Corporate Services for the Natural Resource Sector

Information Management Branch

NRS Data Modeling Standards with EA

Last Updated: March 28th, 2017
Version: 3.2.0
Document: NRS_Data_Modeling_Standards_EA.docx

The page intentionally left blank

Table of Contents

1.0 Version Control

Document Version	Revision	Date	Author(s)	Change Reference
1.0.0	Final	December 19th, 2014	Data Architecture	Initial release of NRS UML Data Modeling standards containing previously published Domain Data Modeling standards and newly completed Logical Data Modeling standards.
1.0.1	Corrections	February 3rd, 2015	Data Architecture	Corrections to UML multiplicity descriptions.
2.0.0	Corrections, Reformatting	August 17th, 2015	Data Architecture	Document name changed from UML Data Modeling Standards for the NRS, Reformatting, Improved readability, Association labelling changes, Stereotypes/tags on keys, tighten up language around should/must
3.0.0	Corrections, Additions	June 13th, 2016	Data Architecture	Addition of statements around Temporal Data, Clarification/re-write of Business Key section, List of Values, File based datasets statement.
3.1.0	Corrections, Additions	November 29th, 2016	Data Architecture	Identifying stereotype replaced by Business Key stereotype, clarification for modeling spatial classes
3.2.0	Corrections	March 28th, 2017	Data Architecture	Aligning standards to the NRS Standards for Modeling with EA v5.0.0 document. (logical data model in requirements and design phase)

2.0 Introduction

2.1 Purpose

This document was created by the Data Architecture section of the Information Management Branch within the Natural Resource Sector (NRS). The purpose of this document is to describe the data modeling standards for the Sector; using Unified Modeling Language (UML) within the Sparx Systems' Enterprise Architect (EA) tool. This guide explains the format, standards, and conventions to be used for all Sector UML data models for information processing within line of business applications. This guide also provides guidance for the Logical data model and should be a reference through the Requirements and Design phases of the Sector's System Development Life Cycle (SDLC).

2.2 NRS Transformation Context

In late 2013 the NRS began a substantial business and systems transformation initiative which includes changes to the tools, processes, and methods for systems development and maintenance. NRS data modeling is being transformed from relational models using the Oracle Designer tool to UML-based models using the EA tool. During the transformation period, pre-existing operational applications already modeled in Oracle Designer will be maintained under the previous ***NRS Data Modelling Standards with Oracle Designer*** standards document.

New systems development and substantial alterations to existing NRS data models will follow the current *NRS Data Modeling Standards with EA* document. Please consult the IMB Data Architect assigned to your project regarding specific application of standards during the transformation period.

This document replaces the following documents published earlier in the transformation period:

- Domain Data Modeling Standards for the Natural Resource Sector (*v1.0 Oct 2014*)
- UML Data Modeling Standards for the Natural Resource Sector (*v1.0 Jan 2015*)

2.3 Audience

The guide is intended for data designers, data architects and data analysts who have knowledge of the techniques and procedures involved in data modeling and will be creating or maintaining object UML data models for the Sector.

2.4 Scope/Exclusions

The scope of this document covers all UML data models delivered to or maintained by the Sector. Where conflicts, if any, are perceived between this document and other standards, the Business Portfolio Manager must be consulted.

2.4.1 Direction on File Based Datasets

The NRS has many file based datasets that are maintained or manipulated using automated tools or partially automated business processes, rather than an NRS standard application interface. Examples of these include Microsoft Excel spreadsheets and Geographic Information System (GIS) datasets maintained in Environmental Systems Research Institute (ESRI) proprietary file formats. These datasets may be valuable for integration with other datasets to support NRS business decisions and in many cases are made accessible to the public later in their lifecycle. Where these datasets are not included in a transformation initiative and are not part of a typical system development project, there may still be value in modeling the conceptual data requirements and/or logical data design. At this time, the NRS standard approach for

integrating and modeling file based datasets during systems development projects is under review. Please consult the assigned IMB Data Architect for guidance as to requirements for modeling these datasets as part of a specific project.

2.5 Assumptions

It is assumed that the audience has working knowledge of the Sector's System Development Life Cycle (SDLC) process, standards around the content of those documents, familiarity with object data modeling and the Sparx Systems' Enterprise Architect (EA) modeling tool used to create and maintain these models.

2.6 Definitions

The following definitions apply throughout this document.

2.6.1 Standards

A standard is a specific statement of the rules and constraints governing the naming, contents, and operations of software. A standard must be followed. There is a contractual obligation on the part of the vendor/developer to adhere to all relevant standards.

2.6.2 Guidelines

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.

2.6.3 Sector

Unless otherwise specified, "Sector" is taken to collectively mean the Ministries and agencies which are included under the umbrella of the Natural Resource Sector (NRS). All are served by a common corporate services division (also known as 'CSNR') and thus Information Management Branch (IMB) with the mandate to formulate and maintain application development standards.

2.7 Contacts

All inquiries regarding these standards should be directed to the Data Architect assigned to the project or contact the IMB Data Architecture Services mailbox at: CSNR.Data.Architecture.Services@gov.bc.ca

3.0 Domain Data Model [optional]

3.1 Purpose

The domain data model, also known as a conceptual data model, is a high-level representation of business information within a project and/or application. The domain model is not an SDLC requirement; however, it may be a useful artefact in early phases of a project to help guide and communicate data requirements.

The purpose of the domain model is to:

- Enable the discussion and discovery of concepts related to the project and application data.
- Outline the current and/or future scope of the business problem, as it relates to information.
- Document the conceptual alignment between high-level business requirements and the data design.

- Provide guidance for data design in the logical and physical data models.
- Communicate business data concepts to an audience with various technical backgrounds; therefore, the model should be simple, described, and easy to interpret.

3.2 Overview

The domain model consists of a class diagram and related descriptions. The data elements on the model will typically describe a person, place, thing, event or concept for which the NRS business has an interest. The model is designed to show enough high-level data to support the business requirements and it also allows for easy interpretation of the data elements and flow without needing a technical background. As such, the model can be used as a communication tool amongst various business stakeholders.

3.3 Domain Model Guidelines

The domain model;

- Is database and technology agnostic; i.e. it is independent of the physical structure in which it may eventually be implemented.
- Includes data elements held within the NRS, as well as data shared with or by external agencies.
- Is modeled in the EA tool as a UML class diagram.
- Is comprised of a diagram and textual description of major classes that describe business data requirements at a conceptual level.

4.0 Logical Data Model **[mandatory]**

4.1 Purpose

The purpose of the logical data model is to provide a detailed business view defining and documenting detailed data requirements as part of overall design. It can also be used to refine the scope of data to be created, determine data placement within the sector data profile where not previously determined, and to determine detailed data sharing plans.

4.2 Overview

The logical model shows 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 designed at the logical level (separated from the physical implementation) as much as possible; however, where an automated transformation process is used (logical to physical) some physical design considerations will be required within the logical model. It is created without any specific computer environment in mind, so little optimization for performance, data storage, etc. is done.

The logical model consists of a class diagram, related class/attribute descriptions, and detailed specifications required for physical transformation. The diagram is used to show classes and display associations between classes in a pictorial format.

The logical model is initiated during the project's requirements phase and is completed, verified, and signed off by IMB Data Architecture during the design phase.

