined functions, packages, and procedures are available for generation. They are generated to files on disk, with the following file name suffixes:

- <File_Prefix>.fnc (functions)
- <File_Prefix>.pks (package specifications)
- <File_Prefix>.pkb (package body specifications)
- <File_Prefix>.prc (procedures)

Note: Included procedures and functions (e.g. inside a package) are always generated in the context of the owning package definition and are not available for selection under the Procedure and Function nodes when you generate the package.

6.2.16.3 Triggers

When defining the trigger, everything between the BEGIN and END statements goes into the PL/SQL Block. For example, the following trigger would have the lower-case text in the PL/SQL block; the capitalized information is generated automatically from other properties:

```
CREATE OR REPLACE TRIGGER LGIS_IV_BR_IUD_TRG
BEFORE INSERT OR UPDATE
ON LGIS_INSTANCE_VALUES
FOR EACH ROW
DECLARE
BEGIN
  if inserting then
    if :new.id is null then
      select prt_staffs_seq.nextval into :new.id from dual;
    end if;
    :new.create_userid := user;
    :new.create_timestamp := sysdate;
  elsif updating then
    :new.update_userid := user;
    :new.update_timestamp := sysdate;
  end if;
END;
```

Property	Rule	Req?
Short Name	<ul style="list-style-type: none"> the trigger name should conform to the naming standards presented in the Object Naming Conventions section must contain application short name prefix (eg. LGIS) must contain the table alias must be suffixed with _<type>_TRG Ministry standard is: <appl. prefix>_<table_alias>_<B/A><R/S>_<I/U/D>_TRG <p>Note: B/A is Before/After, R/S is Row/Statement, IUD is Insert / Update / Delete. For example, LGIS_IV_BR_IUD_TRG is a trigger on the LGIS_INSTANCE_VALUES table, triggered Before Row upon the operations Insert, Update and Delete</p>	Y
Name	<ul style="list-style-type: none"> descriptive name 	Y
Purpose	<ul style="list-style-type: none"> short description for the purpose of the trigger 	Y
Type	<ul style="list-style-type: none"> must be set to Trg-Logic 	Y
Implementation Name	<ul style="list-style-type: none"> recommended to leave blank, so that implementation name is derived from the Short Name 	N
PL/SQL		
PL/SQL Block	<ul style="list-style-type: none"> contains the body of the trigger (everything between the BEGIN and END statements - see example) 	Y
Documentation		
Module Generation History	<ul style="list-style-type: none"> must contain dates, names and brief descriptions for each major modification of the module historical information must be maintained 	Y
Release	<ul style="list-style-type: none"> optional 	N

Property	Rule	Req?
Notes	<ul style="list-style-type: none"> any release-specific information for the current version 	
Description	<ul style="list-style-type: none"> optional a brief description of the trigger 	N
Notes	<ul style="list-style-type: none"> any notes about the trigger 	N

6.2.17 Storage Definitions

Storage definitions can be created in the Repository to ensure that similar classes of objects will have similar storage definitions.

Property	Rule	Req?
Storage Label	<ul style="list-style-type: none"> unique name for this storage definition 	Y
Extents		
Initial Extent	<ul style="list-style-type: none"> size of the first extent to be allocated 	Y
Initial Extent Unit	<ul style="list-style-type: none"> units (Kilobytes, Megabytes), with Bytes used if null Ministry standard is to use Megabytes 	Y
Next Extent	<ul style="list-style-type: none"> size of every extent after the initial one 	Y
Next Extent Unit	<ul style="list-style-type: none"> units (Kilobytes, Megabytes), with Bytes used if null Ministry standard is to use Megabytes 	Y
Min Extents	<ul style="list-style-type: none"> initial number of extents to be allocated 	Y
Max Extents	<ul style="list-style-type: none"> total number of extents that can be allocated 	
Unlimited	<ul style="list-style-type: none"> automatically allocate more extents as needed Ministry standard is to set to No 	Y
Percentage Increase	<ul style="list-style-type: none"> percentage by which each following extent will grow over the preceding one must be zero 	Y

6.2.18 Synonyms

Public Synonyms can be defined in the Repository and created with the Generate Database from Server Model utility. The Synonyms Group underneath the object in each of the Modules, Tables, Sequences and Views nodes is used for this.

The Public Synonym Name must match exactly the object name that it references (e.g. LGIS_FIELD_GROUPS public synonym for the LGIS_FIELD_GROUPS table).

6.2.19 Database Object Grants

Privileges on database objects granted to roles must also be captured, documented and maintained in the Designer Repository.

Once the roles and the actual objects exist, the Database Object Grants group for each object is used to grant the specific privileges to the various roles.

Roles must be hierarchical. Therefore one should only grant the additional rights specific to that role to an object.

For example, assume that there are three roles: APPL_ROLE_1, APPL_ROLE_2 and APPL_ROLE_3. APPL_ROLE_1 is granted to APPL_ROLE_2, and APPL_ROLE_2 is granted to APPL_ROLE_3. For a specific table APPL_ROLE_1 needs select, APPL_ROLE_2 needs select and insert, and APPL_ROLE_3 requires select, insert and delete.

Instead of explicitly granting all the rights for the table to each role, the Ministry standard is to:

- SELECT to APPL_ROLE_1
- INSERT to APPL_ROLE_2
- DELETE to APPL_ROLE_3

and ensure that the role hierarchies are defined and granted correctly

A further example follows, from the Oracle 10g Database Security Guide (Figure 21-1):

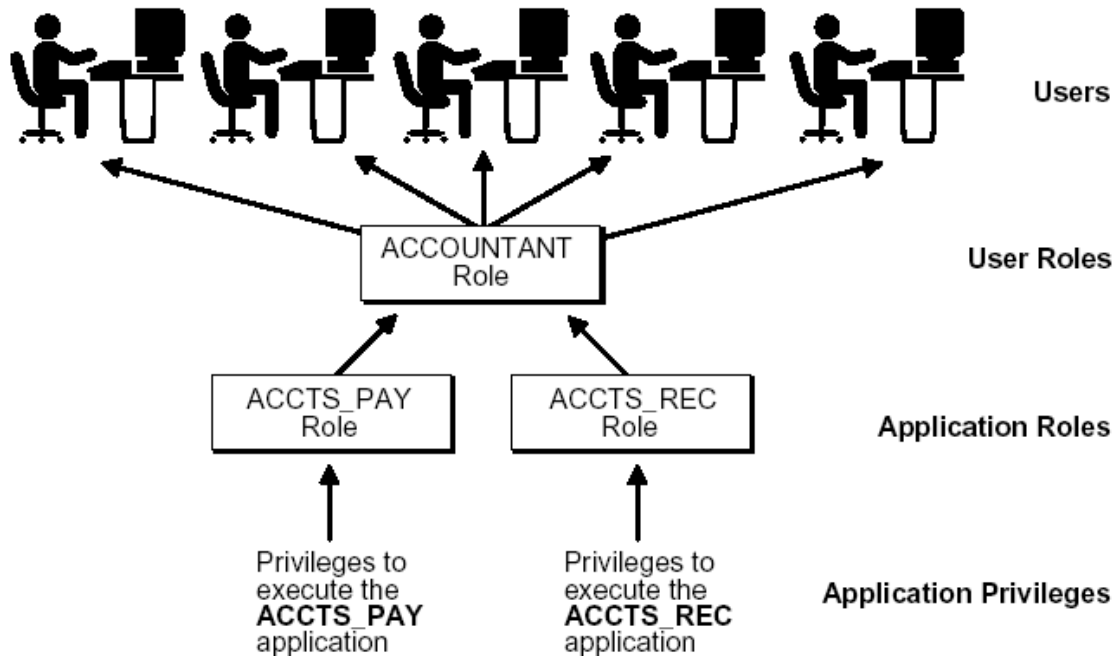


Figure 15: Role Security

It is a Ministry standard that applications set the database role upon user login, and disable the role upon user logout from the application.

This means that the startup module will perform the SET_ROLE command while the ‘user-exit’ condition will explicitly disable the role.

For further details on the modeling of roles and object privileges please refer to Section 6.2.7 – Databases and Section 7.2.2 – Roles.

For further details on application development and Oracle Database Security best practices please refer to the Oracle 10g Database Security guide which can be referenced at:

http://download-west.oracle.com/docs/cd/B14117_01/network.101/b10773/toc.htm

6.2.20 SQL Statement Tuning

6.2.20.1 COST vs. RULE Based Optimization

The ministry does NOT support the use of RULE based optimization in SQL queries. COST based optimization IS the standard for all ministry databases.

The primary reasons for this are that the RULE based optimizer has been de-supported in the Oracle RDBMS (after 9i Release 2), and it is more expensive to maintain queries developed using RULE's due to changes in data content and selectivity over time. As well, a properly designed data structure and efficient SQL will ensure that COST based optimization provides the most efficient data access paths.

6.2.20.2 Explain Plans

Every static multi join query (2 or more tables) must provide EXPLAIN PLAN output. Developers should make available suitable data volumes for testing purposes in order for the explain plan to be properly utilized.

Ensure that prior to generating the explain plans that table and index statistics are updated, as this will influence optimizer behavior. There are standard Oracle supplied packages which simplify this process for developers and the DBA, such as:

```
EXEC DBMS_UTILITY .ANALYZE_SCHEMA ( ' SCHEMA_NAME ' , ' COMPUTE ' );
```

NOTE: For more information on SQL Statement Tuning and general Oracle tuning considerations please reference the Oracle 10g server documentation available from Oracle's technical support website (metalink.oracle.com) and Oracle's technology network site (technet.oracle.com).

6.2.20.3 Embedding of SQL in PL/SQL Code

As discussed in “6.2.16.1.4 Avoid Repetition of SQL Code”, it is a PL/SQL Best Practice to use centralized data access in PL/SQL functions that also includes all of the necessary error handling and optimization logic.

This is in contrast to “SELECT INTO...” statements in multiple places, all accessing the same table or view, but using different variable names or types. This results in excessive parsing and difficulty in optimizing the performance. The database can take advantage of cached statements if the syntax of the statement is exactly the same; this enables more code re-use and better optimization.

6.3 Module Design

The Module Design process involves the definition of modules (programs/program units) to support the business functions of the organization as defined in the Function Modeling phase. These modules may include forms, reports, packages, procedures, functions, triggers and cursors. The definition requirements for triggers have been covered in the Database Design section.

After verifying the completeness of existing business functions and consolidating functionality using the Functional Hierarchical Diagrammer (FHD), the Application Design Transformer (ADT) is used to generate candidate module definitions. These candidate module definitions can be further refined using the tools described below. Upon further refinement of the module definitions, the various generator tools are utilized to generate working modules.

The display characteristics of bound items in modules are inherited from the source column (in the table definition). This is done only at the initial creation of the bound item, and is not automatically kept up to date by Designer (e.g. when the hint text for the column is modified, the modules using this column as a bound item still have the obsolete hint text).

It is the Ministry standard to set the Default Display, Help and Documentation properties at the column level before creating the module components. The following is an example of a column with these properties.

