Ontario

Government of Ontario

Information & Technology Standards

Government of Ontario IT Standard (GO-ITS)

Number 56.3

Information Modeling Handbook (IMH) – Appendices

Version #: 1.5

Status: Approved

Prepared under the delegated authority of the Management Board of Cabinet

Foreword

In 2002 the Information Architecture Domain Working Group (IADWG) recognized the need to have a consistent approach for modeling information within the Ontario Government. A consistent modeling approach facilitates opportunities for integration, reuse and data sharing, and extends knowledge sharing between the business and the IT communities.

With the endorsements of the Architecture Core Team and Information Technology Standard Council and the approval of Architecture Review Board, IADWG published the Information Modeling Handbook (IMH) as a Government of Ontario Information Technology Standard in August 2007.

The IMH provides standards, guidelines and best practices for information modeling and reflects common industry standards and recommended practices across the OPS.

All versions remain methodology and tool independent.

As you use this document your feedback on any of the sections is welcome. Please contact your Cluster Information Architect or any of the IADWG members listed in the Document Control section.

Related Documents

Document Title
Architecture Review Guidance for the Acquisition & Integration of Acquired Solutions, http://intra.net.gov.on.ca/iit/services/enterprise-architecture/domains/#application
Enterprise Architecture Process and Methods Handbook, http://intra.net.gov.on.ca/iit/services/enterprise-architecture/eapm-handbook/
GO-ITS 56 - OPS Enterprise Architecture: Principles and Artefacts, Appendix B – "Corporate Enterprise Architecture Review Requirements Guidebook", http://www.gov.on.ca/MGS/en/IAndIT/STEL02_047303.html
Enterprise Architecture Glossary, http://intra.net.gov.on.ca/iit/services/enterprise-architecture/eapm-handbook/

Document Control

The following IADWG members contributed to the current release of the IMH:

Name	**Title**	Organization
Anna Nadin	Database Technician	Justice Technology Services Anna.Nadin@ontario.ca
Arash Zaryoun	Information Architect	Government Services I&IT Cluster Arash.Zaryoun@ontario.ca
Claude Sam-Foh	Enterprise Architect	Government Services I&IT Cluster Claude.Sam-foh@ontario.ca
David Downs	Information Architect	IT Source David.Downs@ontario.ca
Gennaro Giampaolo	Information Architect	Enterprise Services I&IT Cluster Gennaro.Giampaolo@ontario.ca
Joanne Venema	Information Architect	Health Services I&IT Cluster Joanne.Venema@ontario.ca
Lien Truong	Information Architect	Children, Youth & Social Services I&IT Cluster Lien.Truong@ontario.ca
Lorie Oblak	Information Architect	Labour & Transportation I&IT Cluster Lorie.Oblak@ontario.ca
Richard Pelletier	Information Architect	Central Agencies I&IT Cluster Richard.Pelletier@ontario.ca
Shahid Sheikh	Sr. Information Architect	Land & Resources I&IT Cluster Shahid.Sheikh@ontario.ca
Suzanne Bond	Information Architect	Community Services I&IT Cluster Suzanne.Bond@ontario.ca
Thomas Chen	Information Architect	I&IT Innovation, Controllership and Strategy Division, MGS Thomas.Chen@ontario.ca

The following people contributed to previous releases of the IMH:

Name	Role	Release Involved
Alana Boltwood	Contributing Author & Reviewer	Releases 1, 2, 3, 4
Daniel Chang	Reviewer	Release 3
Ellen Chen	Contributing Author & Reviewer	Release 4
Eugenia Popescu	Reviewer	Release 1.4
Frank Cheng	Contributing Author & Reviewer	Release 1, 2, 3, 4, 1.4
Fred Woodhall	Contributing Author	Release 4
Garry Stoddart	Reviewer	Releases 1, 2, 3
George Berelidze	Contributing Author & Reviewer	Releases 1, 2, 3
Jane Liang	Contributing Author	Releases 2, 3
Kamel Toubache	Contributing Author & Reviewer	Releases 1, 2, 3, 4
Karin Wood	Principal Author	Releases 1, 2
Kathleen Youmans	Contributing Author & Reviewer	Releases 1, 2, 3
Larry Zehnle	Contributor	Release 4
Les Piotrowski	Contributing Author & Review	Release 1.4
Moira Watson-Turner	Reviewer	Releases 1, 2, 3
Norman Lee	Reviewer	Releases 1, 2, 3
Sonia Gluppe	Editor	Releases 2, 3

Table of Contents

1. Introduction

1.1 Purpose of the Handbook

The Information Modeling Handbook (IMH) provides standards, guidelines and best practices for information modeling. Guidelines and best practices should be followed and standards must be applied in order to produce high quality models.

The Ontario Government has adopted the Enterprise Architecture as a way to document and reuse knowledge. IT projects incorporate business perspectives (knowledge) as they develop applications to meet the business requirements.

The Information Architecture Domain Working Group (IADWG) recognized the need to have a consistent approach to modeling information within the Ontario Government. A consistent modeling approach will support opportunities for integration, reuse, data sharing, and extend knowledge sharing between the business and the IT communities.

1.2 Target Audience

The primary audience of this document is Data/Information Architects and Modelers, Database Administrators and anyone performing those roles. The secondary audience includes those performing the roles of Business Analysts, Data Stewards, Data Custodians, Quality Assurance and Project Managers.

1.3 Scope

Activities of people in the IT community can typically be classified into 5 domain groups: business, application, information, technology and security. The IMH focuses mainly on the information domain, at operational and tactical levels, as indicated by the dotted areas in Figure 1-1.