During the design phase, the logical data model is refined and readied for transformation to the physical model. At this time, IMB Data Architecture staff will review the logical model with the designer and Ministry staff to ensure that data integration is considered and that proper data modeling techniques are followed.

4.3 Logical Data Modeling for the Requirements Phase

The logical data model must be initiated during the requirements phase of the SDLC.

Note: At this time, the IMB Data Architect must be engaged; although a detailed model review and/or sign-off is not required. This early engagement will help to provide business context that is necessary for the data architect to review and approve the final logical model.

During this phase, it is expected that the logical model is a high-level representation of the business data while providing guidance to support future design of the project and/or application. The model will contain relevant classes that are clearly described, class associations, and business keys.

The purpose of the logical data model during the requirements phase is to:

- Document the alignment between high-level business requirements and the data design.
- Provide guidance for future detailed data design (detailed logical model and physical data model).
- Communicate business data concepts to an audience with various technical backgrounds; therefore, the model should be simple, described, and easy to interpret.

4.3.1 Logical Data Model Guidelines for the Requirements Phase

This section describes the recommended guidelines for modeling data during the requirements phase of the SDLC.

During the requirements phase, the logical model:

- Is database and technology agnostic; i.e. it is independent of the physical structure in which it may eventually be implemented.
- Includes high-level data elements held within the NRS, as well as data shared with or by external agencies.
- Is modelled in the EA tool as a UML class diagram.
- Is comprised of a diagram and textual description of major classes that initially describe the business data requirements at a high level.

The following list describes the recommended characteristics of a logical model throughout the requirements phase:

- **Classes**
  - Only major classes of significant importance to the business are defined.
  - Class names should follow standards listed in Section 6.1 - General Format of Class Names.
    - *Note: Physical Name tagged values are not required at this time.*
  - Class descriptions should be included which contain:
    - A clear, concise and understandable textual description of the purpose of the class. Refer to Section 6.2 – Class Descriptions for more details.
  - May be colour-coded as per class type. If colour-coding of classes is used:

        - Refer to Section 6.6 - Colour Coding Classes for a list of standard colours.
        - The class diagram should contain a legend of class type.
- **Attributes**
    - Business key attributes must be included where they are known. Where attributes are specified;
        - Attribute names should follow standards listed in Section 7.1 - General Format of Attribute Names.
            - *Note: Physical Name tagged values are not required at this time.*
        - Attribute descriptions, should contain a clear, concise and understandable textual description of the purpose of the attribute. Refer to Section 7.2 – Attribute Descriptions for more details.
        - Attribute data lengths, types and multiplicity are not required.
- **Class Associations**
    - Associations are displayed using UML notation.
    - Associations between classes may be defined with multiplicity; with the exception of Generalizations.
    - Labels on the associations are not required.
    - Refer to Section 5.0 – Association Characteristics for more information.
- **Keys**
    - Business keys showing the unique business identifier for the class must be added where known. Refer to Section 9.0 – Defining Class Uniqueness for more details.
- **Lists of Values**
    - Lists of standard terms or codes are only required where important to describe high level requirements. If lists are defined, refer to Section 6.4 – Lists of Values for more details.
- **Normalization**
    - The model is currently at a conceptual level; therefore, normalization is not required.

4.4 Logical Data Modeling for the Design Phase

The logical data model is refined, validated and signed-off during the design phase of the SDLC.

The logical model:

- Is database and technology agnostic; i.e. it is independent of the physical structure in which it may eventually be implemented.
- Includes data elements held within the NRS, as well as data shared with or by external agencies.
- Is modelled in the EA tool as a UML class diagram.
- Is comprised of a diagram and textual description containing all classes and attributes that reflect the detailed business data requirements at a design level.

4.4.1 Logical Data Model Standards for the Design Phase

This section describes the required standards for modeling data during the design phase of the SDLC.

The following list describes the characteristics of a detailed logical data model:

- **Classes**
  - Identify all classes within the scope of the business requirements being designed.
  - Class names must follow standards listed in Section 6.1 - General Format of Class Names.
  - Class descriptions must be included which contain:
    - A clear, concise and understandable textual description of the purpose of the class. Refer to Section 6.2 – Class Descriptions for more details.
  - Must be colour-coded as per class type. Refer to Section 6.6 - Colour Coding Classes for more standard colours.
  - Must contain a legend of class type on the class diagram.
- **Attributes**
  - Include a complete list of attributes (data elements) for each class.
    - NOTE: Mandatory logging/auditing and time tracking attributes will be automatically applied in the physical data model. These attributes are not to be specified on the logical model.
  - Attribute names must follow standards listed in Section 7.1 - General Format of Attribute Names.
  - Attribute descriptions, must contain the following for each attribute:
    - A clear, concise and understandable textual description of the purpose of the attribute. Refer to Section 7.2 – Attribute Descriptions for more details.
  - Attributes must appear in the class in logical order as per Section 7.3 – Attribute Order.
  - Data types are required as per Section 7.4 – Attribute Data Types.
  - Data lengths are required as per Section 7.5 – Attribute Data Lengths.
  - Attribute Multiplicity is required as per Section 7.6 – Attribute Multiplicity.
- **Class Associations**
  - Associations are displayed using UML notation.
  - Associations must be labelled in both directions and must show multiplicity.
    - NOTE: Associations between a list/code class and a business class do not require labels.
    - Generalizations are not labelled and do not require multiplicity.
  - Refer to Section 5.0 – Association Characteristics for more information.
- **Keys**
  - All NRS classes, with the exception of list and code classes, require a business key to define uniqueness. If a suitable business key cannot be defined, please discuss with IMB Data Architecture.
  - Foreign key (attributes) information is ***not*** textually recorded in the logical model, since the existence of the foreign key is already defined by the association joining the two classes on the diagram.
  - Surrogate keys are generated through transformation and are not typically added as attributes within the class.
  - For definition of types of keys, refer to Section 9.0 – Defining Class Uniqueness for more information.
- **Lists of Values**
  - Lists of standard terms or codes that part of the detailed data design must be reflected on the model. Refer to Section 6.4 – Lists of Values for more information.
- **Normalization**
  - The logical model is expressed in Third Normal Form.

5.0 Association Characteristics

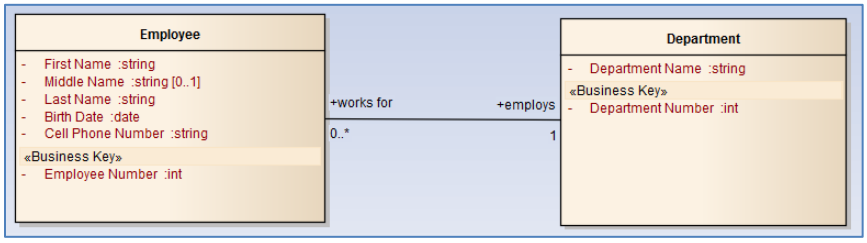
The term *Association* is used to describe how classes within NRS data models are connected to one another.

This section further describes various types of associations commonly seen within NRS data models as well as the standards for how they are to be represented within the models.

5.1 Association Types

An *association* is a link between two classes. Generally speaking, NRS associations are represented as solid lines between classes and are bi-directional.

As shown in the example below, a standard association contains role name labels at each line end and includes multiplicity.

