

Guide S7 - Modelling Standards for Relational Applications

Purpose of Document

This guide illustrates and explains the form, standards, and conventions to be used for data models of applications intended for any of the information processing environments at the Ministry of Forests and Range. Although data models most easily translate to relational-type databases, they must be used in defining requirements for **any application, no matter what the physical platform**.

The guide is intended for systems analysts and systems designers who have some knowledge of the techniques and procedures involved in information engineering methodology. The data modelling standards that we employ are very rigorous, and we firmly believe they help tremendously in identifying and fixing problems earlier in the development cycle. The difficulty with rigorous standards is that those involved in the development process must first take the time to understand the standards, then must use them properly. If you have any questions about this guide or the data modelling process at the Ministry of Forests and Range, see one of the [Data Administration staff](#) in the Information Management Branch (387-1471).

Logical modelling is platform-independent so there are no differences in required standards between the major ministry computer environments (ORACLE, LAN [*local area network*], and PC [*personal computer*], e.g. Microsoft Access). For physical modelling, differences exist in the physical platform implementation, but standards for complete physical definition remain the same. For the ORACLE platform DA and DBA tools exist to implement the physical tables from the data definition. *It is important to build the logical model and physical data definition for ALL environments during the requirements definition and design phases.*

Items Modellers Should Read

Project leaders and staff from the involved business area(s) should understand the **overall process for model validation** ([Deliverables](#)).

Modellers must understand these parts of the guide thoroughly to engage in logical modelling:

- [Principles for Defining MoFR Data](#)
Explains the integrated nature of MoFR data and the ministry's principles for data sharing. Changes to shared data must be analysed for impacts.
- [Logical Model Review Process](#)
Documents the essential questions that will be asked at a review, so developers can prepare the necessary answers prior to the review.
- [Model Display Standards](#)
Lists technical details about how to best display data models, so developers can ensure their documentation is formatted to display the most information possible.

Modellers should provide details to project leaders for impact analysis. Project leaders fill out the [Impact Analysis Checklist](#) from the SDLC [Impact Analysis](#) deliverable to ensure Data Impacts, Corporate Code impacts, Database Impacts, Data Warehouse impacts have been addressed.

Related Guides

See [SDLC](#) for a description of the systems development process at the Ministry of Forests and Range. See [Guide S19](#) for standards on the naming of entity types and attributes in the logical model and the naming of tables and columns in the physical model. See [Guide S21](#) for use of corporate code tables. See [Guide S35](#) for information about responsibility (custodianship) for defining data structures.

Data Modelling - The Big Picture

Data Definition at MoFR

Defining data at MoFR requires business understanding of the item being defined, creation of a single business view of the data and then creation of implementation view(s) of the data for defining physical storage. The [MoFR Data Definition picture](#) shows a high level view of MoFR data definition.

Principles for Defining MoFR Data

The basic principle for defining data is that data is **defined once, captured once and used many times**.

Defined once means that the data is defined from a corporate ministry-wide perspective. One business area is given the responsibility for defining the data and setting data standards to use throughout the ministry wherever that data is needed.

Captured once means the data is captured through one process and therefore has only one source. For example, the source of all MoFR client data is the Client Management System. This is the only system which is used to capture and maintain MoFR client data.

Used many times means that the data is shared throughout the ministry wherever it is needed. This sharing of the data is often accomplished in many ways. There may be distributed access to the source data throughout the ministry as needed. There may also be access to replicated copies of the data where needed. In our client example, Client Management System data is highly shared data used throughout the ministry. It is used by many applications. It is also used by many applications on other ministry platforms. It is made available to other platforms through planned replication.

We need to make a **strong distinction between replication and duplication of data**. Replication of data is a planned activity where the replicated data is updated from the source data on a scheduled basis. Duplication of data is maintaining the same data in more than one place. Replicated data can be relied on to be consistent with the source data except for the time between replications. Duplicated data can never be counted on to have any consistency as it is not maintained through one source. Replicated data can have timing (data currency) issues but as long as the currency of the data is understood these can be managed. Inconsistencies between duplicate data cannot be managed and cause business errors. Therefore the ministry does not define, create or store duplicate data.

One more important principle is the **integration of data, especially spatial and attribute data**. Much of the ministry's attribute data is already integrated through the type of single data definition and data sharing shown above in the Client example. The data is maintained in one place and is seamlessly available where needed. The ministry's spatial data is also being integrated. Integration between spatial and attribute data is a new challenge.

History Problems with Data Definition

Historically, some ministry data has been created at the physical level and consequently does not have a well-defined, clear business definition. Therefore it is often difficult to understand what the data is meant to be. This data is not as easily shared throughout different ministry business areas as it is not easily understood. It is also prone to duplication.

Also historically, some of our spatial and attribute data has been defined in isolation. Spatial and attribute data have also been viewed as distinct and different types of data. Therefore the spatial and attribute data don't always fit together well. Sometimes spatial data has been defined where attribute data would be more appropriate. Sometimes geographical (spatial) information (e.g. map feature location coordinates) has been defined in attribute data.

Dealing with or Living with History

Where possible we will ensure that the data items defined make good business sense, support all ministry business needs for the data, and are well designed from a good data modelling and data management perspective. Where historical data items exist which are not well designed, changing them is a judgement call based on the cost of the change as opposed to the cost of living with the poor design.

With data management tools such as Oracle Locator and SDE, spatial and attribute data are no longer stored separately. Spatial mapping data and textual attribute data are just types of business data defined together in the same data structures. Spatial and attribute data should no longer be thought of as distinct and different types of data. Data designed and grouped to reflect business area needs will be viewed as business item data definition which may have both spatial and attribute components. For example, a Seed Planning Zone will have attributes defining the type of zone and associated seed information. It will also have geometry defining the geographic area of the zone.