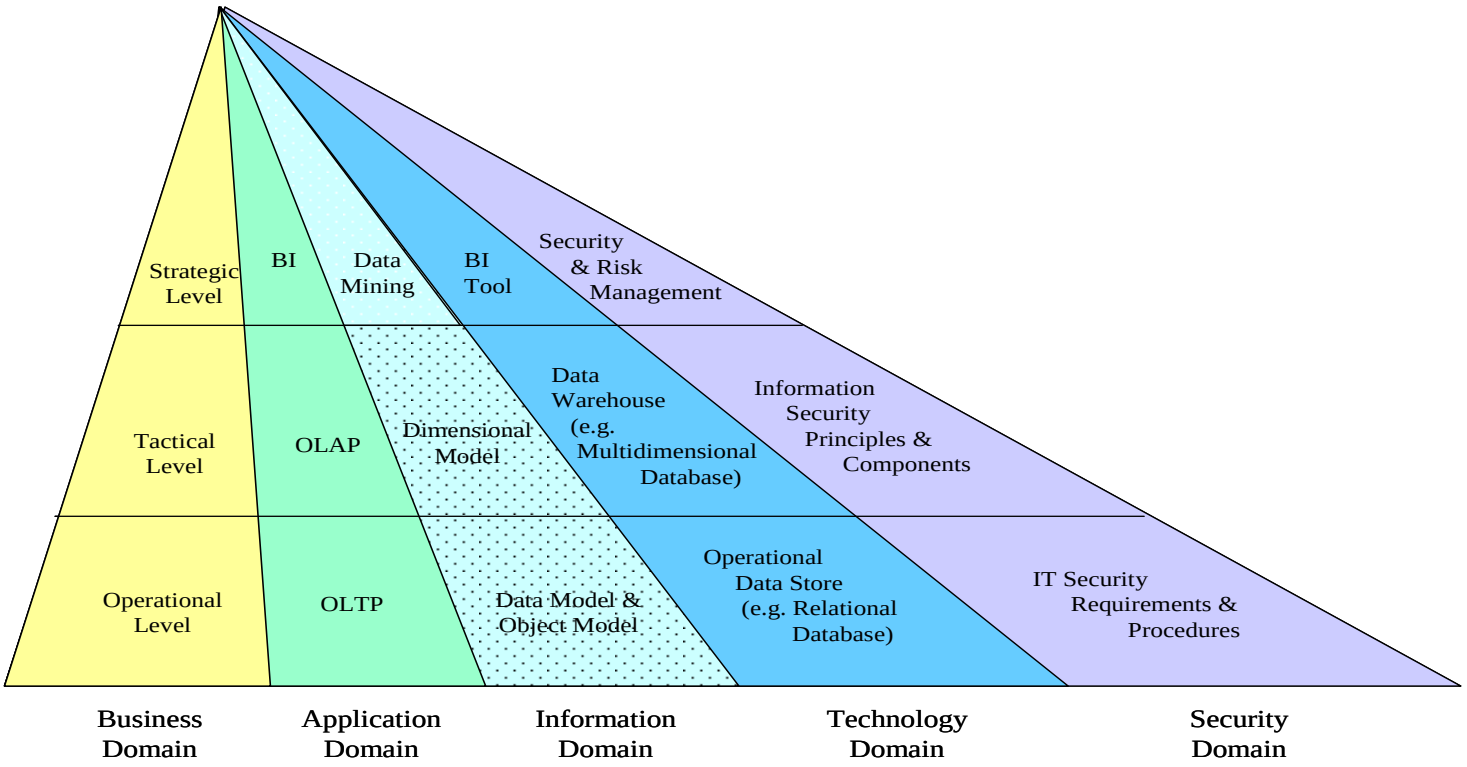


Figure 1-1: Information System Pyramid

What is covered:

- Components of the different data model types (conceptual, logical, physical, dimensional models and XML).
- Data naming standards.
- Data modeling notations.
- Guidelines for information models.

What is not covered:

- Aspects of data management other than information modeling.
- Procedures for managing models within modeling tools.
- Information modeling tools.
- Educational aspects of information and data modeling.
- Data model QA checklists.

The IMH is primarily focused on describing types of models within the information domain and the recommended techniques and standards used to define and produce models.

Figure 1-2 shows the model types described in the IMH, and various contexts within which the modeler may be working. For example, when a project team produces a data model, the data model:

- Should leverage existing enterprise data models, by using them as a starting point.
- Must be aligned with existing enterprise data models to ensure that the specifics of the application fit within the “bigger picture”.
- Must be modeled at a level of abstraction and/or level of operational detail suitable to the type of project.
- Must be mapped against process requirements and further refined for completeness.
- Must be managed and further detailed throughout the transformation stages.

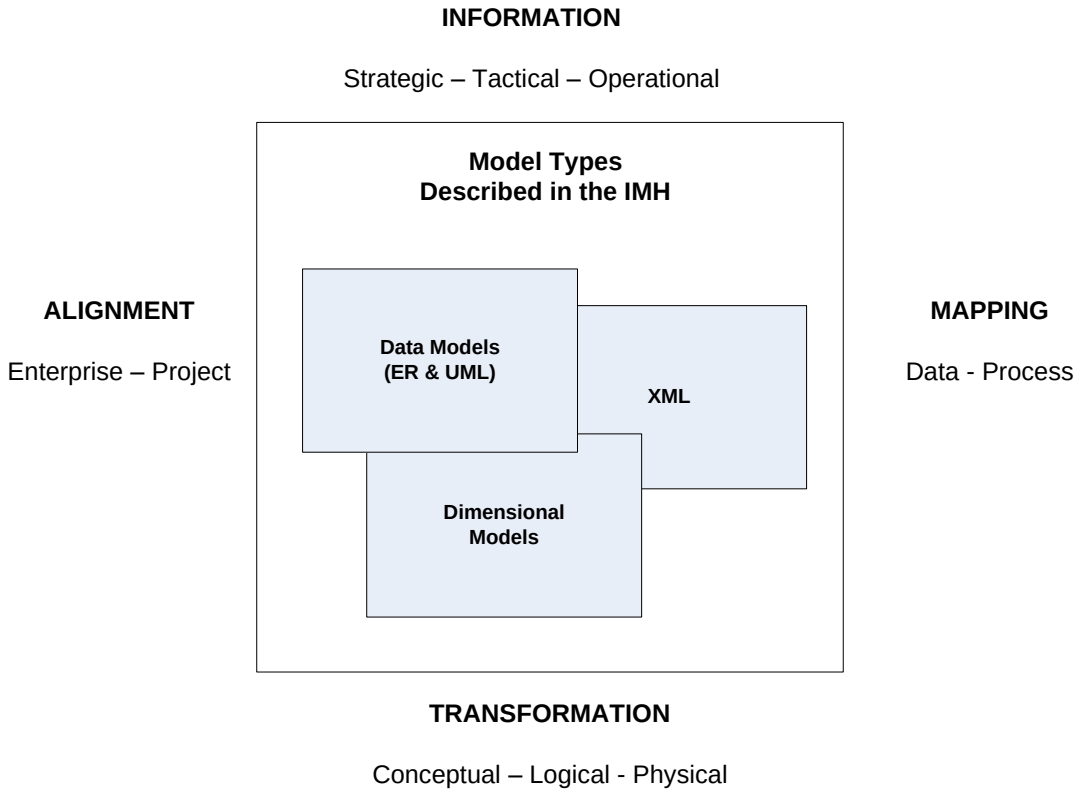


Figure 1-2: Information Modeling Scope / Context

1.4 Purpose of Information Modeling

Information is a valuable resource when it becomes the basis to achieve accurate communication and to develop databases that support business operations. Information modeling rigorously defines data objects, facts, and rules that help the organization achieve this understanding and therefore do business effectively.