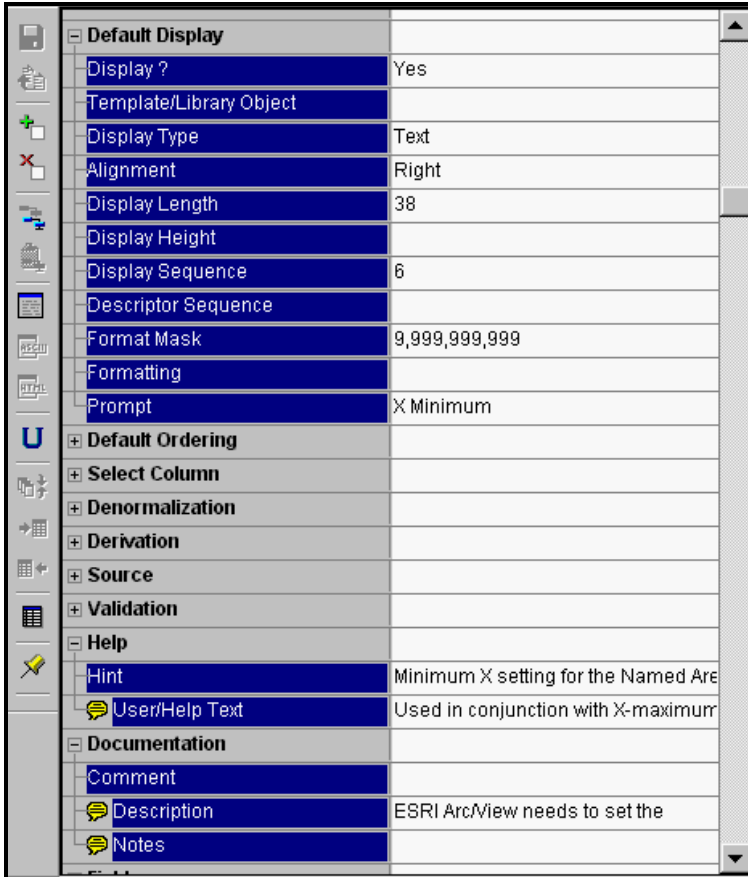


Figure 16: Column Properties

This ensures a consistent look and feel to the screen items. In the case where a column needs a different hint text or display characteristic, this can still be done by over-riding the property in the bound item.

Designer supports four tools that can create or manipulate modules:

Tool	Description
Application Design Transformer (ADT)	<ul style="list-style-type: none"> The Application Design Transformer is used to generate candidate module definitions based upon the function definition developed in the Functional Modeling phase.
Repository Object Navigator (RON)	<ul style="list-style-type: none"> The Repository Object Navigator can be used to review the default properties of the candidate modules, change the default name of the candidate modules and reject or accept the candidate modules.
Module Diagrammer (MDD)	<ul style="list-style-type: none"> The Module Diagrammer shows the modules detailed data usages and the links between the detailed table usages, as well as the layout placement of each detailed table usage.

Tool	Description
	<ul style="list-style-type: none"> The MDD can also be used to create new modules and change the modules detailed data usages.
Logic Editor	<ul style="list-style-type: none"> The Logic Editor provides capabilities for entering, editing, importing, exporting and implementing PL/SQL code. From the definitions of PL/SQL modules, the DDL script files can be generated using the <i>Generate Database from Server Model</i> utility The generated script files can be used to implement the PL/SQL in the Oracle server.

6.3.1 Objectives

The objectives of the Module Design process are:

- to design modules which support the business functions of the organization
- to provide a means of documenting the design of modules within an application
- to establish the framework for the successful generation of programs and program units defined in Designer

6.3.2 Deliverables

The Module Design document to be presented for sign-off will contain the following diagrams and reports:

- Modules in a Container
- Module Definition
- Module Network Diagram

Note: Some Applications, defined in Designer, may be implemented in non-Oracle tools. These Module reports will still be useful, as these modules will be recorded in Designer with module 'header' information (i.e. date and description of change).

6.3.2.1 Modules in a Container

This report consists of a summary of all modules in the specified container:

- all the required functions and modules are defined
- module names are descriptive and follow standards
- module purposes are descriptive

6.3.2.2 Module Definition

The Module Definition Report is used to verify:

- module names are descriptive and follow standards
- the type and language for each module is appropriate
- an appropriate description has been included for each module
- notes have been included where required
- table & view usages have been defined and are appropriate

6.3.2.3 Module Network Diagram

One or more diagrams from the Module Diagrammer tool are required to show the module hierarchy and calling network

6.3.3 Module Naming Conventions

These modules are components of the application that are not stored in the database. Usually, these components are found on the client's desktop or LAN (as in the case of forms), but may also be found on an application server (such as scripts or reports).

In order to facilitate moving these components to the most appropriate platform, these client-side modules must conform to the most limiting platform naming requirements - Windows NT.

These module names must:

- be uniquely identified by a filename (not limited to 8 characters) and 3 character extension
- the filename should be in uppercase
- the filename must be in the format AAAXXXXX where:
- AAA is the Application Short Name

XXXXX is a unique alpha-numeric identification

- if numeric identifiers are used, then all modules should be named numerically; similarly, if alphabetic identifiers are used, then all modules should be named alphabetically
- the "Main" module must be suffixed by _MAIN or 0000
- the 3 character extension must be in lower case and follow these conventions:

Extension	Meaning
.bat	script file for Windows
.fmb	Oracle Forms (uncompiled)
.fmx	Oracle Forms (compiled)
.htm	web HTML on DOS and Win31
.html	web HTML on all other platforms
.mmb	Oracle Forms Menu (uncompiled)
.mmx	Oracle Forms Menu (compiled)
.rdf	Oracle Reports (uncompiled)
.rep	Oracle Reports (compiled)
.sh	script file for Unix shell
.sql	SQL*Plus script

7 Build Phase

7.1 Overall Guidelines

This section presents some overall guidelines to assist in application development within the Oracle Designer environment.

7.1.1 Referencing Objects in Text Descriptions

Whenever the name of another TABLE, COLUMN (or any other object) is used within a textual description, it should be capitalized for easier reading (and reference).

For example, if WRQ_LAB_NO is a column, then the following notes should be used for the LGIS_FIELD_GROUPS table:

"This table has a one-way link to the WRQ system, via the shared identifier WRQ_LAB_NO"

Note: This may make maintenance of this text difficult, as changes in table or column names would necessitate updates to the descriptive text. Therefore, this is a recommended guideline, and not a Ministry standard.

7.1.2 Keeping logical data model current

In the Build Phase, there may be corrections and/or additions to the data requirements (i.e. revised column definition). Aside from de-normalization or other issues specific to physical implementation, all such changes must be re-documented in the logical data model via manual update of entities, attributes and relationships

If the application was not developed using Designer10g it may be permissible to reverse engineer the model into Designer via the *Table to Entity Retrofit Utility*.

See [Synchronizing Entities with Tables](#) for further information on the *Table to Entity Retrofit Utility*.

7.1.3 Documenting Post-Generation Changes

All post-generation module changes (aside from layout modifications) must be documented in the repository, either via a Capture Design or via text in the Module Notes. All changes to database objects must be performed via the repository. Database objects (other than modules) must not be updated directly in the target database, but instead will be updated in the Designer 10g repository and pushed out to the target DB via the CD promotion model.

Electronic Delivery of the Application

All development is done directly against the Ministry Repository, so no explicit delivery is required. However, all vendors must perform a specific number of steps. For details of this standard process, please refer to Section 8.4 (Promotion Management Procedures) of the Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc)

For a complete overview of the Ministry standard Promotion Model, see the Ministry's Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

7.2 Implementation of Database Objects

A feature of Designer is the separation between database objects (e.g. tables, views, etc) and their implementation. Each implementation is for a specific schema (or user); each user-specific implementation may have different characteristics.

A user-specific object shares the properties of the base object, but has unique storage characteristics, implementation details and privileges that are specific to a particular user. For example, storage parameters are no longer recorded against the base table definition, but instead against a user-specific implementation of the table.

The following diagram is from the Designer On-Line Help, under 'About user-specific database objects'.

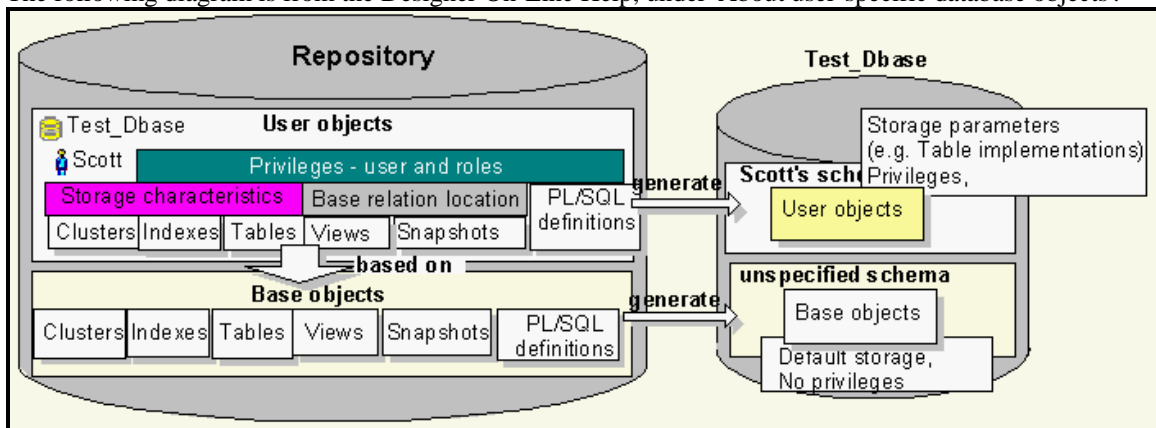


Figure 17: DB Object Implementation

Base database objects	User-specific properties
Relational Tables	<ul style="list-style-type: none"> Storage parameters Tablespace Index storage Space allocation in data blocks Privileges
Relational Views	<ul style="list-style-type: none"> Base Relation Location Privileges
Sequences	<ul style="list-style-type: none"> Sequence values Privileges
Snapshots	<ul style="list-style-type: none"> Refresh details Storage Data blocks Privileges Base Relation Location Snapshot log
PL/SQL Definitions (e.g. Functions, Procedures, and Packages)	<ul style="list-style-type: none"> Privileges
Object Tables	<ul style="list-style-type: none"> Tablespace Index storage Space allocation in data blocks Privileges

Base database objects	User-specific properties
Object Views	<ul style="list-style-type: none"> • Base Relation Location • Privileges

7.2.1 Users

It is not practical to define all the database users in the Designer tool. However, it is useful to define the Schema Owner and Proxy Users as Oracle Database Users in the repository; this permits the documentation of any special roles that the schema owner will need.