[Back to top](#)

The Data Modelling Process

Overview

Data modelling is an integral part of systems analysis and system design. This document describes the data models that are required for each development project in the Ministry of Forests: the *logical data model* and the *physical data model* and, where spatial data is used, rules for *spatial data definition*.

The logical data model consists of an *Entity Relationship diagram* (see [Data Model Examples #1](#) and a *data dictionary* (see [Data Model Examples #2](#) and [#3](#)). The ERD is used to show entity types and display relationships between entity types in a pictorial format. The data dictionary contains textual information about each entity type. Data Administration staff review the logical model with the designer and ministry staff to ensure that integration requirements were considered and that proper data modelling techniques were followed. Developers should consult with [Data Administration staff](#) as early as possible, to obtain relevant data definitions for use within their development projects.

Once the logical model has gone through the validation process, the designer generates a preliminary physical data definition using the logical data model as input (see [Data Model Examples #4](#) (Physical Data Model) and *data dictionary* (see [Data Model Examples #5](#)). The final version of the physical data definition will be a result of a joint effort between Data Base Administration (DBA) staff and the application developer. Data Administration staff also participate to ensure information requirements identified in the logical model are translated properly to the physical model. DBA staff mainly ensure that the trade-offs between performance, data storage requirements, and application complexity are balanced appropriately.

Search Existing Data First

The Ministry of Forests and Range has extensive attribute and spatial data holdings from significant investment over a number of years. As duplicate data is very expensive to maintain and very prone to causing business errors, the ministry does not define, create or store duplicate data unless it is planned redundancy (data replication) for a specific business purpose like a reporting database.

Therefore, **before defining new data we must determine if the desired data already exists** within MoFR. While Data Administration staff will review data models for duplication, it is wise to check for existing corporate data definitions before modelling begins. Most of the ministry's corporate data definitions are held in a searchable data dictionary available on the web. Search for data definitions in the [Integrated Data Dictionary](#). Information on Land and Resource data is also available through the [Land Information BC Discovery Service](#)

Deliverables

There are two major deliverables from the logical modelling process:

1. **First cut model** (for discussion). The contractor generally works independently with business area staff to develop the first cut model. Data Administration staff are available to assist with interpretations, model design, discussion, etc. Business area staff can request higher involvement from Data Administration staff, depending on:
 - project leader (or Data Administration staff) confidence in the contractor's data modelling skills;
 - the contractor's demonstrated knowledge of our data modelling standards.
2. Verification process leading to a **validated logical data model**. There are two major factors in determining the amount of time needed for completion:
 - a) Scope of project.
 - b) Business issues raised requiring resolution effort by the ministry.

Estimating resourcing: a rule of thumb is approximately one hour per entity for the validation process to approve a data model. This includes a minimum of two passes through the model with business staff present (first pass, detailed review; second pass, ensuring corrections are complete). **Requirements cannot usually be correctly defined in only one pass.** Where the contractor is an experienced and competent data modeller and has successfully completed projects within the ministry before, this estimate might be cut in half or even more. Where contractors are inexperienced or it is their first project with the ministry, where the business is complex, or where spatial modelling is included, this estimate might be as much as doubled or more. Reviewing a data model from a technical "modelling" perspective is relatively quick; what takes time is ensuring the business staff understand the information requirements that they are committing to, and in the worst case resolving business issues that are raised during the modelling process. Where already-existing entities can be used without modification, the modelling costs are cut considerably, and the business result is higher quality based on validated business requirements, increased data sharing, and reduced data redundancy. **NOTE: no matter how much the modelling process costs, it is still cheaper to resolve the business issues at that time, rather than ignoring them and trying to fix or enhance the application later!**

For more detail on the Review Process see [Logical Data Model Review Process](#) and [Physical Data Model Review Process](#) for more detail.

Data Models vs. Business Rules

Data models do not inherently describe the business rules of an organization. Data models do describe *some* of the business rules, such as "each order may be placed by only one customer" and "each customer may place many orders". However, business rules such as "a customer who has more than \$100,000 on account may not place an order" can only be implemented by programming, and should be described and documented as processes which interact with the data. The data model has no constructs available to document the processing rules that are normally required between several entities.

This makes it imperative for the system and user documentation to use additional documents to clearly spell out the business rules that act on the data.

Spatial Data Definition

Spatial data and attribute data are different types of information collected for a business item. Most business item data definitions will have attribute information. Some will also have a spatial component to be able to display the business item on a map. For example Forest Roads or Openings

have a spatial component. From the logical data model / business perspective defining spatial data is very similar to defining attribute data in that it must be defined from a corporate business perspective. The same standards for data sharing and non-redundancy which apply to attribute data also apply to spatial data. Each spatial [feature class](#) (like an entity) is the responsibility of a single business area and is assigned a data custodian. The data custodian is responsible for defining the corporate data standards by which the ministry will capture and use the spatial feature data. Whether the implementation is as Oracle Locator Layers or on another platform, features are defined by the data custodian.

Spatial data is also shared among different MoFR business areas, and with other ministries and forest industry business partners. The ministry is currently updating the [Integrated Data Dictionary](#) which holds corporate data definitions. Information on Land and Resource data is also available through the [Land Information BC Discovery Service](#). [Search for existing data first](#) before defining new feature classes will also find classes from external sources (e.g. Terrain Resource Information Management program).

[Back to top](#)

Logical vs Physical Data Models -- Overview

A clear distinction should be made between a Business Area Analysis model, a logical data model, and a physical data model.