The purpose of information modeling is to:

- **Define business data needs:** A thorough and rigorous analysis of business data yields a solid understanding of the business. The primary reason for this is that although business process steps may change, business data needs remain largely stable. Information modeling uses diagrams and sentences to capture business knowledge in the language of the business.
- **Define data independent of technology:** Information modeling defines the information of interest to an organization in a form that is independent of the technologies that are used to store, process and transmit information.
- **Support database design:** Information modeling leads to defining the characteristics of any physical database.
- **Provide the basis for data sharing and integration:** The better the data is understood the easier it is to share. Information modeling serves to inventory the data resources (assets) of the organization.

Information modeling contributes to improving application systems development productivity. Models can be shared between systems development teams thereby reducing the time and effort needed to analyze the organization's data requirements. Furthermore, data modeling supports automation by making it possible for modeling tools to generate data definition language (DDL) scripts.

1.5 Information Domain Overview

Figure 1-3 depicts the current related initiatives and their relationship to the information domain. Data / Information Architects and Modelers, and anyone performing those roles, should be aware of efforts within the Ontario Government in aligning business and IT.

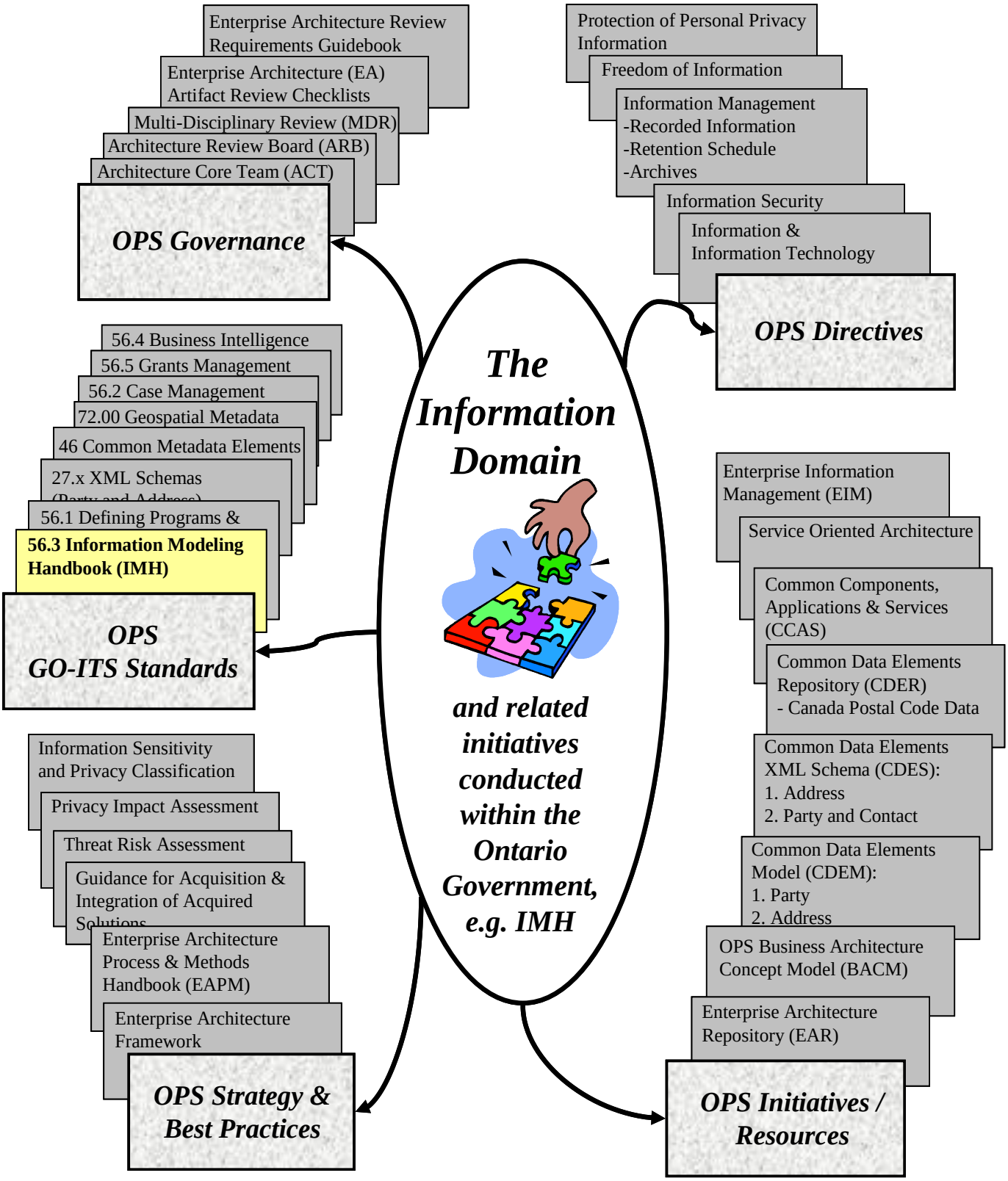


Figure 1-3: Information Domain Overview

2. Components of a Data Model

Data modeling involves identifying things (entities) of importance for an organization, the properties (attributes) of those things and how the things are related to each other (relationships).

A data model can be used to model the data or information requirements for Information Management Systems (MIS, CRM), Online Transaction Processing (OLTP) systems, Online Analytical Process (OLAP) systems (i.e. Data Warehouses and/or Data Marts), XML Schemas, and for general business understanding.

A data model includes the following elements:

- **Diagram** - A graphical representation showing entities, attributes that further describe the entities, and the relationships between the entities.
- **Entity** - An object about which the business collects data. It is a class of uniquely identifiable persons, places, things, events or concepts of interest to the business.
- **Attribute** - A property or characteristic that describes an entity. The attributes of an entity represent the information kept about the entity, which is relevant to the business operations and business functions.
- **Relationship** - An association that exists between two entities, based on business policy. A relationship represents a business rule.
- **Data Dictionary** - A centralized collection of metadata about a data model. It includes data definitions and characteristics of all the data model elements. It also includes cross-reference lookups and usage rules.

2.1 Levels of Data Models

In terms of the Enterprise Architecture Framework, Table 2-1 outlines the different data model types, where each type of data model represents a different level of abstraction.