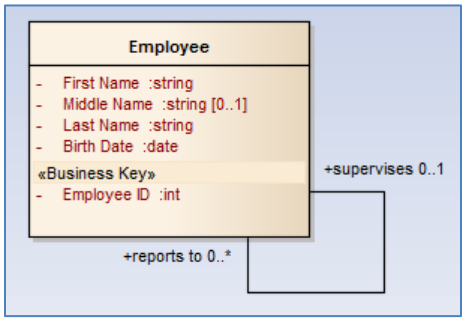


Association example as shown in a logical data model

Although a data model will typically contain standard associations, there may be some cases where the association may be further defined as being Recursive or Identifying.

5.1.1 Recursive Association

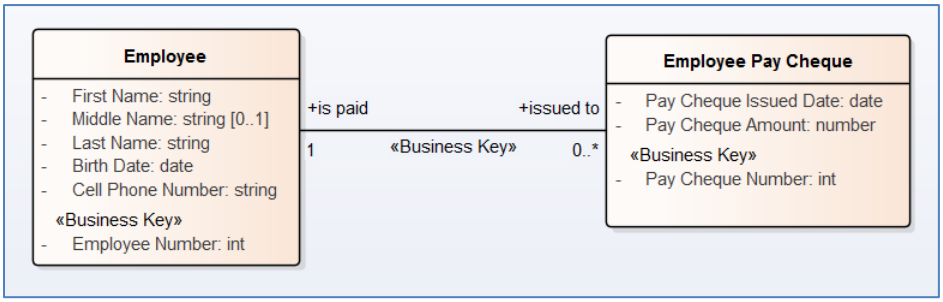
A recursive association; also referred to as a reflexive association, can be used to represent a hierarchy of data within a single class. A recursive association is illustrated by a “pig’s ear” type line. The association is read bi-directional, contains role name labels at each line end, as well as multiplicity.



Recursive association example as shown in a logical data model

5.1.2 Identifying Association

An identifying association represents an association by which the business key from one class is used as the business key (or part of the business key) in a linked class. Identifying associations are indicated on the model through adding a *<<Business Key>>* stereotype to the association between classes.



Identifying association example as shown in a logical model. Employee Number from the Employee class will be used as part of the Business Key in the Employee Pay Cheque (i.e. Employee Number + Pay Cheque Number).

5.1.3 Aggregation and Composition

These types of associations are not typically permitted within NRS data models. Please discuss any usage with the IMB Data Architect assigned to the project.

5.2 Association Role Label Naming

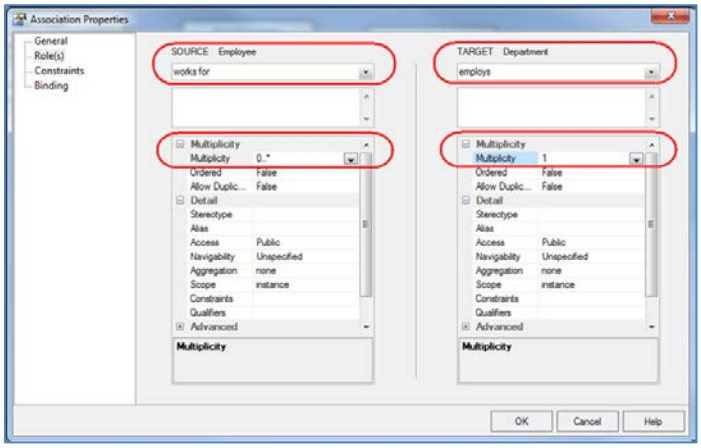
Associations can be further defined through applying role name labels and multiplicity to the association properties. Use of these properties helps to clarify the business data that flows between the classes.

The general rules for naming associations are as follows:

- Labels are required on all associations except list or code class associations. Where associations are used, the following standards apply:
  - Associations are labelled in both directions.
  - Association labels must be meaningfully named.
  - Avoid "catchall" phrases (i.e. 'has', 'has a' 'is', 'is a', 'related', 'associated') in favour of more descriptive association names.
  - Association names do not contain '_' (underscores).
  - Association names are all lower case.
  - Association names may contain spaces.

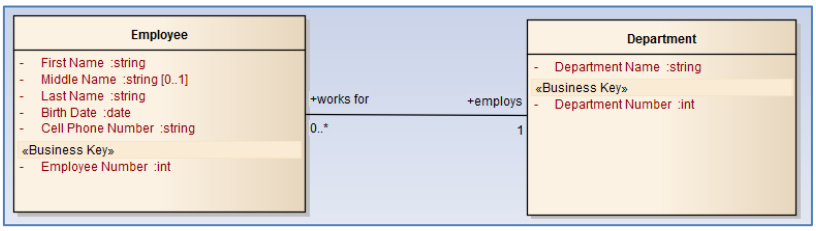
5.2.1 Applying Association Labels in the EA Tool

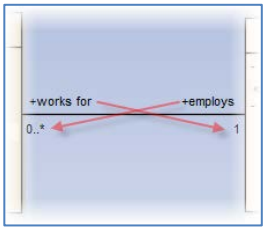
Association labels are created within the *role* fields of the Source Role and Target Role sections in the association properties dialog box.



Role/Multiplicity for EMPLOYEE and DEPARTMENT class (as seen in EA12)

The following example reflects bi-directional labelling within the EA tool.





This graphical representation is the same as saying in plain words:

- An Employee ***works for*** **1** Department (reading left to right)
- A Department ***employs*** **0 to many** Employees (reading right to left)

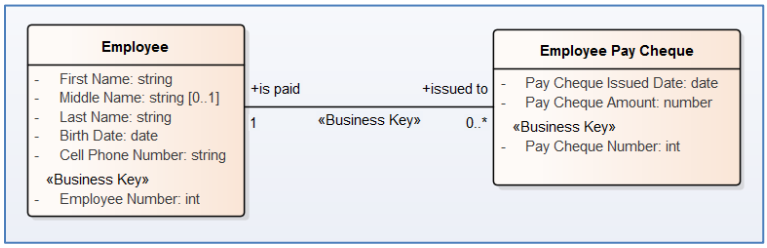
5.3 NRS Association Stereotypes

In order to provide further meaning to an association between two classes, a stereotype might be applied. A stereotype can be thought of as a label or a category in which the association can be interpreted by a human (on a model) or by a computer (building a script to transform into physical data).

The following stereotypes may be used on associations within a NRS class diagram:

Stereotype	Description
Business Key	Indicates the business key for a class requires the business key from an associated class in order to define uniqueness.

An association that has been assigned a stereotype will be shown on the model with the stereotype label on the association. The following example shows an identifying association between two classes.



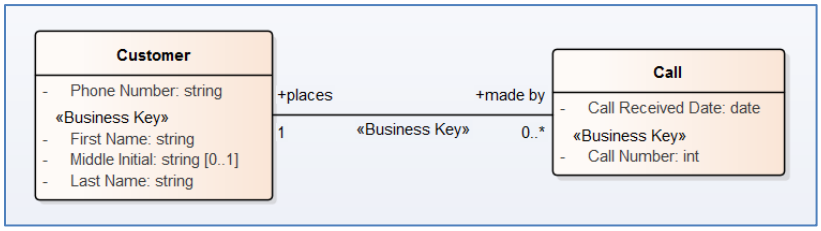
Example of an identifying association using the <<Business Key>> stereotype in a logical data model.

5.4 Association Multiplicity

Multiplicity is a UML term used to describe the optionality and the range of instances of an association between two classes.