It is the Ministry standard that the Schema Owner and Proxy Users be defined as a Database Users in the repository. End Users should not be defined in the repository.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> • should be the same as the Application Short Name 	Y
Initial Password	<ul style="list-style-type: none"> • suggest using something like &NEW_PASSWORD to cause auto-prompting when running the user creation script 	Y
Complete ?	<ul style="list-style-type: none"> • Yes 	Y
Tablespaces		
Default Tablespace	<ul style="list-style-type: none"> • should be the name of the application's table tablespace (e.g. LGIS_DATA) 	Y
Temporary Tablespace	<ul style="list-style-type: none"> • should be the name of the temporary tablespace (TEMP) 	Y
Documentation		
Comments	<ul style="list-style-type: none"> • Optional 	N
Description	<ul style="list-style-type: none"> • Optional 	N
Notes	<ul style="list-style-type: none"> • Optional 	N
Tablespace Quotas	<ul style="list-style-type: none"> • This information is the same as the Tablespace Quotas Group for Tablespaces. The application schema (e.g. LGIS) should have unlimited quota on the application tablespaces. 	
Tablespace	<ul style="list-style-type: none"> • Specifies the name of the tablespace 	Y
Quota	<ul style="list-style-type: none"> • a null value means 'unlimited quota' • 'unlimited quota' should be set for the application schema 	N
Quota Units	<ul style="list-style-type: none"> • should be null for 'unlimited quota' 	N
Roles Granted	<ul style="list-style-type: none"> • This section describes the special system privileges that the user needs. <p>Note: It is preferable to grant explicit system privileges instead of the DBA role.</p>	
Role	<ul style="list-style-type: none"> • name of the system privilege or role (picklist) 	Y
With Admin Option ?	<ul style="list-style-type: none"> • usually should be null • if Yes means that the system privilege will be granted 'with grant option' 	N

7.2.2 System and Application Roles

7.2.2.1 System Level Roles

There are two types of Roles normally created for an application. These are the “System” level Roles such as APP_SCHEMA and PROXY_USER which control system privileges for the user and then there are “Application” level Roles, such as STVDES_WEB_USER which control object level access to the data structures in the schema, primarily through object privileges. The following table outlines the classification

matrix for System roles for the CD/TCA shared 10g Database environment. These must be modeled in every application, but the underlying system privileges are not the responsibility of the developer. Please refer to Section 6.2.7 for a detailed explanation on “SYSTEM” level roles and modeling these in Designer.

The following table outlines the user classification matrix for the System level roles in the CD/TCA Shared Oracle 10g database environment:

USER TYPE -----	DEFAULT ROLE -----	SYSTEM PRIVILEGES -----
End User (eg. CCONRADV)	END_USER	CREATE SESSION, ALTER SESSION
Proxy User (eg. STVDES_WU)	PROXY_USER	CREATE SESSION, ALTER SESSION + custom system privileges as required
Schema User (eg. STVDES)	APP_SCHEMA	GRANT CREATE ANY JOB GRANT ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE PUBLIC SYNONYM, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE TRIGGER, CREATE TYPE, CREATE VIEW, RESTRICTED SESSION

These are the "default" system privileges that will be granted to a user in the CD/TCA shared database environment. . For a schema level user there are also tablespace level resource quotas granted but this has to be done explicitly to the user and cannot be granted to a role.

7.2.2.2 Application Level Roles

Application Roles enable object level access to data structures in the application schema and are mandatory in every application. These must be modeled in the Designer application container and the proper database object privileges created.

Application Roles must be hierarchical; this means that 'higher' roles are granted the lower role, and then the additional grants required. For example, if there are three simple roles:

APPL_VIEWER

- can read all the tables

APPL_USER

- can read all the tables
- can insert and update all tables except for code tables

APPL_ADMIN

- can read all the tables

- can insert and update all tables except for code tables
- can delete from all tables
- can insert, update and delete code tables

This should be implemented as:

```

create role appl_viewer;
grant select on appl_table_1 to appl_viewer;
grant select on appl_table_2 to appl_viewer;
grant select on appl_code_table to appl_viewer;
create role appl_user;
grant appl_viewer to appl_user;
grant insert,update on appl_table_1 to appl_user;
grant insert,update on appl_table_2 to appl_user;
create role appl_admin;
grant appl_user to appl_admin;
grant delete on appl_table_1 to appl_admin;
grant delete on appl_table_2 to appl_admin;
grant insert,update,delete on appl_code_table to appl_admin;

```

There may be times when certain roles cannot be defined in such a manner; at such times, the requirements and reasons for this non-standard approach should be clearly documented.

Property	Rule	Req?
Name	<ul style="list-style-type: none"> • must be prefixed with the Application Short Name, e.g.: LGIS_WEB_USER 	Y
Default Password	<ul style="list-style-type: none"> • required only if the role is password protected • if used, then suggest using something like &NEW_PASSWORD to cause auto-prompting when running the user creation script 	N
Complete ?	<ul style="list-style-type: none"> • Yes 	Y
Documentation		
Comment	<ul style="list-style-type: none"> • mandatory • brief description about the role group 	Y
Description	<ul style="list-style-type: none"> • a description of the role 	Y
Notes	<ul style="list-style-type: none"> • any notes about the role • if the role is not hierarchical, then that should be noted here 	Y
Roles Granted	<ul style="list-style-type: none"> • The Roles Granted Group should not be used to assign the generic roles (Connect, Resource, or DBA) - these roles should be granted directly to users. This group should be used if roles are hierarchical in nature. 	N
Database Object Privileges	<ul style="list-style-type: none"> • These entries are usually managed from the actual database object, but access can be maintained from here as well. • the name of the specific database object should be defined in the appropriate category (table, view, snapshot or sequence) • the appropriate privilege(s) should be marked as Yes 	Y
System Privilege	<ul style="list-style-type: none"> • This group allows specific system privileges to be defined for a particular role. These are not required for Application level roles, and System level roles will be defined by the CD/TCA DBA. 	N
Privilege Name	<ul style="list-style-type: none"> • name of the system privilege being granted, should not be required for Application level roles 	N
With Admin Option?	<ul style="list-style-type: none"> • unless absolutely required, this should be No • controls the 'with grant option' clause 	N

7.2.3 Table Implementations

Standards and guidelines for tables have been discussed in the Design Phase. However, these are only for the base definitions.

The important implementation specific properties are:

Property	Rule	Req?
Complete?	<ul style="list-style-type: none"> should be set to Yes 	Y
Storage		
Tablespace	<ul style="list-style-type: none"> name of the tablespace that the table will be built in Ministry standard is <application_name>_DATA 	Y
Storage Definition	<ul style="list-style-type: none"> name of the storage definition to be used while building the table 	Y

7.2.4 Sequence Implementations

Standards and guidelines for sequences have been discussed in the Design Phase. However, these are only for the base definitions. The important implementation specific properties are:

Property	Rule	Req?
Complete?	<ul style="list-style-type: none"> should be set to Yes 	Y
Specification		
Start	<ul style="list-style-type: none"> specifies the initial value for the sequence when it is created 	Y
Increment	<ul style="list-style-type: none"> specifies the step value for the sequence when it is incremented 	Y
Cache Value	<ul style="list-style-type: none"> applicable to Oracle sequence types only specifies the number of entries that are cached set to null for the NOCACHE option 	N
Cycle?	<ul style="list-style-type: none"> if set to True, then the sequence is cyclical and sequence numbers may be re-used 	N
Order?	<ul style="list-style-type: none"> if set to True, then the ordering of values is important 	Y
Minimum	<ul style="list-style-type: none"> specifies the minimum value for the sequence 	N
Maximum	<ul style="list-style-type: none"> specifies the maximum value for the sequence 	N

7.2.5 User Object Index Storages

Indexes been discussed in the Design Phase. However, these are only for the base definitions. The important implementation specific properties are:

Property	Rule	Req?
Storage	.	
Tablespace	<ul style="list-style-type: none"> name of the tablespace that the table will be built in Ministry standard is <application_name>_INDEX 	Y
Storage Definition	<ul style="list-style-type: none"> name of the storage definition to be used while building the index 	Y

7.2.6 PL/SQL Modules

As part of the Build Phase, it is permissible to use 3rd party IDE's, such as Quest Software's TOAD, to develop and unit test the PL/SQL procedures, functions and packages. However, it is mandatory to place this code back into the Repository, and indeed, to generate these PL/SQL packages directly from the Repository.

There is a generic exception handler, pre-seeded in every application container in the Repository under:

cm_non_generated
db_objects
schema_folders
package bodies
packages

This Exception Handler is based upon Steven Feuerstein's PL/Vision freeware, although we have customized it for Ministry use (i.e. no *UTL_FILE* or *DBMS_PIPE* dependencies). The package names should begin with the short name of the application.

The SQL files (*xxx_plv.tab*, *xxx_plv.pks*, *xxx_plv.pkb*, *xxx_plv.dat*) need to be checked out by the developers, who then do a global search and replace (from the generic "XXX" to the specific application short name). The package procedures and functions will then be similar to (for example):

- *LGIS_PLV_PKG*
- *LGIS_PLVCMT_PKG*
- *LGIS_PLVTYPE_PKG*

These revised SQL files should then be saved under the application specific name (e.g. *lgis_plv.pks*) and checked back into the Repository.

Following a standard exception handler for database errors ensures that you can handle such errors consistently and robustly. The code above is package-based, and covers the raising, handling and logging of exceptions. Of course, each application will require different error handlers (e.g. error message text) and indeed different courses of action (e.g. continue through with a warning, or terminate the current transaction).

Although the source code is 'standard', each application will have its own set of *XXX_PLVxxx_PKG* packages and tables, dedicated for this application's sole use (i.e. no public synonyms).

7.3 Updating Bound Columns in Modules

As discussed in Module Design, display characteristics of bound items in modules are not automatically updated when the underlying column is updated. This problem is compounded by the fact that columns (and attributes) in a domain are not automatically updated when the domain itself is updated.

Therefore, it is a Ministry standard that developers propagate domain changes to columns in domains prior to generating the modules.

This is done by running the *Update Columns in a Domain* utility. This can be accessed from the Utilities->Designer sub-menu of the Repository Object Navigator.

7.4 Preferences

Preferences are parameters that control aspects of the Generator's behavior. Three levels of preferences are currently supported in the Generator:

- application level
- user level
- module level

The three levels of preferences are used in a hierarchical fashion by the Generator. If application level preferences are set and the user and module level preferences are not set, the application level preferences will be used by the Generator. If user level preferences are set and the module level preferences are not set, the user level preferences will be used by the Generator.

7.4.1 Objectives

Consistency in the use of Generator preference settings will result in consistent module coding styles. Applications with consistent module coding styles are easier to enhance and maintain.

Using user level preferences is not recommended, a combination of application level and module level preferences should be used. This way all users of the Generator have access to the preference settings.

Most applications will have one or more common module styles (e.g. data-entry, custom LOV, code table maintenance, etc.). One way to encourage a consistent use of preference settings is to create a 'preference module' or a "named set" for each common module style with applicable preference settings and establish an application standard to enforce the use of the 'preference modules' as module level preferences at generate time.