Business Area Analysis Model

A Business Area Analysis (BAA) usually differs significantly from a project level logical model. BAAs are not concerned with solutions to immediate problems; they instead provide a foundation for developing integrated information systems, and often can also be used to validate or change lines of business. During a BAA, analysts define and refine representations of the activities a business performs (functions and processes), fundamental things of relevance to the business (entity types), and the interaction between the two. The BAA therefore delivers two different models: a functional model and a data model.

Because of its increased scope, the BAA data model is usually at a somewhat high level of detail and may leave some areas incompletely documented. The BAA is meant to give a relatively detailed understanding of the business, but does not necessarily flesh out the full details required to implement an automated system.

Logical Data Model Overview

Detailed logical data models are usually developed at the Ministry of Forests and Range as part of an application development project. The purpose of the logical model is to show the data that the application must store in order to satisfy business requirements. It shows how this data is related and explores any integration requirements with business areas outside the scope of the development project. It is created without any specific computer environment in mind, so little optimization for performance, data storage, etc is done.

Logical data models:

- identify all entity types within the scope of the project and textually describe each;
- identify the ministry Data Custodian of each entity type;
- include the identifiers (primary keys) for each entity type;
- include a complete list of attributes (data elements) for each entity type and textually describe and define each attribute;
- show properly named relationships between entity types, and note identifying relationships between entity types where appropriate;
- are usually expressed in third normal form, although they need not be.

Physical Data Model Overview

The purpose of the physical model is to show how each data element will be implemented and stored on the database. Physical models may vary from the logical model in that the physical model may introduce objects that do not contribute to the business information requirements of the application. These new objects may be created in order to speed response times, ensure that the application fits within the physical limitations of the computing environment, improve maintenance turnaround, or for other reasons.

Physical data models:

- are primarily concerned with physical limitations, performance and space requirements;
- need not be expressed in third normal form;
- identify all tables for the application;
- identify all columns listed in [standard ministry column order](#);
- include definitions for all columns in each table;
- identify the primary index for each table;
- identify alternate indices for each table;
- show relating columns for each relationship (foreign keys).

Spatial Data

The logical side of defining spatial data is the feature class *business definition*. The definition includes:

- a meaningful, intuitive feature class name;
- a clear description of the spatial data item;
- identification of the type of spatial data (point, line, polygon);
- identification of responsibility for the data standards (Data Custodian);
- identification of working level responsibility for the data standards (Data Standards Manager);
- identification of responsibility for the physical data management (Data Steward);
- identification of theme(s) which the feature class is used in

The physical side of defining spatial data is the platform implementation definition. For MoFR operational spatial data this definition will mostly be in Oracle Locator.

After definition, new feature classes are added to the Integrated Spatial Data Dictionary by Data Administration staff.

High Volume Data Design

For physical data management, high volume data must be stored in its own table. CLOBs and BLOBs will **always** be put in a separate table related to the business table. SDO Geometry, depending on the density of the feature, **may** need to be put in a separate table. If the feature is dense, define the geometry in a separate related table - as high volume data could cause physical data management problems such as lengthy unloading and loading times for data changes. If the feature is a low volume of data, as most normally will be, put the geometry column in the business table.

Oracle Locator Layer Definition at MoFR

Oracle Locator is used for MoFR operation spatial data. For Oracle Locator Layers the layer is created through an Oracle's Spatial Data Option (SDO) geometry attribute/column:

- a standard **attribute** in the **entity** with a name of GEOMETRY

- there is no data type and length for the GEOMETRY attribute in Designer so the suggested default is varchar2(1);
- a clear business description of the spatial data item - put in the GEOMETRY attribute description
 - e.g. in the BIOGEOCLIMATIC_Polygon entity the GEOMETRY attribute description is - the Oracle Spatial SDO Geometry representation of a BIOGEOCLIMATIC POLY.
- a standard **column** in the **table** with a name of GEOMETRY
 - data type and length for the geometry column - GEOMETRY Oracle Type SDO GEOMETRY;
- if the spatial data is high volume put the a GEOMETRY attribute/column in a separate, related business **entity** and physical **table**.

Land and Resource Data Warehouse's (LRDW) Spatial Layer Definition (note: the LRDW has recently had it's name changed to the Provincial Geographic Warehouse)

Reporting and analysis data defined for the Land and Resource Data Warehouse is defined as business views of MoFR operational data. These views are mostly denormalized layers holding both spatial (geometry) and attribute data often from more than one operational table.

There is no logical (entity) created for business view layers defined for the LRDW. The business views are defined as physical tables and/or physical views specifically for the LRDW.

Business View Layers for the LRDW:

- are created as Oracle's Spatial Data Option (SDO) layers.
- have a physical table / view defined following ILMB standards
 - prefix all tables and views for the LRDW with the application acronym (3-4 chars) e.g. BEC_BIOGEOCLIMATIC_POLY, RSLT_OPENING_POLY, VEG_COMP_LYR_L1_POLY
- contain a standard column name, Oracle type and length for the geometry column
 - column name: GEOMETRY
 - Oracle Type: SDO GEOMETRY;
- may have views defined at MoFR for use in Spatial Data Replication to extract data to populate the business view layers in the LRDW;
- have a description created for how the LRDW tables / views are populated, including select statement for data selected from operational table(s), so those maintaining LRDW data in the future will understand how it is created / populated / replicated;
 - these descriptions are put in the Notes section of the table / view definition,
 - Notes example: 2007-12-05. This table should really be a materialized view, but due to limitations in Oracle Replication for Spatial tables (as of 10gR2), it is a table instead. The data is retrieved from a view on the MoFR source system and is replicated to the LRDW table using the SDR (Spatial Data Replication) tool.
 - The select statement for the view looks like:
 - SELECT ml.forest_file_id, ... (put the full select statement in Notes)
 - also identify, in the Notes section, the column(s) that trigger updates to the data replicated to the LRDW;
- have a server module diagram created for the LRDW WHSE MODEL showing all tables and views the application has created for the LRDW

See [Integrated Land Management Bureau's](#) web site for additional information.