In the example below, the multiplicity is implied through the numbers of each side of the “places” association.

Reading Left to Right →“*A Customer places zero to many Calls”*
Reading Right to Left → “*A Call is made by one Customer*”



Example of bidirectional association with multiplicity.

Using the example above, here are other methods to use multiplicity:

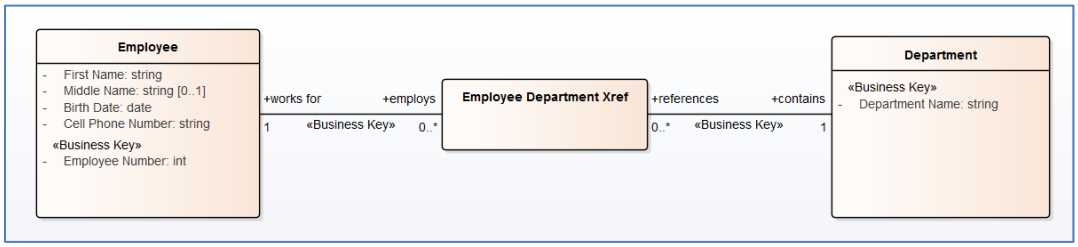
UML Symbol	Customer – Call Example
1 — 0..*	Each Customer places zero to many Calls Each Call is made by one Customer
0..1 — 1..*	Each Customer places one to many Calls Each Call is made by zero to one Customer *(sometimes placed by non-Customer)*
1 — 0..1	Each Customer places zero to one Call Each Call is made by one Customer
0..1 — 0..1	Each Customer places zero to one Call Each Call is made by zero to one Customer
0..* — 0..*	Each Customer places zero to many Calls Each Call is made by zero to many Customers *(sometimes conference call)*
1 — 3..5	Each Customer places three to five calls Each Call is made by one Customer

5.5 Representing Many-to-Many Associations

All many-to-many associations are to be resolved in the logical data model. Where classes are created purely for the purpose of resolving the many-to-many association, the new class should be named with the following standards:

- ***<ClassName1> <ClassName2> Xref***

Where **<ClassName1>** is the name of the first parent and **<ClassName2>** is the name of the second parent.



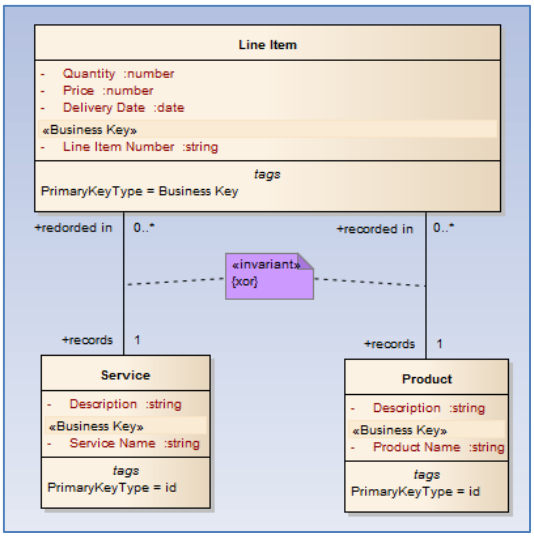
Example of a resolved many-to-many association between employees and departments.

If the combination of the two class names is more than 30 characters, a tagged value must be added to the Xref class to indicate the physical table name to be used. Abbreviations may be

used to meet the physical size limitations. Please refer to Section 10.0 – Logical to Physical Transformation Guidelines.

5.6 Representing Mutually Exclusive Associations

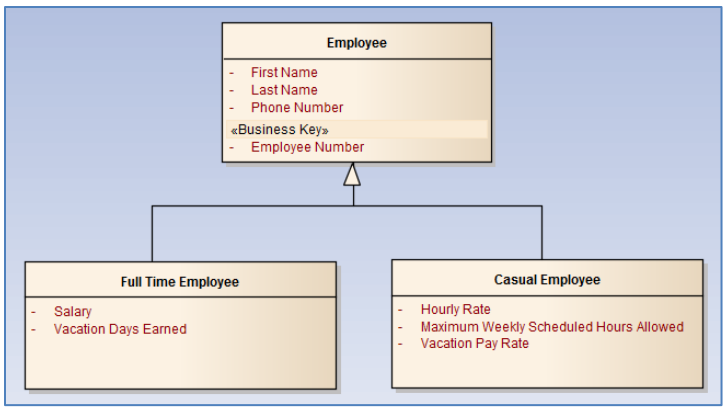
Mutually exclusive associations are represented by a constraint between two or more classes.



Example of an 'XOR' constraint to represent a mutually exclusive association in a logical data model. Tags are shown for descriptive purposes.

5.7 Resolving Sub-Types Through Generalizations

The term *Generalization* is used to describe inheritance between classes. This section further describes various types of associations commonly seen within NRS data models.
In the example below, a generalization has been used to imply the sub-type "Full Time Employee" and "Casual Employee" inherits attributes from their super-type "Employee". There is a dependency between the super-types and sub-types. On NRS data models, generalizations do not contain labels or multiplicity.



Generalization example as shown in a logical data model

6.0 Class Characteristics

This section defines the characteristics required for modeling classes.

6.1 General Format of Class Names

When naming classes, careful thought and planning should be considered in order to ensure names are clear, concise, and tied to business purpose. The class names must be meaningful to the business so that a person reading the model can easily interpret the intent of the data being captured. The following list describes the additional standards for naming classes within NRS data models.

A standard class name:

- Must be repeatable (different people from different areas of the sector reading the name at different times must have the same understanding of what the name means -- a corporate-wide use of the name).
- Must be in mixed case, delimited by spaces between words with each new word capitalized.
- Must consider size limitations:
    - *NOTE: There is no limit of characters for a class name as a descriptive, non-abbreviated, class name is encouraged. However, where the physical implementation is Oracle-based, the table name must not exceed 30 characters. Where the logical class name is longer than 30 characters, the physical name must be shortened to 30 characters or less. This is achieved through applying a Physical Name tagged value to the class. Please refer to Section 10.0 – Logical to Physical Transformation Guidelines.*
- May use a single noun to describe the purpose of the class (i.e. Employee).
- May use modifiers to qualify the class name for uniqueness (i.e. Full Time Employee and Casual Employee).
- Must use singular names to emphasize that the class is a pattern for every instance of the object that is being defined (i.e. Use “Reservation”, not “Reservations”).
- May only use abbreviations or acronyms where no combination of names would produce an acceptable result.
- May require a standard suffix where the class is defined as a specific type. The following list of suffixes can be used to provide clarification of the class’ intent. Please discuss any deviations with the IMB Data Architects.
    - Class objects used to define a list of standard terms must be suffixed with “List”
    - Classes used to define a list of code values must be suffixed with “Code”.
    - Classes that are used as a cross reference between two classes must be suffixed with “Xref”. Refer to Section 5.5 – Representing Many-to-Many Associations.

6.2 Class Descriptions

Descriptions are used to enhance the understanding of class names. Generally, the description must explain what the class is to non-application personnel, or non- business users.