7.5 Code Tables

Usually, there are two different ways to create code tables within an application.

The first technique is to use a single massive code table for all of the information, with a 'code type' field used to differentiate between the sub-types. The advantage of this approach is that it is easier to maintain; the disadvantage is that referential integrity is more difficult to implement. This method is not acceptable for Ministry applications.

The second technique is to create a separate (usually smaller) code table for each code type. This approach makes referential integrity simpler (e.g.: simple foreign key constraints), but they are more difficult to maintain (a separate form must be created for each code table).

The Ministry standard is to use separate code tables for each code type, allowing Designer to generate the simple code maintenance forms. A single 'common' code table is not permitted.

7.6 Designer Generated Reference Codes - REF_CODES

Designer-generated reference codes are placed in a code table called CG_REF_CODES. This leads to name space collisions if other applications also use CG_REF_CODES.

The Ministry standard is to use code tables named APPL_REF_CODES, where APPL is the Application Short Name.

This is controlled via an option. In Design Editor, select Options -> Generator Options -> General.

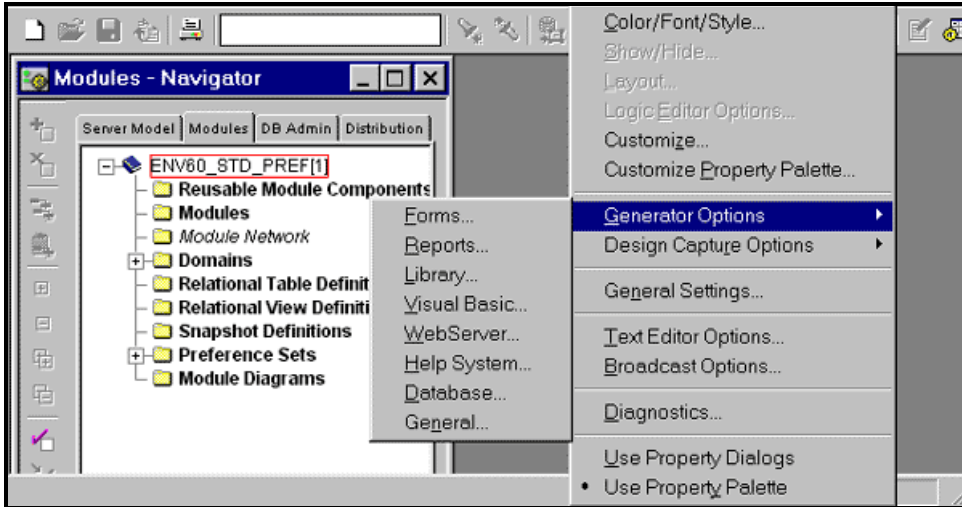


Figure 18: Generator Options

Select 'Container Wide Table' for Scope of Reference Code Table (repeat this step for each developer workstation).

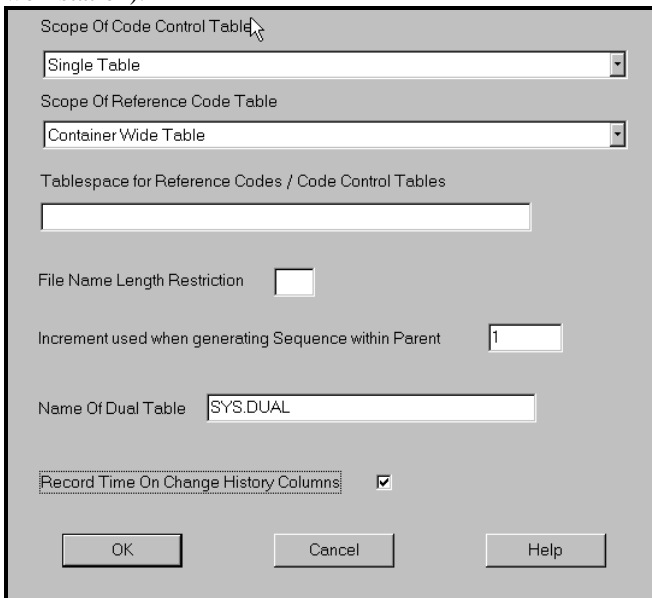


Figure 19: Reference Code Table Scope

It is the Ministry standard to reverse-engineer this <APPL_REF>_CODES table back into the repository application, for documentation purposes.

Delivery of this table may be via an export, or via the DDL creation script and data load SQL script.

8 Maintenance Phase

8.1 Overall Guidelines

This section presents some overall guidelines to assist in keeping the Repository application current during the maintenance phase.

8.1.1 Synchronizing Table Definitions

Changes to tables must be initiated from within Designer, ideally from the logical model, or from the server model table definition). It is not permissible to directly update the table definition in the database in any ministry applications. All changes must be pushed from the Designer SCM repository to the target environment

It is a Ministry standard that all table definitions be forward_engineered from the Designer 10g SCM repository, and that the associated entity be first updated prior to transforming the changes to the server model. There are exceptions to this such as journal tables, which must be reverse-engineered and retrofitted to their appropriate entity. For clarification please contact the ministry SCM Administrator.

See Capture Design of Server Model for more information on synchronizing the table definition in the repository with its definition in the production database.

8.1.2 Synchronizing View Definitions

Changes to views must be initiated from within Designer (from the view definition). It is not permissible to directly update the view definition in the database in any ministry environment. All changes must be pushed from the Designer SCM repository to the target environment.

It is a Ministry standard that all view definitions be forward engineered from the Designer 10g SCM repository, and that the associated view metadata in the server model be first updated prior to the pushing the changes to the server model

See Capture Design of Server Model for more information on synchronizing the view definition in the repository with its definition in the production database The process of capturing view definitions is the same as capturing a table definition.

8.1.3 Synchronizing Domain Definitions

If the allowable values in a domain change, then these changes should be applied to the domain definitions in the repository. Once confirmed, these changes must be propagated to the attributes and columns using the domain.

This is performed via the *Update Columns/Attributes in a Domain*, found under the Utilities menu item of the Design Editor.

Once the columns are updated, then the affected tables should be re-generated using the *Generate Database from Server Model* utility. If the existing application was generated using Designer Version 1.3.2 or previous, then you'll need to:

1. Find the name of the in-line check constraint (e.g. SYS_C00xxxx) in the USER_CONSTRAINTS or ALL_CONSTRAINTS view
2. Drop the obsolete check constraint(s)
3. Manually write the 'alter table' statements, or

Run *Generate Database from Server Model* with a connection to the production database, which will reconcile the differences and create a DDL file with the 'alter table' statements

If the application was generated using Designer 2.1.2, 6.0 or above, then the domains will have been enforced using named checked constraints (e.g. AVCON_XXXXX_). Designer's Generate Database from Server Model will do the above steps for you, as the following screen shot illustrates:

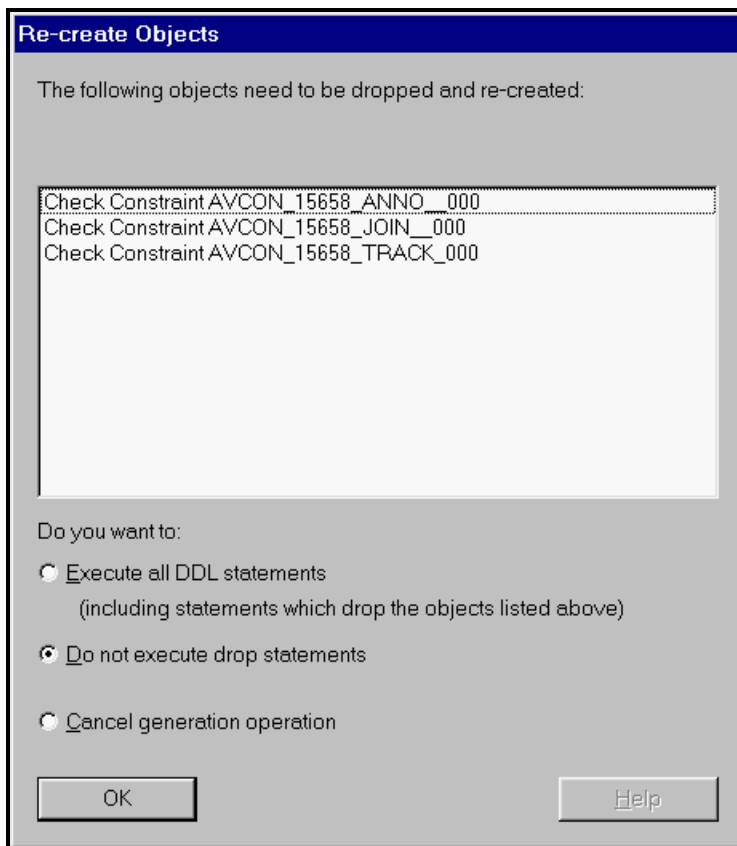


Figure 20: Recreate Domain

This *Generate Database from Server Model* must be run with a connection to the production database; otherwise, the utility will not be able to compare the repository definition against the production database definition.

It is a Ministry standard to keep the domain definitions current and up-to-date with their constraint implementations in the production database.

See Capture Design of Server Model for more information on synchronizing the domain definition in the repository with its constraint implementation in the production database. The process of capturing domain definitions is automatically done when capturing the table definition; the table being the one that has columns under that particular domain.

8.1.4 Synchronizing Display Information / Comments / Help Text

If the details of a data element change, then these changes should be applied to the logical element (e.g. attribute, entity) in the repository. Once confirmed, these changes must be propagated to the columns and tables, using the Database Design Transformer.