EA Level	Data Model Type	Data Model Scope	Entity Level	Relationship Level	Attribute Level
Scope (Contextual)	Business Resource Types	Enterprise or Project (One list of Resource Types per scope)	Business definitions	N/A	N/A
Business Model (Conceptual)	Conceptual Data Model (CDM)	Enterprise or Project (Single CDM per scope)	Business Data Entities: key entities identified, named and described	Business Data Relationships: identified, named and described	Business Data Attributes: representative attributes named, and described
System Model (Logical)	Logical Data Model (LDM)	Enterprise or Project (Single LDM per scope)	Data Entities: normalized and fully defined with properties	Data Relationships: final and fully defined with properties	Data Attributes: all attributes identified, final and fully defined with properties
Technology Model (Physical)	Physical Data Model (PDM)	Enterprise or Project (May consist of multiple PDMs per scope)	Tables or XML elements: final and fully defined with properties	Keys: final and fully defined with properties	Columns or XML elements: final and fully defined with properties
Detailed Representation (Out of Context)	Data Definition (DDL)	N/A	N/A	N/A	N/A
	Physical File Definition	N/A	N/A	N/A	N/A
	XML Schema Definition	N/A	N/A	N/A	N/A
Functioning Enterprise	Data (Database)	N/A	N/A	N/A	N/A
	Messages / Documents	N/A	N/A	N/A	N/A

Table: 2-1: Types of Data Models

Note: The focus of the IMH is on the conceptual, logical and physical data models.

2.1.1 Conceptual Data Model (CDM)

A conceptual data model represents in scope business entities and their relationships.

The conceptual data model is a formal representation of the data needed to support business processes and functions delivered by an enterprise or a business domain. It is the precursor to the logical data model. It focuses on entities that have a business meaning, the important relationships among these entities and the representative attributes of the entities.

The conceptual data model should be built using existing template models such as the OPS Business Architecture Concept Model (BACM) as the starting point. This will help to facilitate data alignment across the OPS.

Format

The conceptual data model includes one or more diagram and supporting metadata of all the mandatory elements, and can be diagrammed using one of the following notations:

- Entity Relationship diagram notation, or
- Unified Modeling Language (UML) class diagram notation representing only the entity classes without showing any methods on these classes.

Impact of Not Developing This Model

The conceptual data model demonstrates a common understanding of in scope business entities, concepts and their relationships. Should this model not be prepared, any combination of the following may occur:

- Lack of scope definition which may jeopardize requirements definition, data analysis and design effort, project estimation activities.
- May miss important information needs and their implications.
- May not identify some key functional requirements related to the missing subject area groupings.
- Inability to assess information sharing requirements across business units or functional areas.
- Lack of understanding of the common definitions, semantics, information, and knowledge across all business domains within an organization.
- Inability to partition the organization’s information and scoping subsequent projects.
- Results in an incomplete picture of the needs of an organization, or a project which may result in erroneous recommendations regarding the development of some solution areas.
- Lack of foundation to develop a coherent database strategy.

Audience

The primary audience for the conceptual data model is the business staff who will validate the model to ensure that it meets their high-level business requirements. The secondary audience is IT staff who need to review and understand the business requirements.

Scope

Enterprise

An enterprise CDM is a conceptual enterprise-wide view of data and their relationships. It normally includes an overview of one or many subject areas (high-level, generic classifications or groupings of entities that share a common business functions) and the relationships between them.

The enterprise CDM defines the common terms and strategic business rules for business entities without technology constraints. The scope of the enterprise CDM should cross the functional and organizational boundaries. For the purposes of this definition, the enterprise can be the entire OPS, a cluster (e.g. ETC), a ministry (e.g. MTO), a ministry division (e.g. MTO Road User Safety), a program or initiative that crosses organizational boundaries (e.g. Water Management).

The objectives of an enterprise CDM are:

- **To provide an enterprise perspective.** One purpose of the enterprise CDM is to discover common threads and develop a cross-functional, common definition of the business entities. The definition of information in common enterprise terms improves the context of information. Establishing common definitions will be useful to the enterprise even if no systems or schemas need to be developed (modeling may stop at the business concept level).
- **To define strategic information needs**. The enterprise CDM is a “to be” model, based on known (current or planned) business policies and strategies. It is strategic and recasts data into a model with vision that transcends biases in today’s data. Because the enterprise CDM is at a conceptual business level, its development explores the business entities and business rules most important to corporate survival and success. It is developed with key decision makers and its purpose is to represent information in a manner that will help the enterprise grow.
- **To form the foundation for developing project level information model.** The enterprise CDM forms a foundation and can be used as a reference template model in developing a project level information model within the enterprise.
- **To form the foundation for developing the enterprise’s systems and schemas**. The goal of the enterprise CDM is to build a foundation for information that is a shareable, consistent, reusable atomic source of data for OLTP systems, OLAP systems (i.e. data warehouse and data marts) and XML data exchange.
- **To initiate business data stewardship**. The purpose of data stewardship is to place the accountability, control, shareability, and quality management of enterprise data in the hands of the business people who define, create and access the data. The subject matter experts that contribute to the enterprise CDM serve as strategic data stewards.

Project

The project CDM represents the primary in scope business entities and relationships for a specific project. It zooms in on the area of the organization or the business domain that is the subject of planning and analysis for the project and provides a high-level view representing the business under study for that organization or business domain.

The project CDM usually covers one or more subject areas from the enterprise CDM. It may instantiate, refine and further elaborate the business concepts and information identified and articulated in the enterprise CDM, and provides a project level view of the business concept and related information.

Mandatory Elements

- All fundamental in scope high-level business entities named and described.
- Representative attributes that support the meaning of the entity (e.g. Given Name, Street Name) named and described.
- Preliminary relationships named and cardinality/optionality (or their equivalent term in UML, multiplicity) specified.

  Note: Many-to-many cardinality relationships are allowed in the CDM.
- Make use of template models, where they exist.
- Diagram must include all defined in scope entities, relationships, and entity representative attributes, with the names of entities, relationships and entity representative attributes clearly shown.

Optional Elements

- Business identifiers.

  Note: Include the business identifier if it is recognized by business staff and is important to their understanding of the entity's definition. In a CDM, identifying attributes and relationships need not be shown, but they may be included for the purposes of providing clarity. A conceptual identifier such as "Person Name" may not be unique in all cases. The conceptual identifier may or may not be used as the business identifier in the CDM and the subsequent LDM.
- Non-representative attributes and their descriptions.
- Associative entities if they are fundamental to the business.
- Attributive entities (like lookup entities) only if required for clarity.
- Attribute multiplicity (i.e. the combination of attribute optionality and cardinality), data type, and length.

Example

The following is an example of a conceptual data model.