[Back to top](#)

Logical Data Model - Description

The logical data model consists of:

- Entity Relationship Diagram (ERD)
- Entity Type and Attribute descriptions (the "Data Dictionary")
- Codes spreadsheet

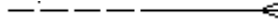
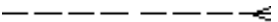


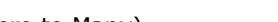
Entity Relationship Diagram (Logical Level)

This is a graphical or pictorial representation of the entity types and their relationships, defined by an analysis of what information the business records and what business rules are used.

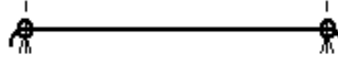
- The logical data model shows the information needs of the business. It is designed at the logical level (separated from the physical implementation) as much as possible, however, the automated transformation process used (logical to physical) requires some physical design considerations within the logical model.
- Each entity type is named (see [Guide S19: Data Naming Standards for Systems Development](#)) and represented by its own rectangle.
- Sub-types of entities can be defined by projects where it makes sense to do so. However, the DBA's may determine that there are performance impacts in using sub-types. See Oracle Designer documentation for a complete discussion on options for implementing sub-types.
- The types of attributes that are displayed on an entity are indicated by the symbol displayed to the left of the attribute name.

Symbol	Description
#	Unique Identifier (UID)
*	Mandatory Attribute
o	Optional Attribute

Relationships (between one or more entities)

- Relationships between entity types are shown (see figures):
 - each relationship is drawn as a line connecting the two entity types;
 - each relationship is given a name that indicates what information it imparts (relationships are named in **both** directions);
 - identifying relationships are indicated by a bar on the relationship line, which shows that a foreign key is being used as a primary identifier (or part of the primary identifier) in the entity type closest to the bar and makes it easy to see which entity types are dependent on other entity types;
 - the *type* of relationship (*cardinality* and *optionality*) is specified as follows: the line style (dash or solid) indicates optionality and the relationship ends indicate cardinality. **Relationship Types - Approximate Graphic Symbol - Oracle Designer:**
 - Zero to Many Relationship (Left to Right)  Many
 - to One Relationship (Right to Left)
 - Zero to Many Relationship (Left to Right)  Many to
 - One Relationship (Right to Left)
 - Zero to One Relationship (Left to Right)  One to
 - One Relationship (Right to Left)
 - Zero to One Relationship (Left to Right)  Zero to
 - One Relationship (Right to Left)
 - Many to Many Relationship (Both Ways Zero to Many) 
 - Note: 1:M relationships must be represented as 0:m on the logical data model. Any 1:M Mandatory requirements must be implemented through the application.
 - Mutually exclusive relationships are represented by an arc covering the 2 or more relationships included (see figure). Please note that the arc would need to be

physically implemented by a stored procedure or trigger, since Designer does not generate anything at the physical level for the arc.



- example diagram showing arc:
- recursive relationships can be used to implement hierarchies and are represented by a single relationship from an entity to itself.

Unique Identifiers/Primary Keys

- Each entity (or entity occurrence) must be uniquely identified. This unique identifier or primary key (physical model) must be stable over time for the data record it identifies.
- The primary key may be made up of a combination of one or more attributes or relationships. It is preferable that this primary key is a stable business key.
- The data model should not contain an extra ID# (surrogate key) on each entity, but instead should use a proper BUSINESS KEY which is unique. If this is not available or the business key is volatile, then a surrogate key is okay, e.g. Project Id.
- If the unique identifier is a surrogate key a unique business key should be defined to avoid duplicate records. Where the entity is strictly for lookup purposes (data is pre-loaded not user entered) in some cases it will not contain a business key.

Entity Type Descriptions ("Data Dictionary")

The logical data dictionary textually describes the entity types drawn on the ERD and includes the following information:

- Entity type descriptions, which contain:
 - the name of the entity type (from the ERD);
 - a brief textual definition of the use and purpose of the entity type (see below - "Definitions");
 - names of attributes (data elements) appearing in the entity type;
 - names that denote business aliases or physical naming decisions made at this stage (physical names must be singular);
 - identifier(s) of the entity type (the attribute(s) and/or relationship(s) that uniquely identifies an instance of that entity type);
 - partition name(s) and corresponding subtypes, if the entity type is further subdivided.
- Attribute (data element) descriptions, which document the following for each attribute:
 - name (see [Guide S19: Data Naming Standards for Systems Development](#));
 - definition (see below -- "Definitions");
 - data type (char is defined as varchar2, numeric, date, etc.);
 - data type of **integer** is not valid – must be data type of number with specified length
 - length including variable length field considerations;
 - valid values or value ranges if necessary (codes are defined in the corporate code tables).
- MoFR does not define global domains for attribute formats.
 - For some standard attributes such as ENTRY_USERID, ENTRY_TIMESTAMP application level domains may be defined.
- Any coded values that will be used must be identified so that they can be reserved at the logical model review (i.e. coded values for lookups, or the use of standard ministry codes, for example silviculture technique codes, inspection type codes, etc.). These are represented by a separate application code table entity. See [Guide S21: Classifying Ministry Data Using Corporate Code Tables](#)
- Projects are allowed to create their own lookup tables to support table-driven rules e.g. rates, etc.