- In the Table Mappings tab, select the updated entity(ies) in the run-set
- In the Other Mappings tab, de-select any un-affected attributes (i.e., select only the changed attributes)
- In the Run Options tab, select only Text as the element you want to modify

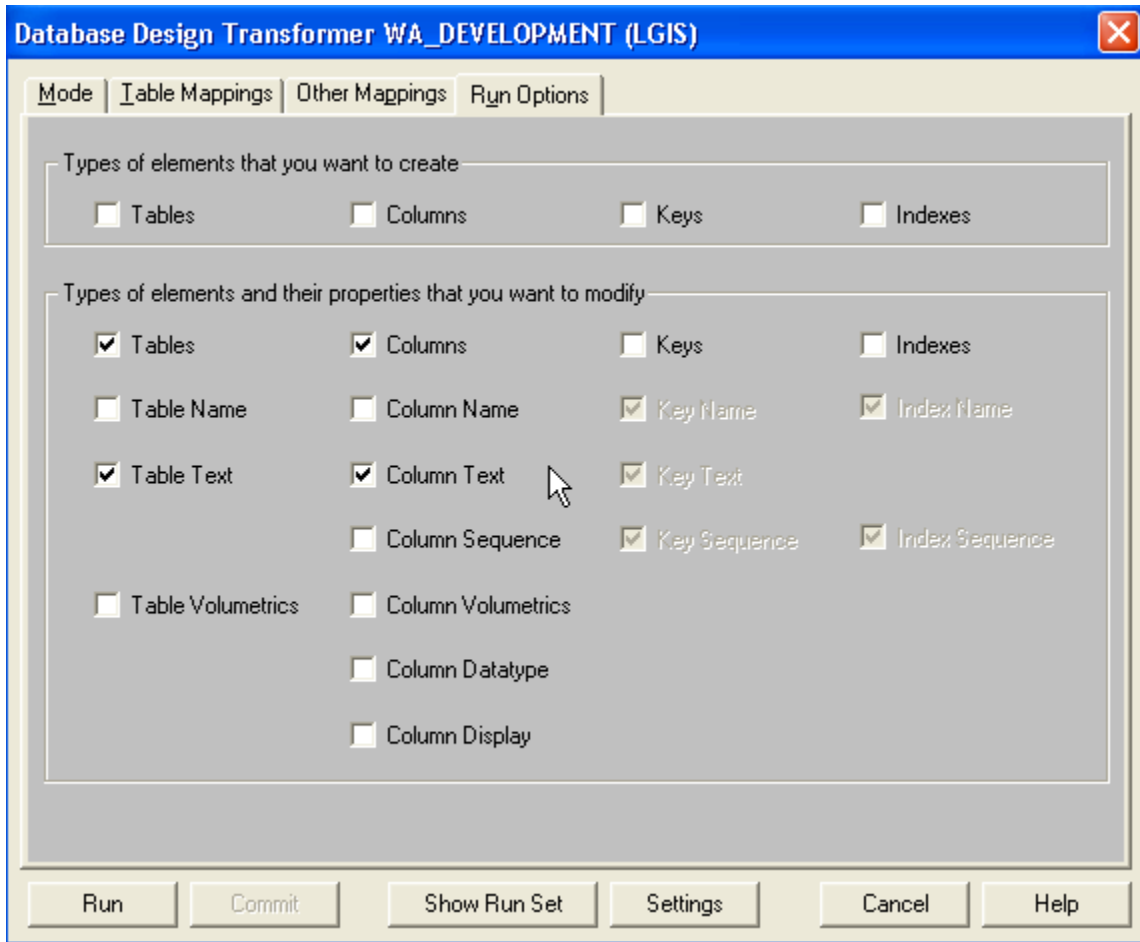


Figure 21: Database Design Transformer Options

8.1.5 Synchronizing Entities with Tables

Synchronizing Table Definitions describes how to synchronize the table definitions. If a table was added purely for physical database design reasons (e.g. sub-type implementations, special journaling tables, or derived summary information), then this is all that is required. Otherwise, it is the Ministry standard to update the logical data model.

This can be done manually, or by using the *Table to Entity Retrofit* Utility, under the Utilities->Designer menu item of RON:

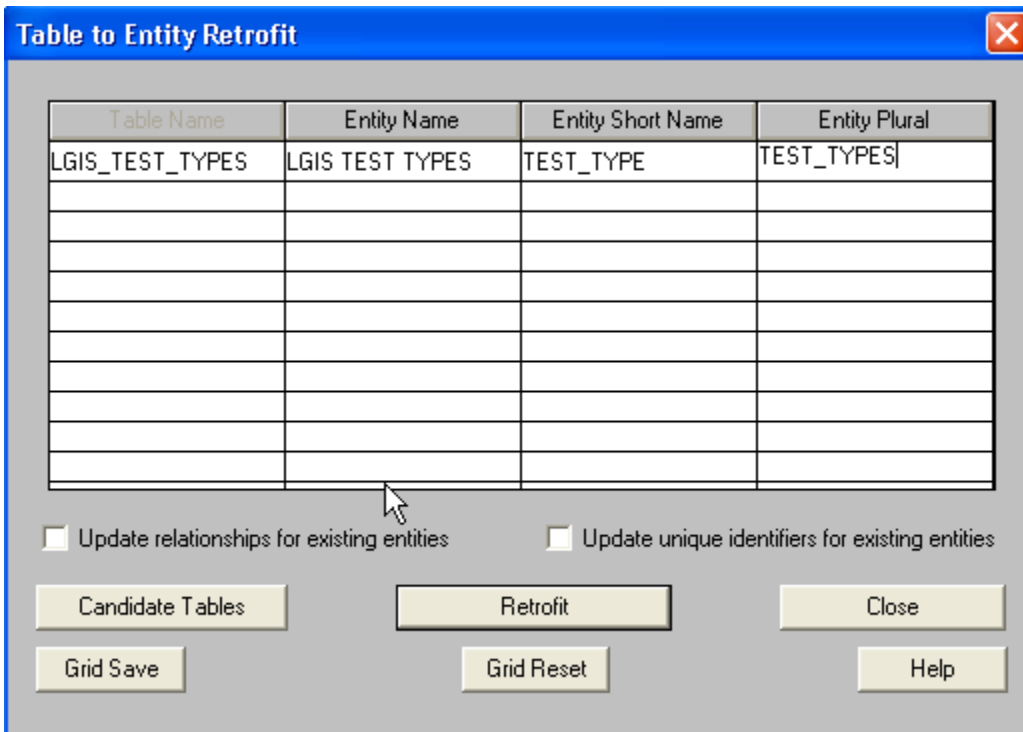


Figure 22: Entity Retrofit

Note: This Retrofit will only process new tables, and not update an existing entity.

8.1.6 Synchronizing Module Definitions

If a forms module has been updated using Forms Builder, then it may be possible to reverse-engineer this modified form using the Capture Design of Form utility (under the Utilities menu item).

Figure 23: Capture Forms

This utility will take an existing form (.fmb file) and create (or modify) the corresponding Repository module definition. If a Repository module definition already exists with the same name as the form you want to capture, you have the option either to capture the form as a new module or to merge the captured module with the existing Repository module.

During design capture, the utility creates:

- a new Repository module definition for the form being captured
- Repository window definitions for windows in the form
- Repository module components for each block in the form
- Repository base table usages for each base table block in the form
- Repository lookup table usages for each table in a list of values query in the form
- Repository bound and unbound items for each item and lookup item in each block in the form (except for control block items displayed on vertical or horizontal toolbar canvases)

- Repository item groups for items on different tab pages of a tab canvas in the form
- Repository module arguments for each parameter in the form

This utility does not capture all the post-generated changes (see Design Capture of Forms), and so this is suggested only for the simplest of forms (e.g. code maintenance forms).

For the same reason, Design Capture of Reports modules is also suggested only for the simplest of Reports.

8.1.7 Electronic Delivery of the Application

All development is done directly against the Ministry Repository, so no explicit delivery is required. However, all vendors must perform a specific number of steps. For details of this standard process, please refer to Section 8.4 (Promotion Management Procedures) of the Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

For a complete overview of the Ministry standard Promotion Model, see the Ministry's Designer Repository Management Guide (CS_TSA_Des_Mngmt_Guide.doc).

9 Designer Generation

Designer has a number of different generators available:

- Generate Server from Server Model
- Forms Generator
- Reports Generator
- Visual Basic Generator - not covered in these standards
- Web Server Generator - not covered in these standards
- MS Help Generator - not covered in these standards
- C++ Object Layer Generator - not covered in the standards

9.1 Generate Database from Server Model

The *Generate Database from Server Model* is a repository utility that produces SQL scripts that can be used to create database objects. The Server Generator is used after the developer has finished the physical database design, and produces DDL command files to build a "live" database.

The Server Generator creates a number of scripts with file extensions with the following conventions:

Database Objects	Generated Oracle Scripts
ALLOWABLE VALUES	<File_Prefix>.avt
CODE CONTROL SEQUENCES	<File_Prefix>.ccs
CLUSTER	<File_Prefix>.cls
CLUSTER INDEXES	<File_Prefix>.cli
CONSTRAINTS	<File_Prefix>.con
DATABASE	<File_Prefix>.db
DATABASE LINKS	<File_Prefix>.dbl
DATABASE DIRECTORIES	<File_Prefix>.dir
FUNCTION	<File_Prefix>.fnc
ROLE/USER GRANTS (database)	<File_Prefix>.grt
INDEX	<File_Prefix>.ind
NON-PERSISTENT QUEUES	<File_Prefix>.agn
PACKAGE	<File_Prefix>.pks
PACKAGE BODY	<File_Prefix>.pkb
Persistent Queues	<File_Prefix>.aqp
PROCEDURE	<File_Prefix>.prc
PROFILE	<File_Prefix>.prf
QUEUE SUBSCRIBERS	<File_Prefix>.aqs
QUEUE TABLES	<File_Prefix>.aqt
REPLICATION GROUPS	<File_Prefix>.rpg
REPLICATION OBJECTS	<File_Prefix>.rob
ROLES	<File_Prefix>.rle
ROLE GRANTS	<File_Prefix>.rgr
ROLLBACK SEGMENT	<File_Prefix>.rbs
SEQUENCE	<File_Prefix>.sqz
MATERIALIZED VIEW	<File_Prefix>.snp
MATERIALIZED VIEW LOG	<File_Prefix>.snl
SYNONYM	<File_Prefix>.syn
TABLE (RELATIONAL and OBJECT)	<File_Prefix>.tab
TABLESPACE	<File_Prefix>.tbs
TRIGGER	<File_Prefix>.trg
TYPES	<File_Prefix>.typ

Database Objects	Generated Oracle Scripts
TYPE METHOD	<File_Prefix>.tyb
USERS	<File_Prefix>.usr
VIEW	<File_Prefix>.vw

Generating DDL should be done in at least two sets of scripts. The first set of scripts will likely have to be run as SYSTEM or some other DBA, and will probably:

- create the necessary tablespaces (prompting for the directories for the data files)
- create the schema owner for the application
- grant any special privileges to the schema owner id

The second (and subsequent) set of scripts will be run as the schema owner and will create the database objects.