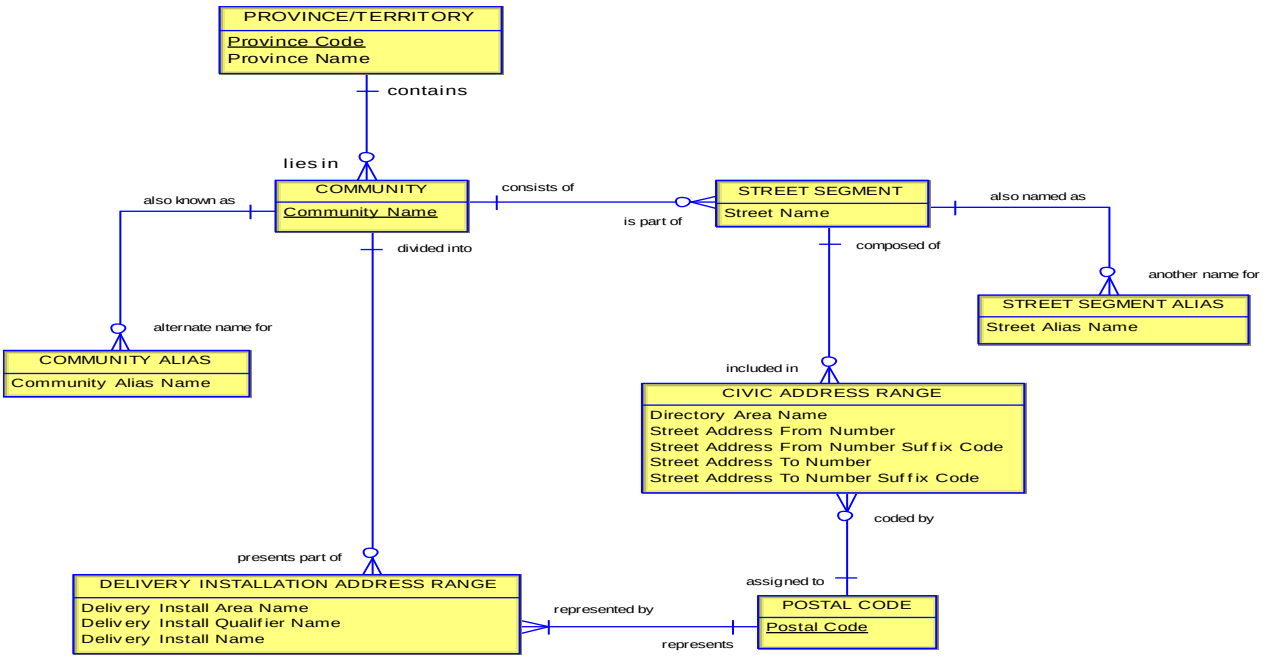


Figure 2-1: Sample Conceptual Data Model Diagram

2.1.2 Conceptual Data Model for Acquired Solution

A Conceptual Data Model (CDM) for Acquired Solution represents all the detailed business data elements needed to support the business processes and functional requirements within the scope of a specific project or initiative.

The CDM for Acquired Solution is a fully attributed CDM that has been extended to include additional details about the business data requirements. It includes specification about both the data content and structural requirements at a sufficient level of detail to inform the Request for Proposal (RFP) for acquiring a desired solution product from a vendor.

The CDM for Acquired Solution is not intended to document proprietary aspects of a solution. The detailed data content and structural requirements specifications included in the CDM for Acquired Solution will help the vendor understand what the key business data requirements are and will help the project understand, during product evaluation, how well the vendor's product meets the business requirements.

The CDM for Acquired Solution should be built using existing template models such as the OPS Business Architecture Concept Model (BACM) as the starting point. This will help to facilitate data alignment across the OPS.

Format

The CDM for Acquired Solution includes one or more diagram and supporting metadata of all the mandatory elements, and can be diagrammed using one of the following notations:

- Entity Relationship diagram notation, or
- Unified Modeling Language (UML) class diagram notation representing only the entity classes without showing any methods on these classes.

Impact of Not Developing This Model

The CDM for Acquired Solution represents the key information requirements that an acquired solution must be able to accommodate. The detailed data content and structural requirements specifications included in the CDM for Acquired Solution will help the vendor understand what the key business data requirements are and will help the project understand, during product evaluation, how well the vendor's product meets the business requirements. Should this model not be prepared, the following may occur:

- The selected vendor product may not be able to accommodate the information requirements of the business resulting in the requirement for additional customization effort.

Audience

The primary audience for the CDM for Acquired Solution is the vendor who is submitting a response to an RFP and the project team reviewing the vendor solution to ensure it is capable of meeting the needs of the business.

Mandatory Elements

- All fundamental in scope business data entities named and described.
- All business relevant data attributes named and described.
- Data types and sizes specified for all data attributes that are required for:
    - Data to be persisted in the database that requires precision and accuracy (i.e. minimum number of digits before and after the decision point in a number, the minimum number of characters in a text field, and/or special data value set);
    - data to be migrated;
    - data required to interface with other applications.
- All business relevant unique identifiers identified and optionality specified.
- All relationships named and with cardinality/optionality (or their equivalent term in UML, multiplicity) clearly specified.
- Many-to-many relationships representing additional business data requirements resolved to show these data requirements.
- Domains of data attributes that are of significant interest to the business defined or at least described in the description of the attributes.
- Data steward and data custodian specified for entities that are of significant interest to the business.
- Data source specified for entities that are of significant interest to the business.
- Information sensitivity and privacy classification (ISPC) specified for entities that are of significant interest to the business.
- Volume estimates specified for entities that are of significant interest to the business.
- Volatility requirements specified for entities that are of significant interest to the business.
- Data retention requirements specified for entities that are of significant interest to the business.
- Make use of template models, where they exist.

- Diagram must include all defined in scope entities, relationships, and entity attributes, with the names of entities, relationships and entity attributes clearly shown, and the optionality of each entity attribute indicated.
- Must show model alignment with existing OPS reference data models.

Optional Elements

- Attributive entities.
- Resolution of many-to-many relationships for technical implementation purposes.
- Data model normalization to 3rd normal form.
- Definition of constraints related to implementation such as domains, referential integrity, etc.

Example

The following is an example of a Conceptual Data Model for Acquired Solution.

Figure 2-2 shows the detailed business data requirement, while Figure 2-3 shows how it aligns with the CDEM Party model.