- "Large objects" (BLOBs, CLOBs) data type attributes will be separated into their own entity types, normally with a 1:1 relationship to the main business entity types. For geometry data type attributes, refer to the section [High Volume Data Design](#) for design considerations.
- operational Spatial data is defined as a GEOMETRY attribute (suggested data type of Varchar2(1)) within the business entity or in a separate related entity for large volume data
- Some projects will build temporary tables. If it is a temporary table to be used ongoing by an operational system, then include the table in the logical model. Don't include temporary tables created only for migration purposes.

Foreign key (attributes) information is **not** textually recorded in the logical model, since the existence of the foreign key is already defined by the relationship joining the two entity types on the ERD.

Data Custodian is recorded by Data Administration staff, and is added to the data dictionary later.

Definitions

Entity type and attribute definitions should clearly describe what business information they record, using:

- *precision* - they resolve ambiguities and qualify imprecise terms;
- *completeness* - all terms are defined;
- *clarity* - **plain** English, few if any buzzwords;
- *brevity* - brief and to the point;
- *compatibility* - with other definitions.

Example - Good Definition

CLIENT - Any individual or registered company who is dealing, has dealt, or plans to deal with the ministry. This is effectively the client name and address list for the ministry.

Example - Bad Definition

CLIENT - A ministry client.

Attribute Order

To understand the logical model most easily, attributes should be ordered within each entity as follows:

- First: Primary identifier(s)
- Next: Attributes (these can be grouped by similar subject - e.g. address attributes are together)
- Last: Entry Userid, Entry Timestamp and Update Userid, Update Timestamp (unless they participate as a primary identifier)

Note: The above order will be required for the physical model column order. BLOBs, CLOBs and spatial data types are created in separate, related entities for physical data management.

Attribute Format Standards

The following specific format standards should be followed:

- varchar2 for all character fields instead of char or varchar.
- number with specific length for all numeric fields instead of integer
- DATE format for timestamp fields.

- date fields should use format **YYYY-MM-DD-HH24.M1.SS**

Mandatory/Optional Columns

- Primary Key attributes or relationships are always mandatory.
- The following attributes are mandatory on all data tables: Entry UserID, Entry Timestamp
- The following attributes are optional: Update UserID, Update Timestamp, Effective Date, Expiry Date. However, attributes such as Update UserID and Update Timestamp are mandatory and must be included on all operational data tables when the table participates in data warehousing applications, data replication processing or when required by business processes.

Supporting Documentation (optional)

There may be other documentation that contains further information about the business area, or helps to describe the business in more detail. Supporting documentation may include:

- Statement of Scope (Terms of Reference)
- Requirements Document
- Process Model Diagram, Data Flow Diagram, or other useful diagrams

[Back to top](#)

Physical Data Model - Description

The physical data model consists of a physical model diagram, supporting table and column definitions and Data Definition Language (DDL).

Oracle Physical Model

Oracle Table Generation

In Oracle, a physical data model containing the implementation tables and columns is generated from the logical data model. Logical model to physical implementation mappings are created and refined during the generation process. As the physical model shows the table, column (in ministry standard column order) and foreign key structures that are the basis of all relational databases, once implementation mapping has been refined and the physical model generated there should be very little tweaking. Any modifications should be documented and a list submitted with the physical data model.

Oracle Table Generation Scripts (DDL)

Table creation scripts, or DDL scripts for Oracle databases are generated from the physical data model/table definitions in the Oracle Designer Repository.

General Notes for the Physical Data Model

The following items should be noted:

- The table definitions show the physical structure of the records (tables), fields (column definitions), and indexes which will be implemented in the database (for an example see [Data Model Examples](#), Example 5).
- There must be a one-to-one match between the table definitions and the database tables being implemented on the physical platform.

- Logical model subtypes cannot exist in the physical model; they are resolved into one or more table(s) during table generation. The business data structure is identified in the logical model and can be referred to there if necessary.
- The order of columns (fields) for the physical tables is finalized to ministry standard column order in the table definition.
- Indexes needed for physical implementation are finalized in the table definition.
- In Oracle, if the primary key is a single surrogate key, an Oracle sequence can be used to generate the value for the id. However, if the primary key is a concatenated key, made up of a business key + a surrogate key (sequence), sequencing within the business key is not supported automatically by an Oracle Sequence. Therefore, if this is required, the application must build a separate routine to select max (surrogate key) for a given business key and increment by 1.
- DBA's prefer stored procedures for implementation of complex business rules.
- Use stored procedures or triggers instead of specifying default values for columns.
- Provide physical table size & growth estimates (# of rows, expected yearly growth). Describe in table documentation, not in DDL.
- If any tables have GEOMETRY columns for spatial data the Data type is null and the Oracle Type is MDSYS.SDO_GEOMETRY.
- Additional DDL standards, additional tips for developers are found on the [Database Administration](#) web page
- Both DA and DBA's review DDL.
- For a description of the application delivery process, including DDL, please see the MoFR system development guidelines [Migration Package page](#).

Column Order

For best physical database design, columns should be ordered within each table as follows:

- **First: Primary key column(s)**
- **Next: Columns (these can be grouped by similar subject - e.g. address columns are together)**
- **Last: Entry Userid, Entry Timestamp and Update Userid, Update Timestamp (unless they participate as a primary identifier)**

Also, columns that change at the same time should be placed next to each other, since this can reduce the amount of logging required. This will only make a difference if the application is updating **only those columns** (this means the SQL UPDATE statement must only reference those columns; SQL does not differentiate between columns which are referenced in the UPDATE statement whose value is being changed, from those which are referenced in the UPDATE statement but whose value is being set to the current value).