The Target tab should have the following choices set:

- *DDL Files Only* should be set (unless you are running a Reconciliation Report, in which case you select Database)
- *Type* should be Oracle10g depending on the target database
- *File prefix* should be set, so you know the names of the DDL files that are being generated
- *Directory* should be set, so you know the location of the DDL files that are being generated

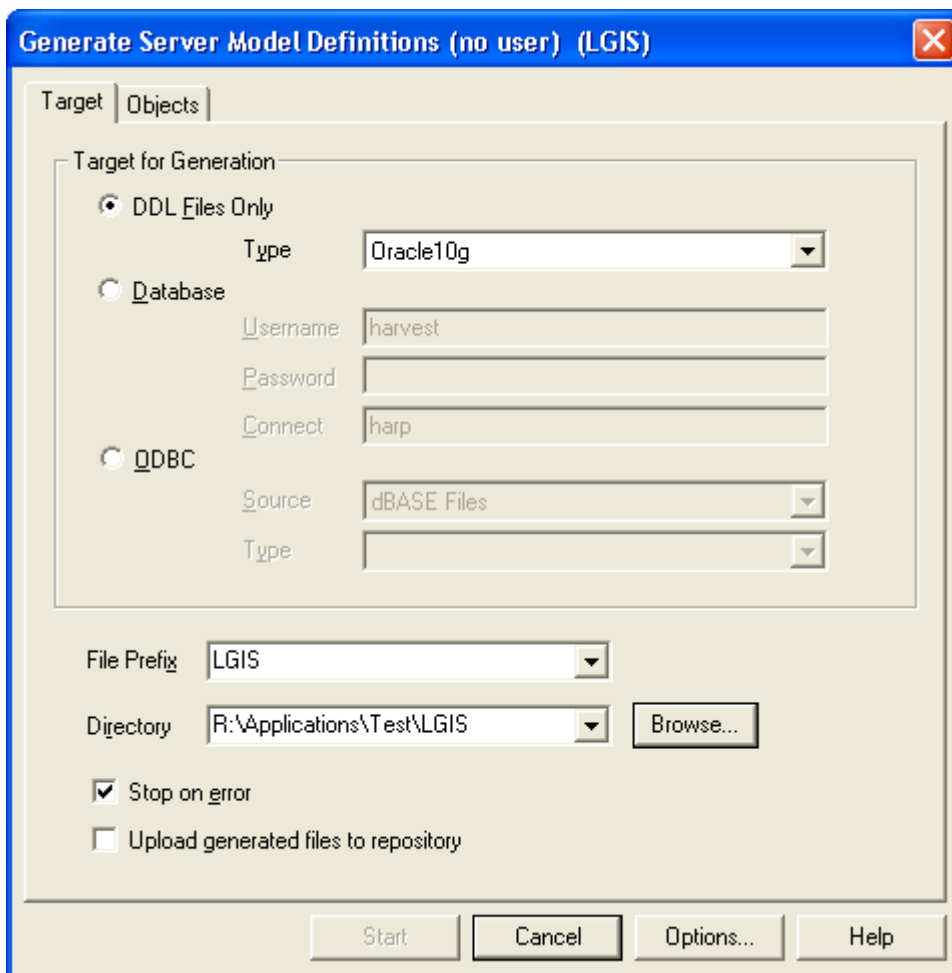


Figure 24: Generate Server Model Options

When generating DDL, the following options should be used:

Tab	Options
General	<ul style="list-style-type: none"> • <i>Generate Indexes</i> must be checked • <i>Generate Integrity Constraints</i> must be checked • <i>Generate Comments</i> must be checked • <i>Automatic Creation of REF_CODES</i> must be checked • <i>Foreign Key Generation Required</i> must be checked
Oracle Specific	<ul style="list-style-type: none"> • <i>Generate Triggers</i> must be checked • <i>Generate Valid Value Constraints</i> must be checked • <i>Generate Grants and Synonyms for Users and Roles</i> must be checked • <i>Generate Distributed Capability</i> should be checked if needed • <i>Assign Objects to Replication Code</i> should be checked if needed

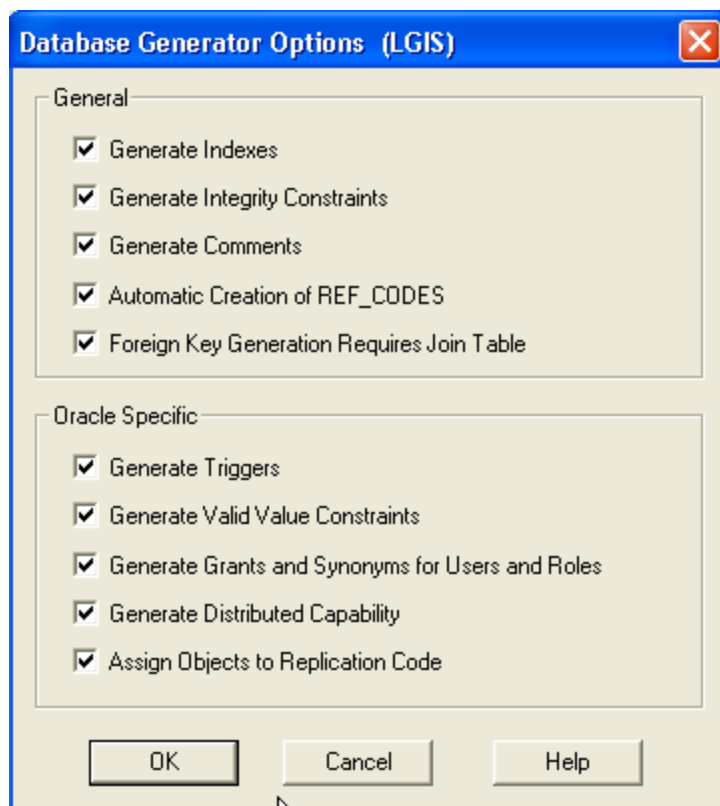


Figure 25: Database Generator Options

After the scripts are created, there may be three changes to the 'master' SQL script that may be required:

- if the scripts use '&' parameters to prompt for information then you will have to change SET SCAN OFF to SET SCAN ON
- if you specified an explicit filename prefix, you should remove the extended pathname from the commands where they are used. This is required because the paths are likely to be different when the scripts are actually run, and one can assume that all files are in 'the current directory'
- the order of the file should be reviewed to ensure that:
 - <File_Prefix>.tab is run before <File_Prefix>.con and <File_Prefix>.ind; with all three run before
 - <File_Prefix>.fnc, <File_Prefix>.prc, <File_Prefix>.pck, and <File_Prefix>.trg

In order to generate a specific implementation of the database objects (this is required to get the tablespace names and storage parameters, among other implementation specific items), you should be on the DB

Admin tab of the Design Editor, with your cursor on the database definition and user which will hold the production database objects. The 'Objects' tab will show a database and user name at the top of the left and right panes (otherwise, it says "no user").

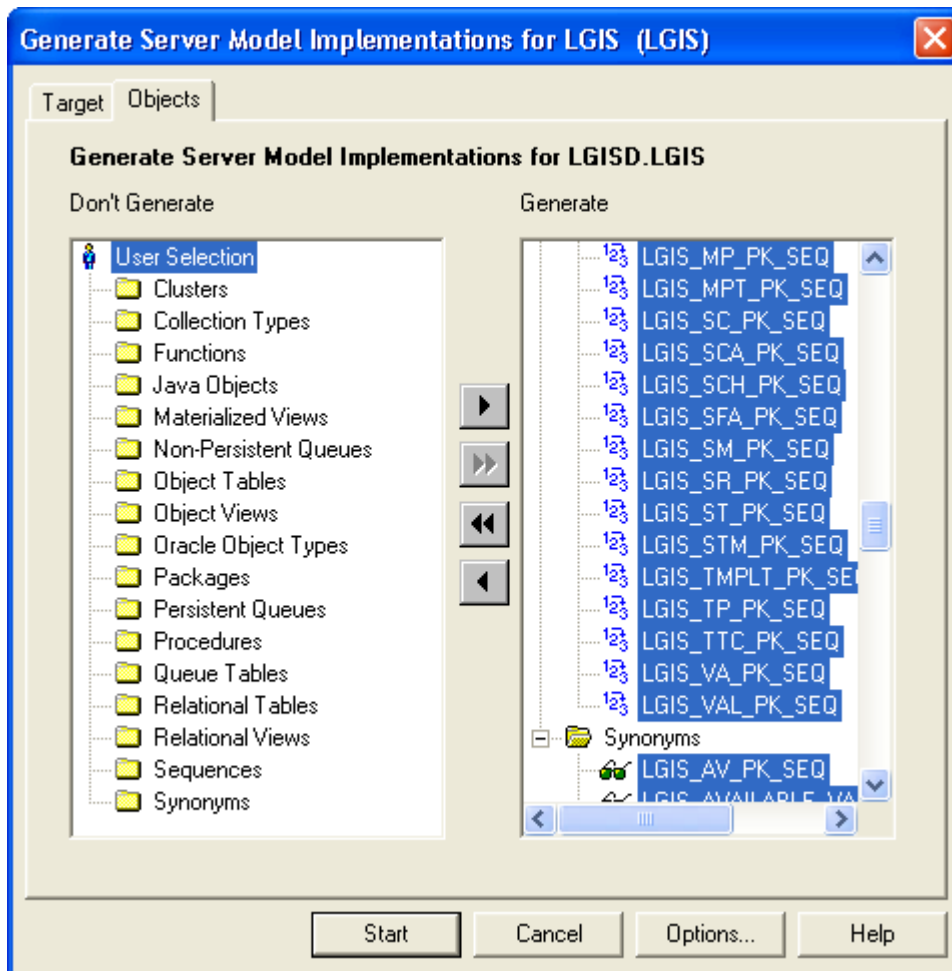


Figure 26: Generate Server Model Objects

Once finished, the utility will display the status of the implementation:

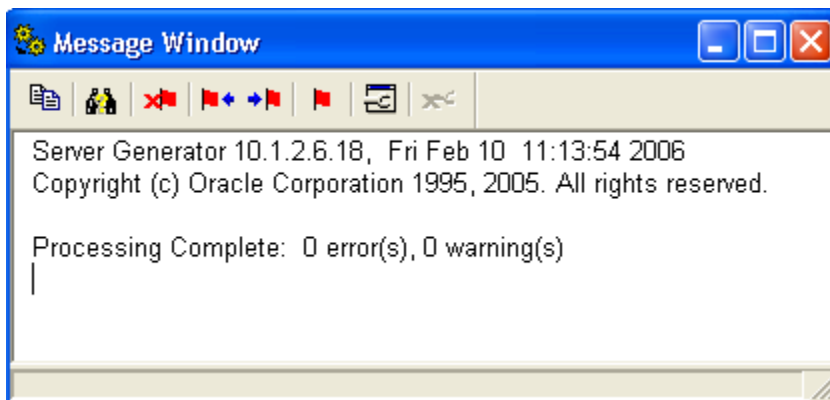


Figure 27: DDL Generation

9.1.1 Post-Generation Changes

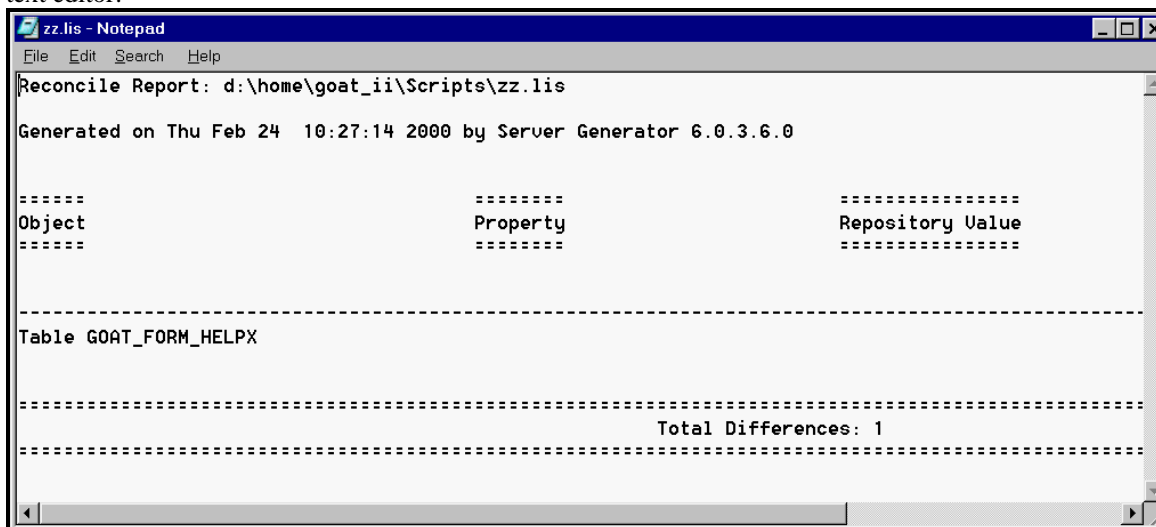
All changes to the server model should be documented in the repository first, and then generated using the 'Generate Database from Server Model' utility.