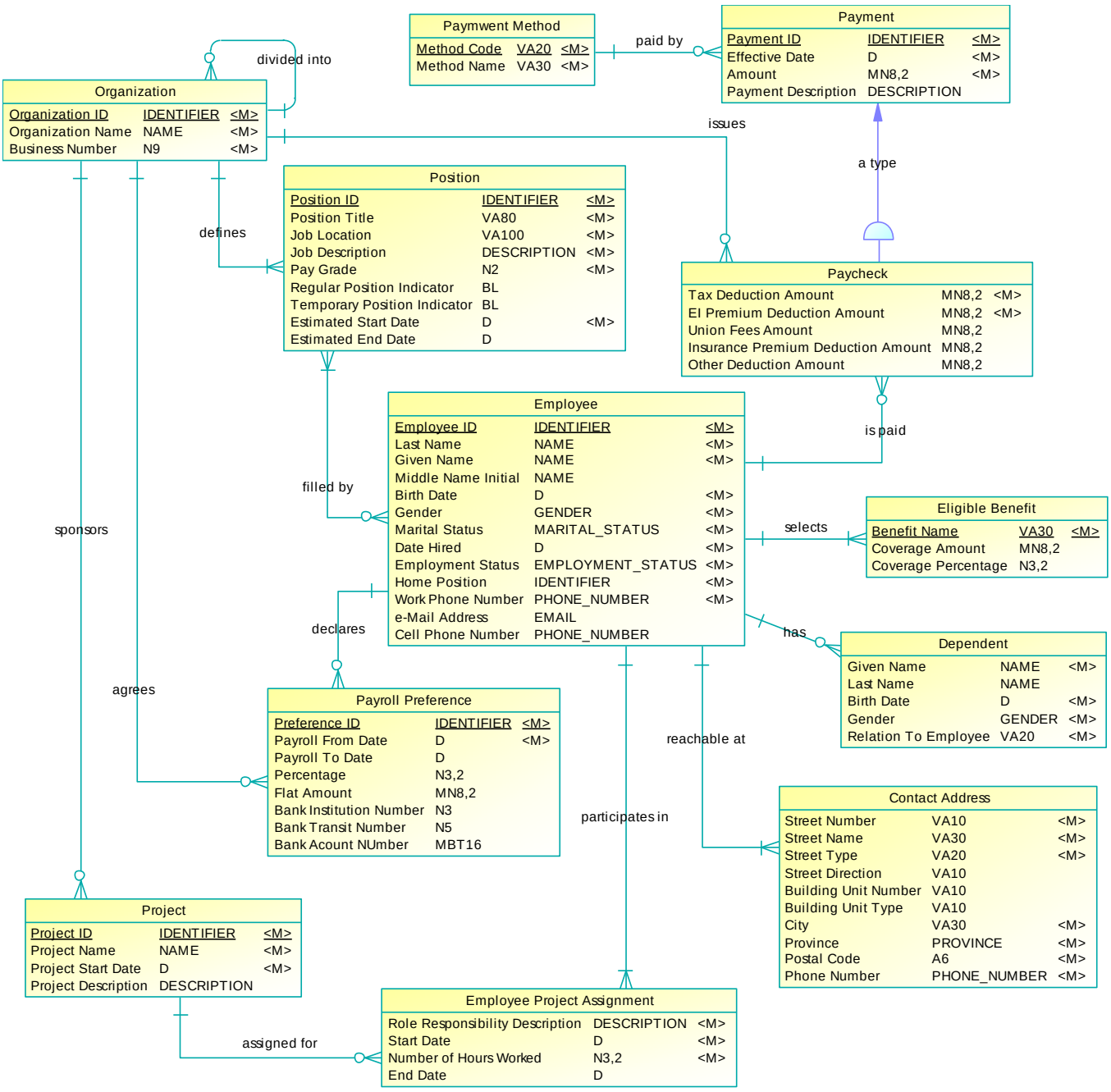


Figure 2-2: Sample CDM for Acquired Solution Diagram Showing Detailed Business Data Requirements

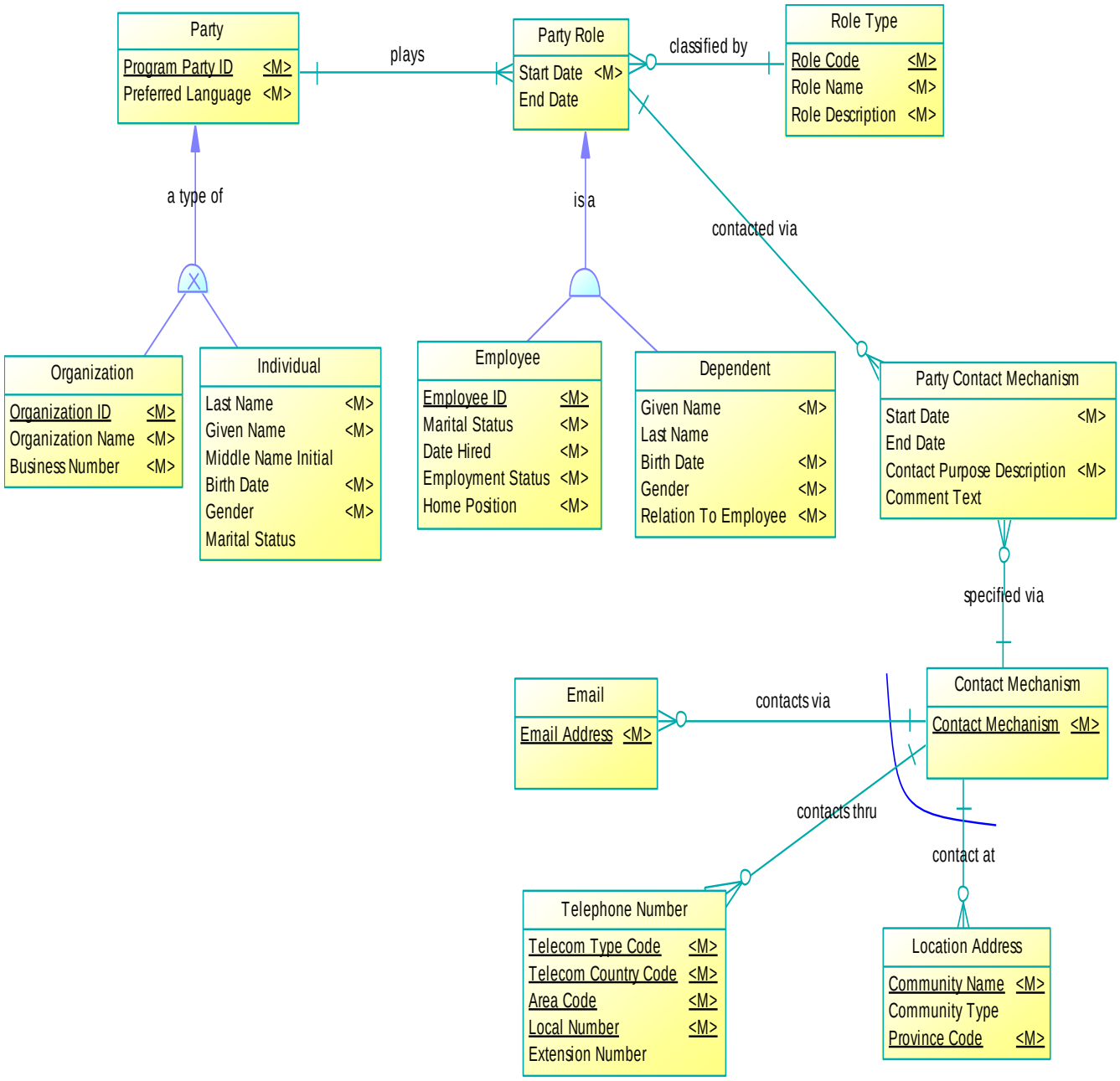


Figure 2-3: Sample CDM for Acquired Solution Diagram Showing Alignment with CDEM Party

2.1.3 Logical Data Model (LDM)

A logical data model represents in scope business entities, their relationships, and their attributes.