The way applications are generally written at Forests, a single UPDATE statement is used which updates all columns except the primary key. This statement is used for all updates, regardless of which columns are actually being changed. In these cases grouping of updated columns does not apply since all non-primary-key columns will be logged, so just use standard ministry column order. This is acceptable for most applications. However, if you have an application which only updates a few columns of a table, and the application is run often or updates a large number of rows at a time (e.g. batch processing), then it is important and worthwhile to consider the logging implications. For these high volume transactions, you should consider placing the columns which are updated next to each other, and also code the SQL UPDATE statement appropriately.

The DBA staff is aware of these considerations, and does take them into account during physical model reviews.

BLOBs and/or CLOBs must be placed in a separate table.

Application Code Tables

All codes in MoFR are managed corporately for maximum sharing and minimum redundancy. Codes are physically implemented within Application Code Tables for code lookup and referential integrity. The physical structure of an Application Code Table is created through the data modelling process. A replication process is used to automatically populate and update Application Code Table values from a corporately managed code source.

Projects should not submit insert scripts for Application Code Tables.

Please see [Guide S21](#) for more information about Application Code Tables.

Referential Integrity

A relationship between entities will result in the generation of a foreign key constraint in the physical table definition. The resulting data column(s), if populated, must have a corresponding entry in the associated table. This is referential integrity. Also, entries in the associated table, e.g. lookup table, cannot be removed if there are data columns referencing them.

Referential integrity (Foreign Key constraints) is generally left on. The only exceptions are if the target table is sourced from an external database, or to temporarily remove foreign key constraints for conversion purposes.

Cross Server RI cannot be implemented through foreign key constraints. However, it can be implemented through triggers, but there would be a performance hit. Therefore, generally it is not recommended.

Supporting Documentation

Examples:

There may be other documents that show additional information about the business area, oriented towards physical design decisions.

As well, sometimes it is necessary to make modifications to the Physical Data Model directly, if the result cannot be achieved by generating from the logical model. These revisions should be documented in a separate spreadsheet and submitted with the physical model for review. For example, when there are multiple relationships between two entities, the data column names for the 2nd (or 3rd etc.) relationships are generated from the relationship name. The result may not be desirable and therefore manual name changes might be necessary.

Managing Views - Data Access in an Integrated Environment

Data integration in systems development raises some issues regarding physical access procedures within the integrated database. The following is a brief outline of how general table/view access and security is currently managed.

- The use of interactive SQL in the production environment is strictly controlled.
- The Data Custodian of the table is responsible for ensuring that the ministry's information needs are met by that table. The Data Custodian is responsible for authorizing access to a given table. If another client requires the use of only some of the columns of that table, and is limited from using the full table, then that client (i.e. the "external user") must build a view, have it approved by the Data Custodian, and implement it so that all required ministry areas can access the appropriate data.
- Multiple views can be used to manage access to sensitive fields. The initial view built includes all columns on the base table. A subsequent view can limit access to only non-sensitive columns. "External" systems will likely use the non-sensitive views (*external* here means those systems not developed by the Data Custodian of the table). However, the number of views defined should be kept to a minimum, for example only in exceptional circumstances would there be more than two views defined for any particular table. All

views will have their view names suffixed with "_VW", and then can use another 27 characters for the full name.

- There must be no views with "select *" as part of the view definition.

[Back to top](#)

Logical Data Model Review Process

In general, attendees at logical modelling sessions must include: Data Custodian representative(s), Data Administration representative, contracted staff responsible, and usually a DBA representative. The final approval meeting(s) *must* include a DBA representative for their information leading to the physical modelling effort.

There are usually several modelling sessions scheduled before models are at the stage where they can be validated. The first meeting usually includes an overview of the model's business rules and relationships (the ERD), as a first cut review of the technical quality. During this process, cross-program entity types are identified that may require other Data Custodian resources to resolve issues. The second meeting will attempt to resolve cross-program issues with appropriate staff in attendance representing other Data Custodians where necessary. If all issues are resolved quickly, the cross-program staff can leave and the technical quality review of the model can resume. Subsequent meetings will review the technical quality of the model until it is agreed that the model is complete and correct.

Logical Model Checklist

At the logical modelling meetings, those present should look for:

- *Identification of cross-program entity types* (i.e. entity types of interest to another program area within the ministry) that may require other Data Custodian representatives to attend.
- *Identification of protection of privacy security risks* (i.e. access to data, e.g. personal information, that is private to the business) that may require an **access** userid and date/time be captured to verify proper data access.
- *Identification of historical data needs* (i.e. entity types where data captured in the past has on-going relevance to the business) that may require capturing of historical data.
- *Technical quality of the Entity-Relationship Diagram:*
 - Does the cardinality (one-to-one, one-to-many, many-to-many) and optionality (sometimes, always) of each relationship reflect the true business information requirement?
 - Were relationship names chosen properly? Are they meaningful, do they describe the business? (e.g. a relationship called "has" doesn't describe much)
 - Are identifying relationships marked properly?
 - If multiple relationship paths between the same entity types are really required, then what are the reasons?
 - Are many-to-many relationships truly many-to-many? (90% are not)
 - Are mandatory one-to-one relationships really required or should the two entity types be collapsed into one?
- *Technical quality of the Data Definitions:*
 - Do entity type/attribute names clearly indicate the meaning of what they store? Where platforms restrict the length of column names, are the physical names identified? Note: Oracle automatically makes physical names plural; these must be changed to singular.
 - Do entity type definitions accurately describe the business usage of what the entity type contains (including any cross-program usage)?
 - Are entity type identifiers (primary keys) properly defined, including identifying relationships? Are all identifier attributes mandatory (i.e. no blanks allowed in primary key columns)?
 - Are attributes listed in the proper order (primary identifier, attributes, update userids or timestamps, varchars)?