It is the ministry standard that all changes to the server model “must” be made in the repository. It is not acceptable to “hand-bomb” DDL and create new objects in the database and then reverse engineer.

9.1.2 Reconcile Report

The Reconcile Report compares the database objects in the target database against the definitions in the repository. This can be used to check through the changes that are required, or implement the changes directly on the target database.

Unlike previous versions of Designer, there is no way to directly invoke this report. You invoke this report as part of the *Generate Database from Server Model* utility, when the finished utility displays the DDL Generation Complete dialog. By selecting 'View Report', the Reconcile Report is displayed in the default text editor:



```
zz.lis - Notepad
File Edit Search Help
Reconcile Report: d:\home\goat_ii\Scripts\zz.lis
Generated on Thu Feb 24 10:27:14 2000 by Server Generator 6.0.3.6.0

=====
Object                Property              Repository Value
=====
-----
Table GOAT_FORM_HELPX

-----
Total Differences: 1
-----
```

Figure 28: Reconcile Report

The *Generate Database from Server Model* must be run with a connection to the production database (e.g. not "generate DDL to a script file"); otherwise, the utility will not be able to reconcile the repository definitions against the production database definitions.

9.1.3 Capture Design of Server Model

This feature is useful where no metadata exists about a database schema and it is deemed necessary to capture database objects and their implementation metadata back into the Designer 10g repository. Designer will reverse-engineer database objects into the repository, updating an existing object where necessary (e.g. add a new column to an existing table). The Generate menu item has *Capture Design of Server Model* to capture the production database definition into the repository.

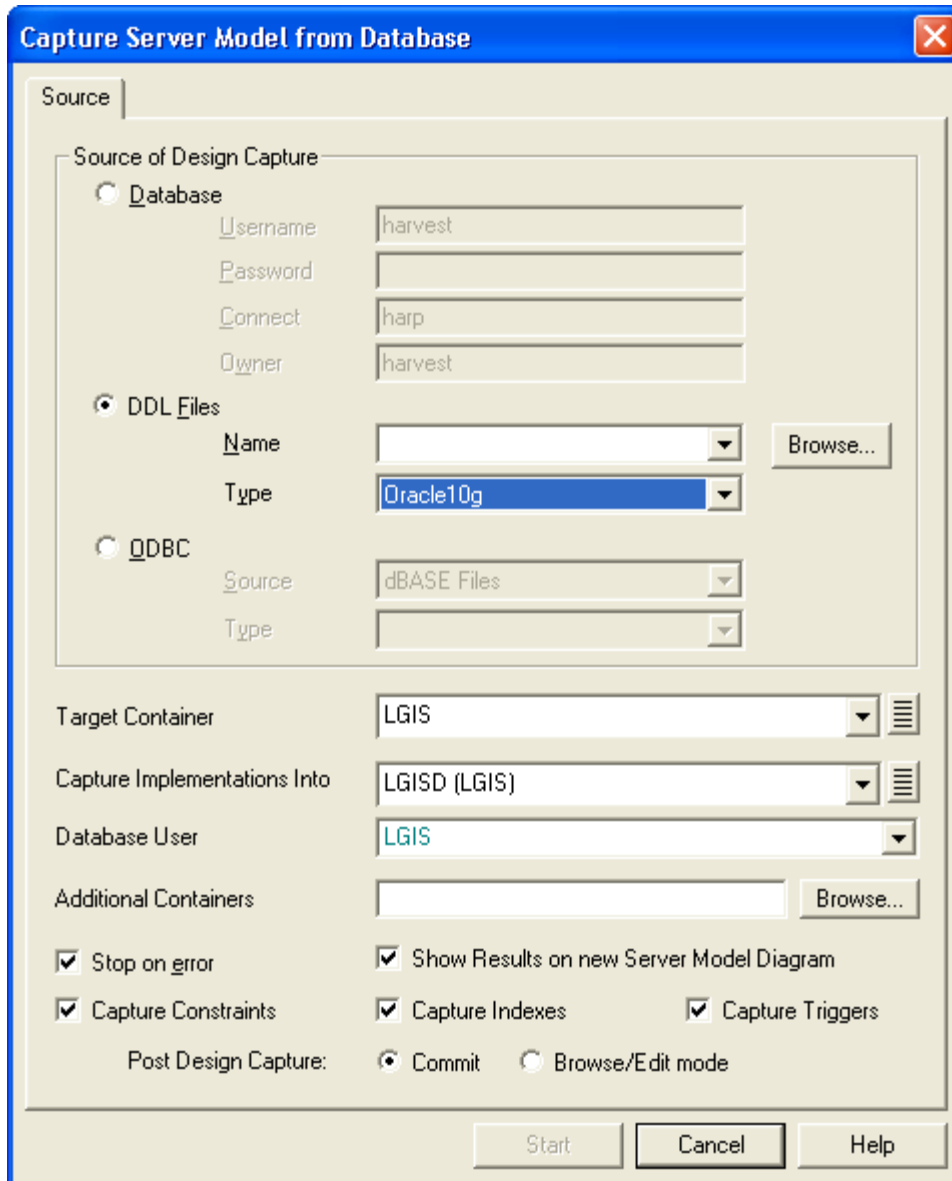


Figure 29: Capture Server Model

The checkboxes on the bottom provide options for reverse-engineering constraints, indexes and triggers. These should always be captured, along with the table definitions, and should always be checked.

Note: Table comments are not captured into an existing table definition unless the table structure differs (i.e. new column). This applies to the table's column comments as well. See Metalink Document #1077093.6 for further information.

9.1.4 Capture Design of Supporting Tables

Designer automatically generates journal tables and ref_code (e.g. for enforcement of domains), but has no method of recording storage parameters or comments against these.

Therefore, it is the Ministry standard to use reverse engineer (Design Capture of Server Model) these tables back into the repository.

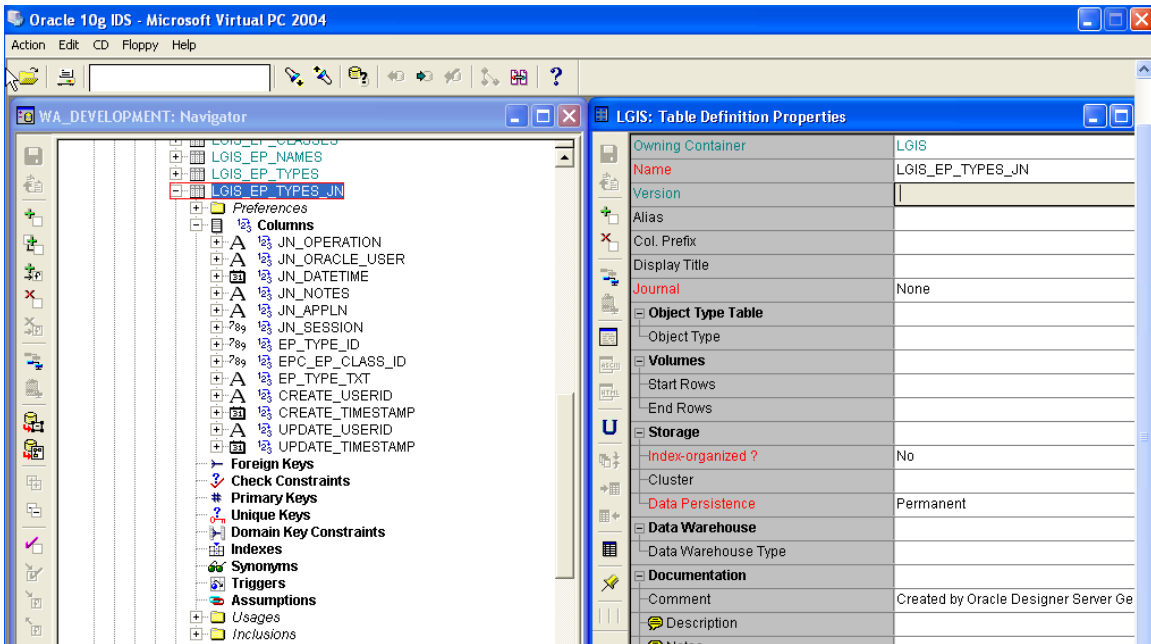


Figure 30: Journal Tables reverse engineered

10 Repository Extensions

The Designer Repository holds the meta-model of all the analysis/design elements and their properties. This meta-model is implemented as a database schema of views and columns, and can be extended by creating new properties, associations, or adding new properties to existing element types.

This is done via the User Extensibility facilities provided by the Repository Administration Utility. Although external contractors are free to extend their own repositories, the Ministry cannot maintain these extensions in its own repository.

Therefore, it is a Ministry standard to disallow repository extensions in the development of Ministry applications.

11 Summary

This document provides a set of standards and guidelines for the development of consistent and maintainable application systems utilizing Oracle's Designer tool set.

The goal of any development standard is the delivery of quality software solutions. In the case of the development of application systems, following these few simple rules can be the difference between the success and failure of a project.