The logical data model describes the data requirements and needs in support of the in scope business activities in as much detail as possible without any regard to the physical implementation environment or to performance considerations.

The logical data model can be used:

- To enhance communication between IT and business in the enterprise or business domain.
- To discover, uncover, and clarify business rules involving the business information.
- To understand all the required business information and data.
- As a common reference to describe how business activities (functions) in the scope to produce their respective outcome by manipulating data (CRUD) and exchanging messages (flows).
- To provide the underlying structure of a physical data model (PDM).

The logical data model should be directly traceable to the CDM.

Format

The logical data model includes one or more diagram and supporting metadata of all the mandatory elements, and can be diagrammed using one of the following notations:

- Entity Relationship diagram notation, or
- Unified Modeling Language (UML) class diagram notation representing only entity classes without showing any methods on these classes.

Impact of Not Developing This Model

Without the logical data model, the stored business information is captured solely by a functional model that describes where data stores are created and placed to serve the data needs of the functional processes. There is no unified view of all data, and data normalization is not possible. A physical data model would have to be designed from a conceptual data model and process model, which may result in data ambiguities and redundancies. There is no communication document to refer to in the event of turnover in the project team. Key drawbacks expected from not developing this model include:

- There is a danger of missing important information needs and their implications.
- Business users may become confused by technical considerations they do not care or need to be exposed to. This may lead to a disengagement of the business side.
- Business signoff is complicated and hence possibly compromised.

Audience

The primary audiences for the logical data model are the business staff who need to understand, clarify and verify the business information requirements, and the IT staff including architects, application and database designers who need to review and understand the business information requirements.

Scope

Enterprise

The enterprise LDM is a further elaboration of the enterprise CDM. It shows common data structures, data definition, data organization, and data usage patterns that must be in place in order to satisfy business requirements, including privacy requirements, within the scope of the enterprise or a specific domain of the enterprise.

An enterprise LDM can be divided into one or more logical subject areas. This effort will define and validate the logical subject areas for the enterprise, and should include the priorities for completion of the analysis for each logical subject area.

Subject areas are high-level, generic classifications or groupings of entities that share a common business function. These are used to partition the enterprise data model into smaller logical parts. The model should be partitioned in a way that minimizes cross-subject area relationships. As such it is necessary to ensure that entities and relationships are consistently assigned to one subject area.

The enterprise LDM is about business data entities that can collectively describe and support the business function, not every single entity needs be implemented. It is very important that the identified business data entities are evaluated to ensure that they represent business entities as the business perceives them, and are not completely driven by the technology and systems concerns.

Project

The purpose of the project LDM is to show the structure of data, data organization, and data usage patterns that must be in place in order to satisfy business requirements, including privacy requirements, within the scope of the project.

During the data analysis phase, the business information requirements captured in the project CDM are further elaborated and refined into the business data requirements described in the project LDM. It shows how the data is related and explores any data integration requirements with business areas outside the scope of the project. The project LDM is created without any considerations for a specific technology implementation environment, performance optimization, data storage, etc. The model can be divided into subject areas for presentation purposes.

Mandatory Elements

- All fundamental in scope data entities named and defined, including associative and attributive entities.

  Note: While the entities in the LDM should be traceable to the CDM they may not match the entities in the correspondent CDM on a one to one basis, because complex business rules that did not appear in the CDM will appear in the LDM.
- All attributes (data elements) properly named with class words (see Appendix B), described, and defined with data type, length, and optionality specified.
- Domains and validation information specified for attributes, where applicable. All domains must be properly defined with name, description, data types, length, and allowed values or range of values.
- Primary key (surrogate key or candidate key) and any unique identifiers (business identifiers) for each entity.

- All data are in at least full 3rd normal form, with all data dependency clearly identified and data redundancies eliminated.
- All relationships named and defined with multiplicity (or cardinality and optionality combination) clearly specified. All many-to-many cardinality relationships must be resolved using an associative entity.
- Diagram must include all defined in scope entities, relationships, and entity attributes, with the names of entities, relationships and entity attributes clearly shown, and entity identifiers, optionality, data type and size of each entity attribute indicated.
- Data steward and data custodian specified for each entity.
- Data source specified for entities that are of significant interest to the business.
- Information sensitivity and privacy classification (ISPC) specified for each entity.
- Volume estimates specified for entities that are of significant interest to the business.
- Volatility requirements specified for entities that are of significant interest to the business.
- Data retention requirements specified for entities that are of significant interest to the business.
- Make use of template models, where they exist.

Optional Elements

- Data entity supertypes/subtypes may be resolved. This may require the use of a subtype discriminator attribute.
- Specification of data source, volume, volatility and retention requirements for other data entities.
- Specification of information sensitivity and privacy classification for attributes.

Example

The following is an example of a logical data model.