In general, the descriptions should follow these guidelines:

- Where possible, the intent of the class should be shown. This allows the reader to understand the rationale for the class and fit their own understanding of the business into the one represented by this class.
- For each class, if the description contains an acronym or abbreviation in its definition, the acronym must be fully qualified, in the suggested format:
    - “Full Name (acronym)”
    - i.e. “Universal Transverse Mercator (UTM)”

Class Description Examples:

Class Name	Class Description
Customer	Customer is an individual, also known as a camper, who has made a reservation request.
Reservation	Reservation holds detailed information for a Parks campground customer reservation to allow reservation fulfillment and good customer service. Available through calling the Parks Call Centre service or making the reservation with a campground Park Ranger.

6.3 External Classes and Other Data References

Sharing of data between various sources is a common occurrence within the NRS. In order to understand data dependencies, it is necessary to represent the external sources on a class diagram. Where an external data reference is required, the class diagram will include the associations(s) as a high-level representation. At this time, implementation methods of the physical associations to external data sources have not been determined.

6.3.1 External Classes

Please note; the standards around external classes are being refined. If data from external sources is required for a project, discuss requirements with the IMB Data Architects. Any external classes represented on a class diagram will need to include the following:

- A new class, named the same as the external class name.
- The shared business key from the external class.
- Any attributes from the external class may be added to provide further details.
- An association from the external class to the business class.
- The external class is coloured light blue as per the standards in Section 6.6 – Colour Coding Classes.

6.4 Lists of Values

Most business applications within the Sector use lists of values for data capture and validation. A list of values, also known as reference data or an enumerated list, is used when consistent business meaning is needed for the information item being captured. Additionally, a list of values is used to standardize terminology used within the business and across business areas.

The content of a list is determined and authorized by business area custodians. A list may contain business terminology or it may contain values determined through an external authority. Due to the high usage and potential sharing of a list of values, it is important that the list is fairly stable and not subject to frequent change. Options are typically discussed with the IMB Data Architect throughout data modeling activities.

The NRS defines a list of values as a categorized set of ***standard terms*** and/or ***code values*** that are permitted for use within a business application. For example; a drop-down list on a screen.

- **Standard Terms** are represented as self-explanatory values. For example; days of the week -- Monday, Tuesday, Wednesday, etc. A term may or may not require further

description for clarity. For this reason, a list of terms has been further categorizes into *simple* and *described* lists.

- A **simple list** uses real-world terms that do not require further definition. The terms are self-explanatory and easily interpreted. For example;

Standard Term
Monday
Tuesday
Wednesday

- A **described list** uses real-world terms that require further description. The terms may be easy to understand, but the term itself does not provide a clear interpretation of its purpose within the business context. For example;

Standard Term	Description
Monday	Monday is the first day of the business week
Tuesday	Tuesday is the second day of the business week
Wednesday	Wednesday is the third day of the business week

- **Standard Code Values** consist of a shortened word/number that is used as a substitution for another word or phrase. In most cases, code values are abbreviations or acronyms. For example; days of the week -- "MON", "TUE", "WED", etc.

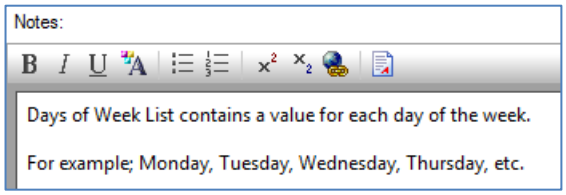
Code Value	Description
MON	Monday
TUE	Tuesday
WED	Wednesday

6.4.1 Modeling Standard Terms via Simple Lists

To define a simple list within a data model, the following standards are to be applied.

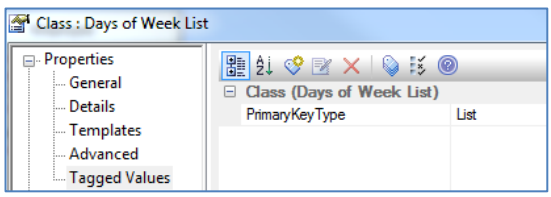
A simple list:

- Is represented as a Class Object.
- Class description contains examples of the terms used within the list.

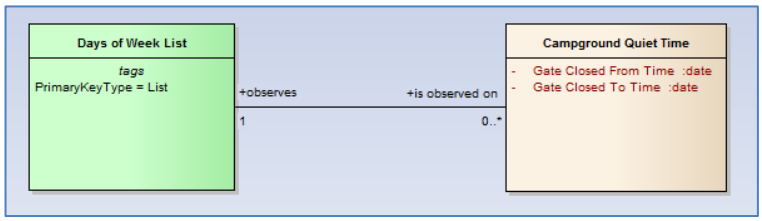


Example of a description for a simple list.

- Contains a tagged value to specify the class as a "List". This tagged value will build a standardized table structure during physical transformation. Refer to Section 10.0 – Logical to Physical Transformation for more details.



- Is colour coded as per standards in Section 6.6 – Colour Coding Classes.
- Is named as per standards in Section 6.1 – General Format of Names.
- Contains no attributes.
    - *Exceptions: If attributes within a standard list table require customization, please refer to Section 10.0 – Logical to Physical Transformation and discuss requirements with IMB Data Architecture.*



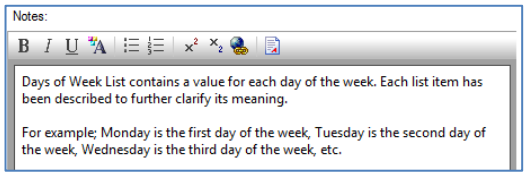
Example of Simple List. Tags shown in class for descriptive purposes.

6.4.2 Modeling Standard Terms via Described Lists

To define a described list within a data model, the following standards are to be applied.

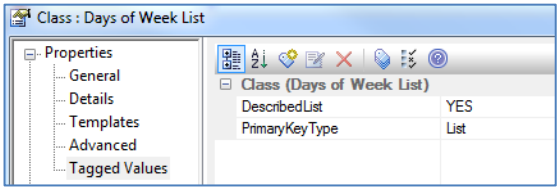
A described list:

- Is represented as a Class Object.
- Class description contains examples of the terms used within the list.

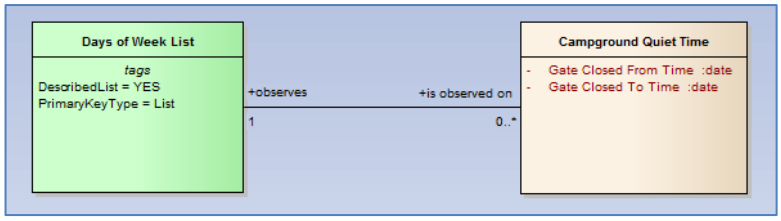


Example of a description for a described list.

- Contains tagged values on the class to specify:
    - The class is a “List”. This tagged value will build a standardized table structure during physical transformation.
    - The list is “Described”. This tagged value will add a “description” column to the standardized table structure during physical transformation.
    - Refer to Section 10.0 – Logical to Physical Transformation for more details.



- Is colour coded as per standards in Section 6.6 – Colour Coding Classes.
- Is named as per standards in Section 6.1 – General Format of Names.
- Contains no attributes.
    - *Exceptions: If attributes within a standard list table require customization, please refer to Section 10.0 – Logical to Physical Transformation and discuss requirements with IMB Data Architecture.*



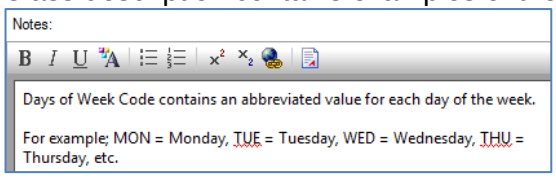
Example of Described List. Tags shown in class for descriptive purposes

6.4.3 Modeling Code Lists

To define a simple list within a data model, the following standards are to be applied.