12 Appendices

12.1 Appendix A – Summary of Deliverables

Phase	Deliverable	Outputs	Responsible		Participating	Min QA Reviews	Approv.	Approv. Sign-Off	QA Turn around Time (Days)
			Prim.	Sec.					
Analysis	Functional Model	Functional Hierarchy Diagram	CTR	BA	CTR, BA, UC, DA	BA	UC	Project Sponsor	10 (Functional Design)
		Function to Entity Matrix (CRUD Matrix)	CTR	BA	CTR, BA, UC, DA, DBA	DA, BA	UC	Project Sponsor	
		Function Definition Report	CTR	BA	CTR, BA, DA, UC	BA	UC	Project Sponsor	
	Entity Relationship Model	Entity Relationship Diagrams (Conceptual & Logical)	CTR	DA	CTR, DA, BA, DBA, UC	DA	DA	Project Sponsor	10 (Entity Relationship Model)
		System Glossary Report	CTR	DA	CTR, DA, BA, UC,	DA	DA	Project Sponsor	
		Entity Definition Report	CTR	DA	CTR, DA, BA, UC	DA	DA	Project Sponsor	
		Entities and Their Attributes Report	CTR,	DA	CTR, DA, BA, UC	DA	DA	Project Sponsor	
		Entity Completeness Checks Report	CTR,	DA	CTR, DA, BA, UC	DA	DA	Project Sponsor	
		Domain Definition Report	CTR ,	DA	CTR, DA, BA, UC,	DA	DA	Project Sponsor	
		Attributes In a Domain Report	CTR,	DA	CTR, DA, BA, UC	DA	DA	Project Sponsor	
	Business Area Model	Business Unit Definition	CTR	BA	CTR, BA, UC	BA	UC	Project Sponsor	2 (Business Area Model)
		Business Unit to Function Diagram	CTR	BA	CTR, BA, UC	BA	UC	Project Sponsor	
	Process Model	Process Model Diagram	CTR	BA	CTR, BA, UC, DA	BA	BA	Project Sponsor	10 (Process Model)
Design	Data Base Design	Proposed Database Design	CTR	DBA	CTR, DBA, DA	DBA, DA	DBA	Project Sponsor	5 (Data Base Design)
		Entity to Table Implementation	CTR	DBA	CTR, DBA, DA	DBA, DA	DBA	Project Sponsor	
		Table Definition Report	CTR	DBA	CTR, DBA, DA	DBA,	DBA	Project Sponsor	
		Server Model Diagram	CTR	DBA	CTR, DBA	DBA, DA	DBA	Project Sponsor	
		Database Table and Index Size Estimates	CTR	DBA	CTR, DBA	DBA, DA	DBA	Project Sponsor	
		Role Definition (Design/Implement Security)	CTR	DBA	CTR, DBA	DBA, DA	DBA	Project Sponsor	
	Module Design	Modules in a Container	CTR	AA	CTR, AA, DBA	AA	AA	Project Sponsor	5 (Module Design)
		Module Definition	CTR	AA	CTR, AA, DBA	AA	AA	Project Sponsor	

Phase	Deliverable	Outputs	Responsible		Participating	Min QA Reviews	Approv.	Approv. Sign-Off	QA Turn around Time (Days)
			Prim.	Sec.					
		Module Network Diagram	CTR	AA	CTR, AA, DBA	AA	AA	Project Sponsor	
Build	Data Base	DDL Scripts	CTR	DBA	CTR, DBA, AA	DBA	DBA	Project Sponsor	5 (D/B Design)
		Complex SQL Query Analysis	CTR	DBA	CTR, DBA, AA	DBA	DBA	Project Sponsor	
	Application	Client-Side Modules	CTR	AA	CTR, AA, DBA	AA	AA	Project Sponsor	5 (Application Design)
		Ancillary Objects (e.g. Help Text, Scripts)	CTR	AA	CTR, AA, BA	AA	AA	Project Sponsor	
Maint.		Updated Repository Configuration	DA, DBA		DBA, DA			Project Sponsor	
		Microsoft VSS (for non-Oracle tools)	??		??				

Legend:

BA – Ministry Business Analyst; DA – Ministry Data Administrator; DBA – Ministry Database Administrator; AA – Ministry Application Analyst; UC – User Client; CTR – Contractor

12.2 Appendix B – Glossary of Terms

Term	Definition
Application Configuration Manager	<ul style="list-style-type: none"> This is the vendor's delegate for the Ministry's Repository Administrator, responsible for the delivery of their specific application.
DWS	<ul style="list-style-type: none"> Development and Web Services
Role	<ul style="list-style-type: none"> This refers to the authority or role that a user is granted within the Designer Repository. The Repository Admin Utility is used to grant the roles. The two possible roles are USER and MANAGER.
DBA	<ul style="list-style-type: none"> Data Base Administrator
DDL	<ul style="list-style-type: none"> Data Definition Language
LOV	<ul style="list-style-type: none"> List-of-values picklist
Meta-Model	<ul style="list-style-type: none"> A meta-model describes the types of elements and associations which are used when constructing particular kinds of models.
Repository Administrator Group	<ul style="list-style-type: none"> This is the group of people who are authorized to administer the Designer Repository. Often, these are a subset of the Database Administrators (DBA's) who have extensive DESIGNER experience. These people have been granted the MANAGER Case Role. These people may also know the Repository Owner password.
Repository Owner	<ul style="list-style-type: none"> This is the Oracle userid under which the repository was built, and should be the owner of all the applications within the repository. Access to this account should be limited to qualified personnel only

12.3 Appendix C – Standard Approved Abbreviations

12.3.1 Mandatory Abbreviations

Verb or Noun	Abbreviation
AVERAGE	AVG
DESCRIPTION	DESC
CODE	CD
HECTARES	HA
IDENTIFICATION	ID
INDICATOR	IND
MAXIMUM	MAX
MINIMUM	MIN
NUMBER	NO
PERCENT	PCT
SURROGATE KEY	SKEY
TIME	TM
TRANSACTION	TXN
XREF	XF
YEAR-TO-DATE	YTD

12.3.2 Preferred Abbreviations

Verb or Noun	Abbreviation	Verb or Noun	Abbreviation
ADDRESS	ADDR	ADMINISTRATION	ADMIN
ALTERNATE	ALT	AMOUNT	AMT
AMERICAN	USA	A PPLICATION	APPL
AUTHORITY	AUTH	BUSINESS	BUS
CANADIAN	CDN	CATEGORY	CAT
CLASSIFICATION	CLASS	CLIENT	CLI
COLLECTION	CLCTN	COLUMN	COL
COMMENT	CMT	COMMISSION	COMM
COMMITTEE	CTTE	COMPANY	CO
CONDITION	CONDTN	CONTROL	CTL
CONVERSION	CNV	COORDINATE	COORD
CORPORATION	CORP	CORRECTION	CRCTN
COUNT	CNT	CREDIT	CR
DATE (Gregorian Date)	DT	DAY	DY
DESTINATION	DEST	DEPARTMENT	DEPT
DETAIL	DTL	DEVELOPMENT	DEV

Verb or Noun	Abbreviation	Verb or Noun	Abbreviation
DIAMETER	DIAM	DISTRICT	DIST
DIVISION	DIV	DOCUMENT	DOC
EFFECTIVE	EFF	ELEMENT	ELMNT
ERROR	ERR	ESTIMATE	EST
EXECUTIVE	EXEC	EXPIRY	EXP
FACTOR	FCTR	FEDERAL	FED
GROUP	GRP	HEIGHT	HGHT
HOUR	HR	INDEX	INDX
INITIAL	INIT	INVENTORY	INV
JURISDICTION	JURIS	LATITUDE	LAT
LENGTH	LEN	LETTER	LTR
LICENCE	LIC	LOAD	LD
LOCATION	LOCN	LONGITUDE	LONG
MANAGEMENT	MGT	METHOD	MTHD
MINUTE	MN	MONTH	MO
NAME	NM	ORGANIZATION	ORG
PAYMENT	PAY	PERMIT	PRMT
PIECE	PCE	POSITION	POS
PREVIOUS	PREV	PRIMARY	PRI
PRODUCT	PROD	PROJECT	PROJ
QUANTITY	QTY	RECEIVED	RECV
REFERRED	REF	REGION	REG
REGISTRATION	REGN	RESPONSE CENTRE	RCC
REQUEST	RQST	REQUIRED	REQ
REQUIREMENT	RQMT	RETURN	RET
REVENUE	REV	SCHEDULE	SCHED
SCREEN	SCR	SEARCH	SRCH
SECONDARY	SEC	SECTION	SECT
SEQUENCE	SEQ	SERVICE	SRVC
SOURCE	SRCE	SPECIES	SPP
STATEMENT	STMT	STATUS	STS
STATUTORY	STAT	STATISTICS	STATS
TENURE	TENR	TEXT	TXT
TIMESTAMP	TS	TITLE	TTL
TOTAL	TOT	TREATMENT	TRTMT
TYPE	TYP	USERID	UID
VALUE	VAL	VERSION	VER
VISITATION	VISIT	VOLUME	VOL
WITHDRAWAL	WD	WEIGHT	WGT
YEAR	YR		

12.4 Appendix D – Developer Guidelines

Due to “Merging” issues in the versioned repository, developers actively developing in the Repository should note the following guidelines:

1. It is not necessary to check-out/check-in the application container (eg. LGIS) on a daily basis. It can stay checked out until a configuration savepoint is reached. This will prevent unnecessary internal versioning of the container in the repository.
2. Individual elements (e.g. LGIS_FIELD_GROUPS table) also don't have to be checked in on a daily basis, unless a savepoint is desired. Even if an element is checked out and the user logs off the repository will maintain the current state of the element and any changes made. This will also prevent excessive versioning of elements.

Prior to creating or updating a UDS, if a file needs to be uploaded for delivery, the proper process is to check out the current version in WA_DEVELOPMENT and upload (with overwrite) the new version.

NOTE: Do not delete the file(s) in WA_DEVELOPMENT before uploading the new version. This breaks the version tree for that particular object.

12.5 Appendix E – Developer SCM Guidelines

In conjunction with Oracle Repository SCM usage, developers performing post-generation changes on Forms and/or Reports must use the proper Folder Structure in their application Container.

The Folder structure consists of the following:

admin	Scripts used in delivery
bin	Compiled executables (e.g. .fmx, .mmx, .plx)
data_conversion	Data Conversion scripts and export files
docs	user and system documentation
form_letter	Document management templates
forms	Source forms (.fmb, .pll, .mmb)
icons	Forms .ico files
logs	Output log files from processing scripts
misc	DOT, ICO, and image files
reports	Source reports (.rdf)
scripts	Scripts used on a regular basis

For example, the STVCMS container would have the following Folder structure:

The screenshot displays the Oracle Repository Object Navigator interface. The left pane shows a tree view of the repository structure. The 'STVCMS' container is expanded, showing a 'Forms SCM' folder with sub-folders: admin, bin, data_conversion, docs, form_letters, forms, icons, logs, misc, reports, and scripts. Below these are 'Files' including various SQL scripts like STVS_DBA.rls, STVS_DBA.sql, STVS_DBA.usr, STVS_DBA_CREATE_NEW_USER.sql, and numerous DDL files for tables, views, and packages.

The right pane shows the 'LGIS: Application System Properties' dialog box. The 'Name' field is set to 'LGIS'. The 'Version' field is '1.6CO(1)'. The 'Title' field is 'LGIS Release 5'. The 'Authority' field is empty. The 'Owner' field is 'REPOS_MANAGER'. The 'Data Warehouse?' field is 'No'. The 'Documentation' section is expanded, showing fields for Priorities, Constraints, Comment, Summary, Objectives, Description, Notes, and OS Timestamp.

Once post-generation changes have begun, developers will need to download the source files down to their local drive.

Upon release, there will need to be one final upload, to synchronize the Repository files with the files on the file server. Note that you should first **check-out** the Repository file.

Once uploaded, these objects should be checked-in and then included in the UDS (see Section 4.2) for migration.

For more information, please see the Designer on-line help, under:

- uploading files, uploading files or directories to the repository
- file systems, downloading files or containers to