- Do attribute definitions accurately describe the business usage of what each attribute contains? Is the datatype, length, valid value range(s) listed?
- Do entity types have the proper Data Custodian identified?
- Do attributes fit the entity type (i.e. proper normalization), none there that seem to belong somewhere else?
- Do proposed physical names (abbreviations) follow *Guide S19* abbreviation standards, and are reasonable? Do the abbreviations make sense, capture meaning, don't intrude on other ministry business areas? Are there no abbreviations unless necessary (Oracle - table length 30, column length 30)?
- Do attributes containing codes include a representative sampling of the code values? Are the codes listed new codes, or are they equivalent to or a subset of already existing codes? Does the information stored really need a code or would a Yes/No indicator be better used?
- *Code definition for the ministry's standard code tables:*
 - Is the code definition complete, i.e. description, custodian, update contact?
 - Is the list of code values defined? Refer to [Guide S21](#) for standards for code creation.
 - Are the codes' characteristics (character type, length) reasonable, with thought given to how the codes' use may change in the future?

The data model reviews are usually conducted using the [DA Review Template](#).

[Back to top](#)

Physical Data Model Review Process

Data administration staff participate in the physical data model review process to ensure the physical model is translated properly from the logical data model.

Physical Model Checklist

The following checklist is from the Data Administration perspective only. At the physical modelling meetings, those present will:

- Review logical entity types and corresponding physical tables.
- Review physical model printouts with the logical data definition:
 - Do proposed physical names (abbreviations) follow *Guide S19* abbreviation standards, and are reasonable? Do the abbreviations make sense, capture meaning, don't intrude on other ministry business areas? Are there no abbreviations unless necessary?
 - Are the logical entity type identifiers appropriately and properly translated into physical keys in the physical tables?
 - Are all relationships on the logical ERD properly translated into foreign keys on the appropriate table, using the name of the primary key(s) as defined in the foreign table?
- Review any manual changes made between the logical and physical models. These should be described in a separate document. Are these manual modifications really necessary or could they have been avoided?

Data Base Administration Checklist

Data Base Administration (DBA) staff review the model looking at physical design decisions, including those for database efficiency, and may request further changes to the physical model. In general, DBA staff review:

- Index requirements
- Datatype selections
- Column order

- Review physical model:
 - Are entry points (indexes) fully defined ? Do they support all database access requirements?
 - Are space estimates available for all tables?
 - Are specific views of the data necessary for performance or security reasons?

DBA will not action any test releases until any associated table DDL (eg. Create table scripts) have been approved by Data Admin.

See DBA staff for more complete descriptions of their physical data model review concerns.

Reporting Model

Overview

An application is considered "Reporting" only when using 100% existing data.

- If the data is in its original structure (e.g in a reporting database like DBRP1 or run through a reporting application like the Corporate Reporting System) or using a subset, no model is required.
- If there is a changed structure (e.g. denormalized tables, concatenated columns, data summarization), **a model is required**. e.g. often a warehouse component of the application data model.

Data Currency

Users must be aware of timing mismatches when relating data which may have been transferred to the reporting database at different times, and may therefore have different levels of currency and accuracy.

Modelling Responsibilities

Cross-program Issues

Items to think about during logical modelling stage:

- In our highly integrated environment most system development projects define their own entity types, but will also require information from an entity type(s) whose Data Custodian is a different ministry program area (in this context the "different" entity type is called a "cross-program entity type"). Extra communication and consideration is necessary between all functional areas to ensure that all parties' needs are satisfied. Cross-program entity types require a representative from the Data Custodian to attend modelling working sessions and/or model approval meetings. For example, if a data model in the Silviculture area includes the CLIENT entity type, the appropriate Data Custodian representative should be involved in the Silviculture reviews, when CLIENT information is dealt with.
- When changes to cross-program entity types are being proposed, representatives from **all affected areas** must be involved (not just the Data Custodian). For example, for changes to the OPENING entity type, the Data Custodian (Forest Practices Branch) would have to ensure direct input from all other branches that have existing or planned data linkages.
- Data currency issues must be considered for data replicated or bridged between platforms.

Responsibilities

It must be noted here that the business area staff and contracted staff are responsible for determining the **completeness and correctness** of the logical model information. Data Administration staff are responsible for facilitating communication between participating program areas of the ministry, and for indicating whether the logical model design makes good modelling sense. For example, an architect can create a drawing (model) of what a client has said they want in a house. The architect can even tell the client where standards prevent a particular design or make a design impractical (e.g. if kitchen appliances are built to a standard 32" counter height, and the client has specified a 28" counter height. You can still have the 28" counter height but there may be other things to consider (for example, finding shorter appliances!). But the architect cannot assure the client that the three-room house they requested will serve the client's needs five years down the road (e.g. in five years the client may have three children and not enough room for them). The **business needs** of the client can only be determined, decided upon, and committed to **by the client**. Data Administration staff can indicate that a data model correctly represents the business needs as business area staff have described them, but cannot assure the business area that the data model will meet future business needs.

Model Display Standards

The following are standards for creating easier-to-read data models. These items are mandatory; if they are not included, the model will have to be changed before it can be properly validated. Contact [Data Administration staff](#) for further explanations.

Item: Complex Entity Relationship Diagrams

Explanation:

If a model's ERD is quite complex (large numbers of entity types and/or relationships and/or attributes), some extra steps may have to be taken to display the ERD.

- If the ERD has so much on it that the printing is getting too small to decipher, design sections of the model so that all printing can be easily read. Common sense can be used to determine the limits.
- The layout of the ERD should be planned carefully so that crossed or undrawn relationship lines are kept to a minimum. Entity types should be positioned so that relationship names are adequately shown.
- The diagram will be more readable and understandable if all or most of the relationships can point in the same general direction (e.g. up). This also helps identify circular relationships.
- Similar topic entities should be grouped by subject area and color-coded. Color schemes should be noted in a legend.
- Each ERD should show the "Summary information" e.g. the diagram name, created date, modified date, author, application system, and work area.