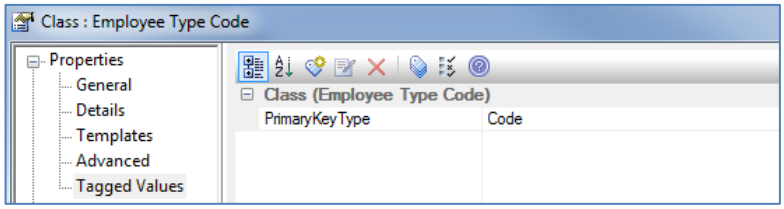
A code list:

- Is represented as a Class Object
- Class description contains examples of the terms used within the list.

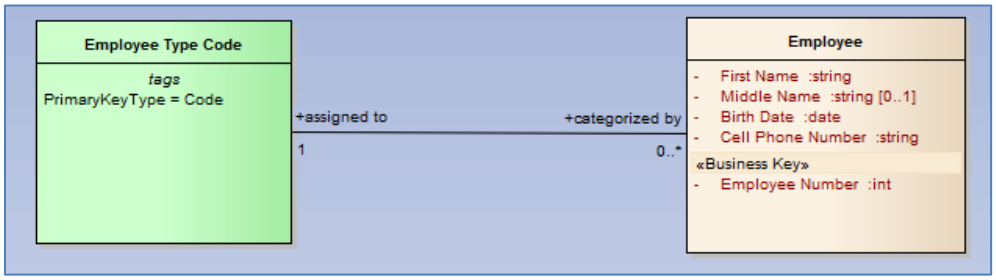


Example of a description for a code list.

- Contains a tagged value to specify the class as a "Code" list. This tagged value will build a standardized table structure during physical transformation. Refer to Section 10.0 – Logical to Physical Transformation more details.



- Is colour coded as per standards in Section 6.6 – Colour Coding Classes.
- Is named as per standards in Section 6.1 – General Format of Names.
- Contains no attributes.
    - *Exceptions: If attributes within a standard code table require customization, please refer to Section 10.0 – Logical to Physical Transformation and discuss requirements with IMB Data Architecture.*



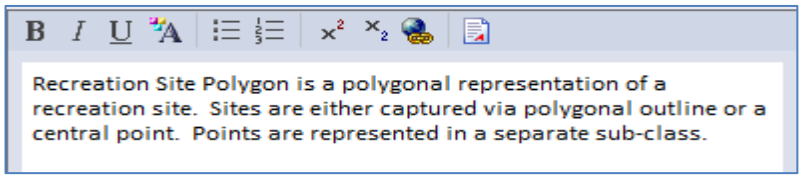
Example of a Described List. Tags shown in class for descriptive purposes

6.5 Modeling Spatial Classes

To define a spatial class within a data model, the following standards are to be applied.

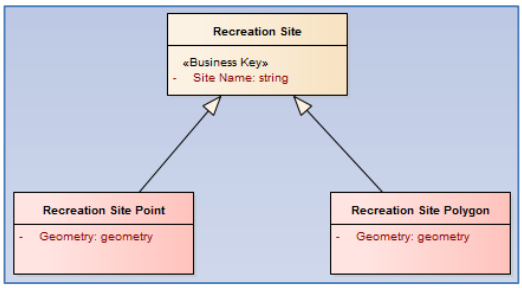
A spatial class:

- Is represented as a Class Object.
- Class description contains a clear description of the geometry and the geometry type.



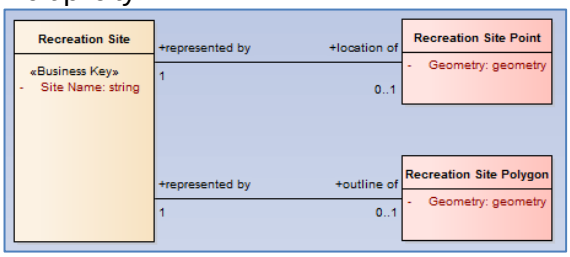
Example of a description for a spatial class.

- The class must contain one and only one GEOMETRY attribute.
- A class must contain only one geometry type (point, line, or polygon).
- If the geometry can contain multiple types then separate types must be modeled;
    - if each instance can have only one type (mutually exclusive), i.e. a recreation site can have either a centre point or a polygon outline, then model as a generalization,



Example of spatial class with two mutually exclusive geometry types.

    - if each instance can have multiple types, i.e. a recreation site can have a centre point and a polygon outline, model as child classes and use appropriate multiplicity.



Example of spatial class that allows one of each geometry type (point and/or polygon).

- Is colour coded as per standards in Section 6.6 – Colour Coding Classes.
- Is named as per standards in Section 6.1 – General Format of Names.

6.6 Colour Coding Classes

Colour must be applied to enhance the readability of the diagram. The standard colour fill for classes on the diagram are as follows:

Class Type	Assigned Colour	HSL	RGB	Example
Business Class *	White	0,0,255	255,255,255	
Lists of Values **	Lt Green	77,226,229	215,252,206	
External Reference	Pale Blue		153,204,255	
History/Audit	Lt Orange		255,173,91	
Spatial Class	Rose	4,255,240	255,228,225	

(*) Note: EA 10 and EA 12 have different default colours for typical business classes. Regardless of which version is being used, the recommendation is to leave the default colour as-is. In EA the classes will appear as off-white, whereas in EA 12, the classes will appear as white.

Additional colours may be included within a data model diagram if required. All colours must be shown within a legend on the diagram.

7.0 Attribute Characteristics

7.1 General Format of Attribute Names

When naming attributes, careful thought and planning should be considered in order to ensure names are clear, concise, and tied to business purpose. The attributes must be meaningful to the business so that a person reading the model can easily interpret the intent of the data being captured. The following list describes the additional standards for naming attributes within NRS data models.

A standard attribute name:

- Must be unique within the class.
- Must be in singular form to emphasize that the attribute name (and corresponding definition) is a pattern for every instance of the object that is being defined.
- Must be mixed case, delimited by spaces,
- May have size limitations:
    - *NOTE: NOTE: There is no limit of characters for an attribute name and a descriptive, non-abbreviated, attribute name is encouraged. However, where the physical implementation is Oracle-based, the column name must not exceed 30 characters. Where the logical attribute name is longer than 30 characters, the physical name must be shortened to 30 characters or less. This is achieved through applying a Physical Name tagged value to the attribute. Please refer to Section 10.0 – Logical to Physical Transformation Guidelines.*
    - May contain terms (modifiers) that define the attribute further or allow the attribute to be uniquely identified within the class. Modifiers should be used sparingly and are only used to help provide clarity of an attribute within a class. For example;

Recommended		Not Recommended	
Class Name	**Attribute Name**	**Class Name**	**Attribute Name**
Customer	*First* Name	Customer	Name
	Last Name		Name
	Cell Phone Number		Phone Number
Reservation	*Reservation* Number	Reservation	Number
	Cancellation Reason		Reason

- May use acronyms where they are universally understood, accepted, and stable. For example; SIN (Social Insurance Number). Note: Application acronyms may be applied where a class is specific to one application within a shared database environment; however, use of application acronyms is expected to be minimal.