Item: Cross-program entity types

Guide Reference: [Cross-program Issues](#)

Explanation:

If access to information from another ministry program area is required, create the external entity(s) to which access is required and create the necessary relationship(s) between it and the entity(s) in the current model. Within the external business area, include only those relationships that are important to the business currently being modelled. The definition for the external entity(s) should include the external entity's description and primary key attributes.

Item: Foreign key attributes in logical model

Guide Reference: [Entity Type Descriptions \("Data Dictionary"\)](#)

Explanation:

Do not put foreign key attributes in the logical model. Relationships between entity types provide that documentation. This is important, since the foreign key is not a true attribute of the entity type. In cases where there is an *identifying relationship*, the **relationship** should be indicated as the identifier (do not use the foreign attribute (key)).

Item: Entity Subtypes

Guide Reference: [Entity Type Descriptions \("Data Dictionary"\)](#)

Explanation:

Where appropriate, subtypes should be used in the *logical* model to show subdivisions of entity types -- a collection of entities of the same type, to which a narrower definition and additional attributes and/or relationships apply. For example, the entity type PRODUCT could be divided into subtypes FABRICATED PRODUCT and PURCHASED PRODUCT. The partitioning attribute should also be named and displayed, for example, PRODUCT TYPE CODE.

Item: Different Attribute (Data Element) Names

Guide Reference: [Entity Type Descriptions \("Data Dictionary"\)](#)

Explanation:

Where the physical name of a foreign key attribute needs to be different from the name in its source table, the physical name can be changed in the table definition. For example, ORG UNIT NO may be the original column name, but the related table's column name should reflect the current business use (e.g. OWNER ORG UNIT, GEOGRAPHIC_ORG_UNIT).

[Back to top](#)

Data Model Tools

- **Oracle Designer** for the logical and physical data modelling for all applications, and table generation scripts for Oracle database implementation
- **IDD** ([Integrated Data Dictionary](#)) for storing and displaying logical and physical data model information for all MoFR applications, and for searching corporate codes

Logical and Physical Data Model Tools

There are many tools available on the market that can help with the development, production, and maintenance of both the entity type relationship diagram and the supporting documentation. The Ministry of Forests and Range has standardized on the use of Oracle Designer CASE tool product. The diagrams produced by these tools are completely integrated with their data dictionary, which can be used to provide the supporting documentation. Because the ability to clearly represent named entity types, properly named relationships, identifying relationships, cardinality, and optionality is essential, *Oracle **must** be used to produce documentation for the logical model and physical data definition*, using the version and operating system currently in use by the ministry. Once the logical model has been completed and the Physical Data Structure has been generated, DBA's will run "create table" scripts, provided by projects, to implement the changes to the database.

Summary of tool uses:

- The developer uses Oracle Designer to create the logical data model.
- The developer uses Oracle Designer to design and generate the physical data model from the logical.
- The developer uses Oracle Designer to generate the DDL from the physical data structure.

- The Database Administration group uses the generated DDL to create the physical database tables.
- The Data Administration group uses the data model to populate IDD.

Supporting documentation can be produced using other tools.

Please see the samples in the [Data Model Examples](#) section of this guide, which clearly illustrate the standards expected.

Tool Standards: Oracle Designer

- Naming of Containers: Use the acronym of the project name as the container name, e.g. **RESULTS**. Do not add version numbers to the container name.
- Naming of ERD Models: If the ERD is split into subsets (more than one subject area ERD), use the Project Acronym plus a subset subject area name for each ERD, e.g. "**RESULTS - Openings**".
- Settings: When generating the Physical Table Definitions from the Logical Entities, in Database Design Transformer, please use the following settings:
 - NO Plurals
 - NO automatic Unique Key creation (ID)
 - NO Prefixes added for: foreign keys, surrogate keys, or columns in general
 - NO Table Prefixes.

Data Dictionary Tools

Before defining new data structures, it is important to determine whether the desired data already exists. A data dictionary helps keep track of what has already been defined.

MoFR Data Dictionary

Definitions (a.k.a. "metadata") for much of MoFR's stored data are available in the [Integrated Data Dictionary](#). The metadata is organized and accessible by application, by entity/table and by attribute/column, and includes who the Data Custodian is along with both logical and physical data definitions for attribute and spatial data. Corporate codes and Organization Unit (ministry office) information can be searched and queried, including office location and phone numbers.

Land Information BC Data Dictionary

Many data definitions for land and resource data are available in the [Land Information BC Discovery Service](#). Land data themes and feature types (classes) are available describing mostly geographic (spatial) data.

Data Model Examples

The following examples have been taken from various applications, all of which have been developed at different times. You will need adobe acrobat to view these files.

The examples have been labelled to indicate how each was produced in Oracle Designer or other tool.

- Example 1: [Logical Data Model](#) (Oracle E-R Diagram) - a subset of the FTA Logical Data Model.
- Example 2: [Sample Supporting Documentation](#) (Oracle Entities and Attributes Report) - for the FTA Logical Data Model.
- Example 3: [Sample Supporting Documentation](#) (Oracle Tables and Columns Report) - for the FTA Physical Data Model.

- Example 3: [LRDW Warehouse Data Model](#) (Oracle Server Model Diagram) – for the LRDW subset of the FTA Physical Data Model.
- Example 7: [LRDW Warehouse Oracle Table Definition](#) (Oracle Tables and Columns Report) -
– for the LRDW subset of the FTA Physical Data Model

[Back to top](#)

[Back to Data Administration Section](#)