- May use abbreviations when the attribute length needs to be shortened. Please refer to the ***NRS Physical Data Modeling Standards*** for a list of standard abbreviations.
- May require a standard suffix where the attribute is defined as a specific type. The following list of suffixes can be used to provide clarification of the attribute's intent. Please discuss any deviations with the IMB Data Architects.
    - DATE attributes must be suffixed with "Date".
    - TIMESTAMP attributes must be suffixed with "Timestamp".
        - *NOTE: As per the physical data model standards, TIMESTAMP is only used when a true sub-second granularity is required.*
    - GEOMETRY attributes must be named "Geometry".
    - GUID attributes must be suffixed with "GUID".
    - BOOLEAN attributes must be suffixed with "Ind".

7.2 Attribute Descriptions

Descriptions are used to enhance the understanding of attribute names. Generally, the description must explain what the attribute is to non-application personnel, or non-business users.

In general, the descriptions should follow these guidelines:

- Where possible, the intent of the attribute should be shown. This allows the reader to understand the rationale for the attribute.
- Where there may be confusion between two similar attributes in a class, either because their names are similar or because their descriptions are similar, there must be explicit text which explains the difference between the attributes.
- For each attribute, if the definition contains an acronym or abbreviation, the acronym/abbreviation must be fully qualified in the following format:
    - "Full Name (acronym)"
    - i.e. Social Insurance Number (SIN)
- An example of the usage of the acronym or attribute should be included for clarity.

Attribute Description Examples:

Class Name	Attribute Name	Attribute Description
Customer	First Name	First Name contains the legal first name of a customer using the Reservation Booking System.
Reservation	Reservation Number	Reservation Number is a system-generated number used to uniquely define the reservation made within the Reservation Booking System. A customer may use this number as a reference when enquiring about their reservation.

7.3 Attribute Order

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

- **First:** Business key
- **Second:** Business attributes (these can be grouped logically by similar subject – i.e. address attributes are together; and, where applicable)
- **Third:** Spatial attribute (if present)

- **Fourth:** Temporal attributes (where applicable)
- *NOTE: Mandatory logging/auditing and time tracking attributes will be automatically applied in the physical data model. The following attributes do not apply to the logical model:*
    - *REVISION_COUNT, CREATE_USER, CREATE_DATE, UPDATE_USER, UPDATE_DATE*

7.4 Attribute Data Types

Each attribute on a logical model must be assigned a data type. The data type will determine what type of information will be physically stored in the database. Because the intent of the logical model is to be technology agnostic, the data types to be used in the model will be generalized; to be later transformed into an Oracle-friendly format during physical implementation.

The following list describes the standard NRS data types to be used within the EA Tool, as well as a description of what the data type will be transformed to for physical implementation in an Oracle database. Use of data types not within the NRS standard list will require approval from IMB Data Architects and Database Administrators. For a list of restricted data types, please refer to the ***NRS Physical Data Modeling Standards***.

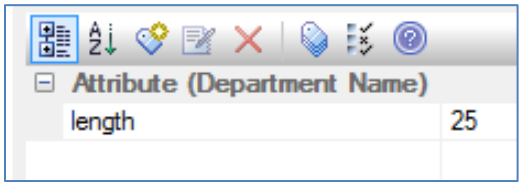
Data Type	Implementation Description - Oracle Database
int	– Used to describe whole numbers (i.e. no decimals) – Will be implemented as an Oracle NUMBER – Must have tagged value **precision** to describe the number of digits required for the attribute
number	– Used to describe numbers that require decimals – Will be implemented as an Oracle NUMBER – Must have tagged value **precision** to describe the number of digits required for the attribute – Must have tagged value **scale** to describe the number of decimal places required for the attribute.
boolean	– Will be implemented as an Oracle VARCHAR2(1) datatype with a constraint on values to 'Y' and 'N' – Must be mandatory. If optional is required, discuss with IMB Data Architecture.
string	– Will be implemented as an Oracle VARCHAR2 datatype – Must have tagged value **length** to describe the length of the column
date	– Will be implemented as an Oracle DATE datatype
timestamp	– Will be implemented as an Oracle TIMESTAMP datatype *(where sub-second granularity is needed)*
geometry	– Will be implemented as an Oracle Spatial column
guid	– Will be implemented as an Oracle RAW(16) datatype
blob	– Discuss with IMB Data Architecture o DBA Approval is required to implement unstructured data. May be implemented as a new table with a 1:1 association to the business table.

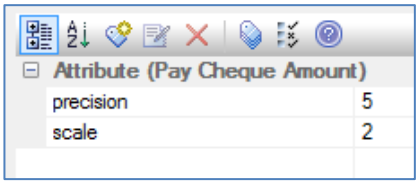
7.5 Attribute Data Lengths

The length of a data attribute is identified through a “Tagged Value” within the EA tool. Tagged values are used during the physical transform and represent the physical length of a column. Tagged values include the following:

Tagged Value	Description	Applicable Data Type
length	Indicates the physical length of a non-numerical attribute, such as a String.	string
precision	Indicates the total digits required for a numerical attribute.	integer number
scale	Indicates the number of decimal places required for a numerical attribute. *NOTE: When a decimal value is required, the **scale** tag is applied in conjunction with the **precision** tag. For example;* • *99.99 would require **precision** = 4 and **scale** = 2* • *99999.999 would require **precision** = 8 and **scale** = 3*	number
n/a **no tag required*	No lengths need to be defined	date timestamp boolean guid geometry

As seen in the example below, tagged values are identified within the “Tagged Value” section of the attribute properties within the EA tool.

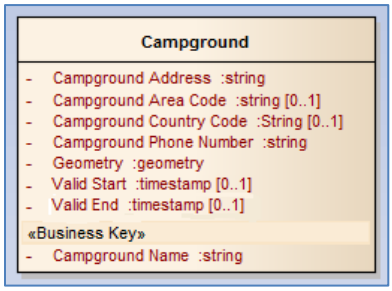




7.6 Attribute Multiplicity

Determining whether an attribute is optional or mandatory is defined through the attribute multiplicity Lower bound and Upper bound within the attribute “Detail” properties in the EA tool.

- **Mandatory** - By default, all attributes diagrammed in UML are mandatory (i.e. its Multiplicity Lower Bound is 1 and its Upper Bound is 1).
- **Optional Attribute** – Attribute name followed by a [0..1] implies an optional attribute (i.e. its Multiplicity Lower Bound is 0 and its Upper Bound is 1).



Example of attribute multiplicity in a class

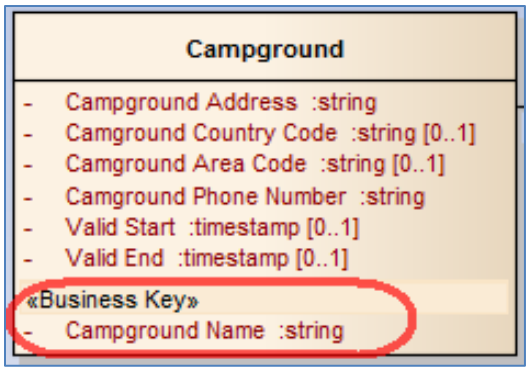
7.7 Attribute Stereotypes

In order to provide further meaning to an attribute, a stereotype might be applied. A stereotype can be thought of as a label or a category in which the attribute can be interpreted by a human (on a model) or by a computer (building a script to transform into physical data).

The following stereotypes may be used on attributes within a NRS class diagram:

Stereotype	Description
Business Key	Indicates the attribute is part of a business key for the class.

Any attribute with a stereotype will be shown in the class diagram under the grouping in which it was assigned. The following example shows that the *Campground Name* attribute is assigned as the business key.



Example of a Business Key stereotype in the logical model

8.0 Temporal Data

Temporal data refers to any data with one or more associated time periods during which that fact is deemed to be effective or valid.

Temporality allows the business to determine:

- The validity of a (data) fact at a specific point-in-time. In other words, when the fact is considered “true”. This could include validity in the past, present, or future.
- The specific date and time that a fact was created and/or updated in the database.
- The state of the business data represented in the database at any point in time.
- A history of changes to a specific (data) fact over time.

Depending on business requirements, the data design will need to represent how temporality is to be achieved.

***NOTE:** At this time, the standard for representing temporal data is under review; therefore, any temporal requirements must be discussed with IMB Data Architecture and IMB Database Administration.*

9.0 Defining Class Uniqueness

An important factor in designing a data model is determining uniqueness for each class; a modeling principle used to prevent data redundancy. In a logical data model, this is achieved through defining business keys.

A **business key**, also known as a *natural key*, is used to represent an attribute or set of attributes that uniquely identify a record in the class. Typically, these attributes occur naturally within the dataset and therefore, provide business meaning.

In most cases, a class will contain one business key. The standards outlined in the following section provide guidance for how to implement a business key in the data model. If the business requires additional business keys, the data model may require further elements not covered within this document. Please discuss such requirements with your IMB Data Architect and refer to Section 10.0 – Logical to Physical Transformation.

9.1 Business Key Characteristics

When defining business keys, careful thought and planning should be considered in order to ensure keys are clear, tied to business purpose, and provide uniqueness to the data elements within the class. The following list describes the general standards for business keys within NRS data models.

A business key:

- Is not a surrogate key.
    - *Exceptions:*
        - *An attribute, such as Employee Number, that is derived from a unique system generated number and is specifically given (or inherits) business meaning. Sometimes referred to as a Naturalized Surrogate Key. In this case, an attribute would be added to the class and stereotyped with "Business Key".*
- Is stable over time.
- Must be defined on all classes.
    - *Exceptions:*
        - *Business keys are not explicitly shown on the model for list and code classes.*
        - *Classes that do not have a suitable business key must be discussed with IMB Data Architecture.*
- Is derived from:
    - A single attribute or association, or
    - A combination of attributes and/or associations
- Contains the minimal attributes/associations needed to uniquely identify the record.
- Will be transformed into the physical data model as a:
    - Primary key (with all components mandatory) or,
    - Unique constraint

- *Note: Where a surrogate key is defined as the primary key.*

9.2 Modeling Business Keys

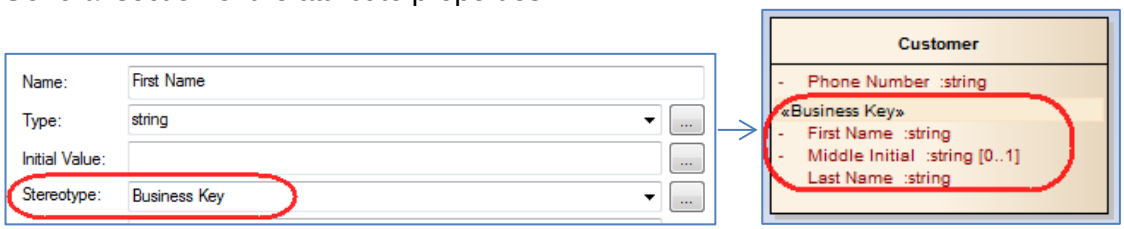
For the purpose of creating a logical data model, the business key is a set of attributes and/or associations that identifies class uniqueness in a way that makes sense to the business.

This section describes how to represent business keys on a logical data model.

9.2.1 Defining Business Key Attributes

For each business key attribute, the following steps are required:

- Each attribute is defined as per Section 7.0 – Attribute Characteristics.
- Each attribute is labelled with the ***Business Key*** stereotype. This is done within the *General* section of the attribute properties.

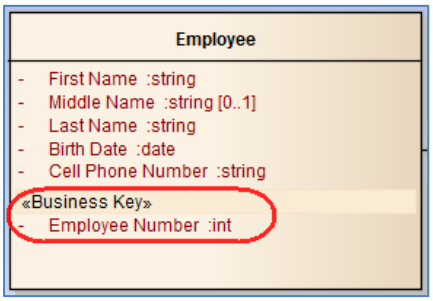


The Business Key stereotype is applied to attributes that form the (Composite) business key

9.2.2 Defining Naturalized Business Keys

Where the business key is intended to be a system-generated number that brings business value, for example; Employee Number, these additional standards are required:

- The key consists of **one** attribute.
- The attribute name will be suffixed with ID or NUMBER (whatever makes more sense to the business requirements).
- Attribute data type is **'int'** and must have a specified length.
- The attribute is **mandatory**.



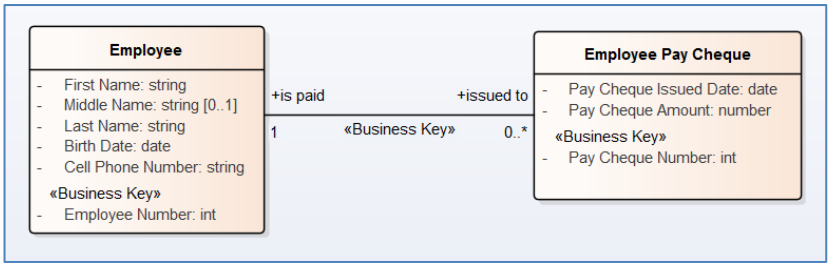
Example of a naturalized surrogate key used to define a business key for a class

9.2.3 Defining Business Key Using Associations

A business key may be comprised, partially or solely, from another class' business key. Where this occurs, a business key attribute is implied through the association rather than a specific attribute within the class. It is important to understand that a business key may include a combination of association(s) and attribute(s).

Where a business key is comprised of an association, the following applies:

- The association between the parent and child class is categorized using a <<*Business Key*>> stereotype. This is done within the *General* section of the Association Properties. Refer to section 5.3 – NRS Association Stereotypes for more details.



This example implies that the Employee Number from Employee class will become a part of the business key within the Employee Pay Cheque class. The identifying association shows the Employee Number is required to make the Employee Pay Cheque business key unique.

10.0 Logical to Physical Transformation Guidelines

In order to transform the logical data model through to physical database implementation, a number of steps are required. NRS has created a document that supports the requirements necessary to complete the transformation.

The ***NRS Logical to Physical Data Model Transformation*** document describes the process of taking a logical data model in the EA tool, transforming it to a physical database model, and finally generating the Data Definition Language (DDL) scripts used to create the database objects.

Note that in conjunction with the transformation document, references should be made to the ***NRS Physical Data Modeling Standards***.

11.0 Summary

This document has provided a comprehensive description of the Sector standard for data modeling using the EA tool and UML notation. As the Sector works through refinement of standards, this document may be updated to reflect any changes.

Inquiries regarding these standards may be forwarded to the IMB Data Architecture section mailbox at CSNR.Data.Architecture.Services@gov.bc.ca.