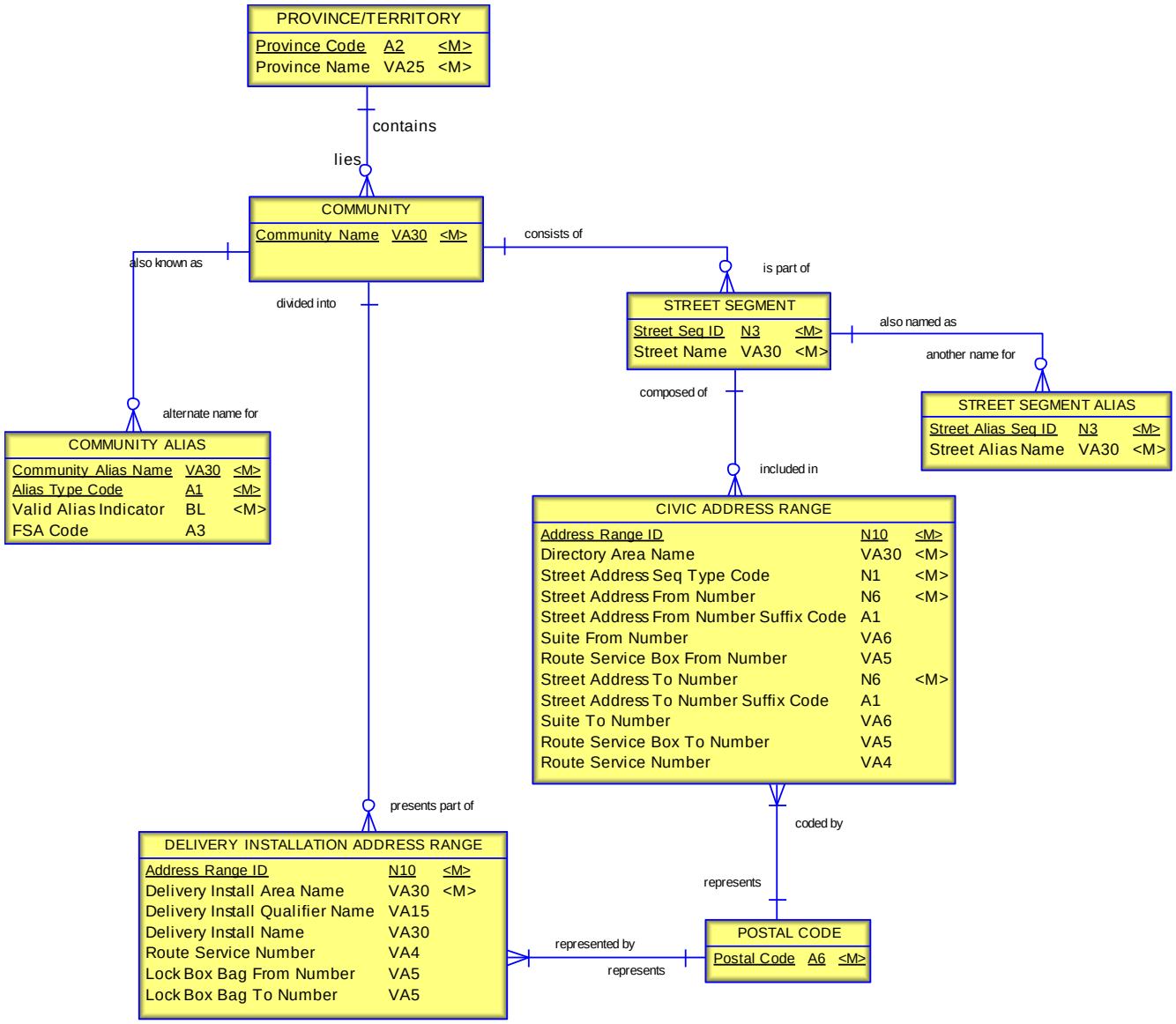


Figure 2-4: Sample Logical Data Model Diagram

2.1.4 Physical Data Model (PDM)

A physical data model defines the physical implementation of the logical data requirements using a particular technology within an intended implementation platform and environment.

The physical data model is primarily concerned with physical limitations, performance and space requirements. It may introduce various new implementation objects, such as database triggers, primary key and foreign key constraints, and check constraints in the database environment to ensure that the business rules in the logical data model are specified during the physical implementation.

The physical data model may also introduce new implementation objects such as indexes that do not contribute to the business information requirements of the system application. These new objects may be created in order to speed up response time, ensure that the application fits within the physical limitations of the computing environment, improve maintenance turnaround, etc.

The physical data model should be traceable to the LDM.

Format

The physical data model includes one or more diagram, supporting metadata of all the mandatory elements, a mapping or design document which provides traceability back to the LDM, and can be diagrammed using one of the following notations:

- Relational diagram, or
- Other formal graphical representations (e.g. tree diagram for XML model).

Impact of Not Developing This Model

Without a physical data model, some special features and characteristics that are available on a specific technology platform may not be fully recognized or utilized; some of the good design considerations and options for efficiency improvement, performance enhancement, and cost reduction may not be fully explored, evaluated, captured and represented. For example, without doing some data de-normalization or creating indexes, the resulting database will be fully normalized with no additional structures to assist performance. If the access patterns interact with multiple structures performance may not meet requirements. The only solution in this situation would be to acquire faster platforms to host the databases. This is not a cost-effective approach given that database performance can be increased substantially by making modifications to the logical database schema.

Audience

The primary audience for the physical data model are IT staff; architects, application designers and developers, database designers, and database administrators. PDMs contain all the information that is needed to implement and maintain the database and XML documents.

Scope

Enterprise

An enterprise PDM is required for the development and implementation of a physical enterprise database that contains functionally independent data, common across the enterprise. For example, the enterprise database may contain a set of common code data look up tables. An enterprise PDM may be comprised of multiple PDMs.

Project

A project PDM specifies the physical implementation of the application database. It considers the details of actual physical implementation and takes into account both software and data storage structures. The model can be modified to suit the performance or physical constraints.

Mandatory Elements

- All tables are named and described, including a short name.
- The primary key (surrogate key or candidate key) and unique keys (unique identifiers) for each table are identified, named and defined.
- All columns in each table, are named using class words, where applicable, and are defined with domains (data type, length, allowed values, validation information) and optionality specified.
- Relating columns for each relationship (i.e. foreign keys) are identified and named to replace relationship name.
- Table constraints (primary key constraint, foreign key constraint, unique key constraints, and check constraints) are identified, named and defined.
- Indexes (primary key index, foreign key index, unique key index, clustering index) are identified, named and defined.
- Database triggers and stored procedures are identified, named, defined and described, as required.
- Database views are identified, named, defined and described.
- Diagram must include all defined tables, table columns, and foreign key referential integrities, with the names of tables, table columns and foreign key referential integrities clearly shown, and table key identifiers, optionality, data type and size of each table column indicated.
- Any particularly complex file relationships and access schemas are described.
- Major decisions about the data structures and the reasons for the decisions are documented; for example, the reasons why a table was de-normalized.
- Guidelines for use of the data model are documented. For example, the locking and release of data items that the application accesses.
- Information sensitivity and privacy classification (ISPC) specified for each table.
- Data steward and data custodian specified for each table.
- Data source specified for each table.
- Volume estimates specified for each table.
- Volatility requirements specified for each table.
- Data retention requirements specified for each table.

Optional Elements

- The organization technique, such as B-tree, or hashed, for databases that allow the specification of the organization of the index tables.
- A synonym (same as table name) for each table.
- Storage definitions and tablespaces identified, named, defined and described, where applicable. Assignment of storage definitions and tablespaces to database objects (tables and indexes).

Example

The following are examples of a physical data model, one for relational database design and two for XML schema design.

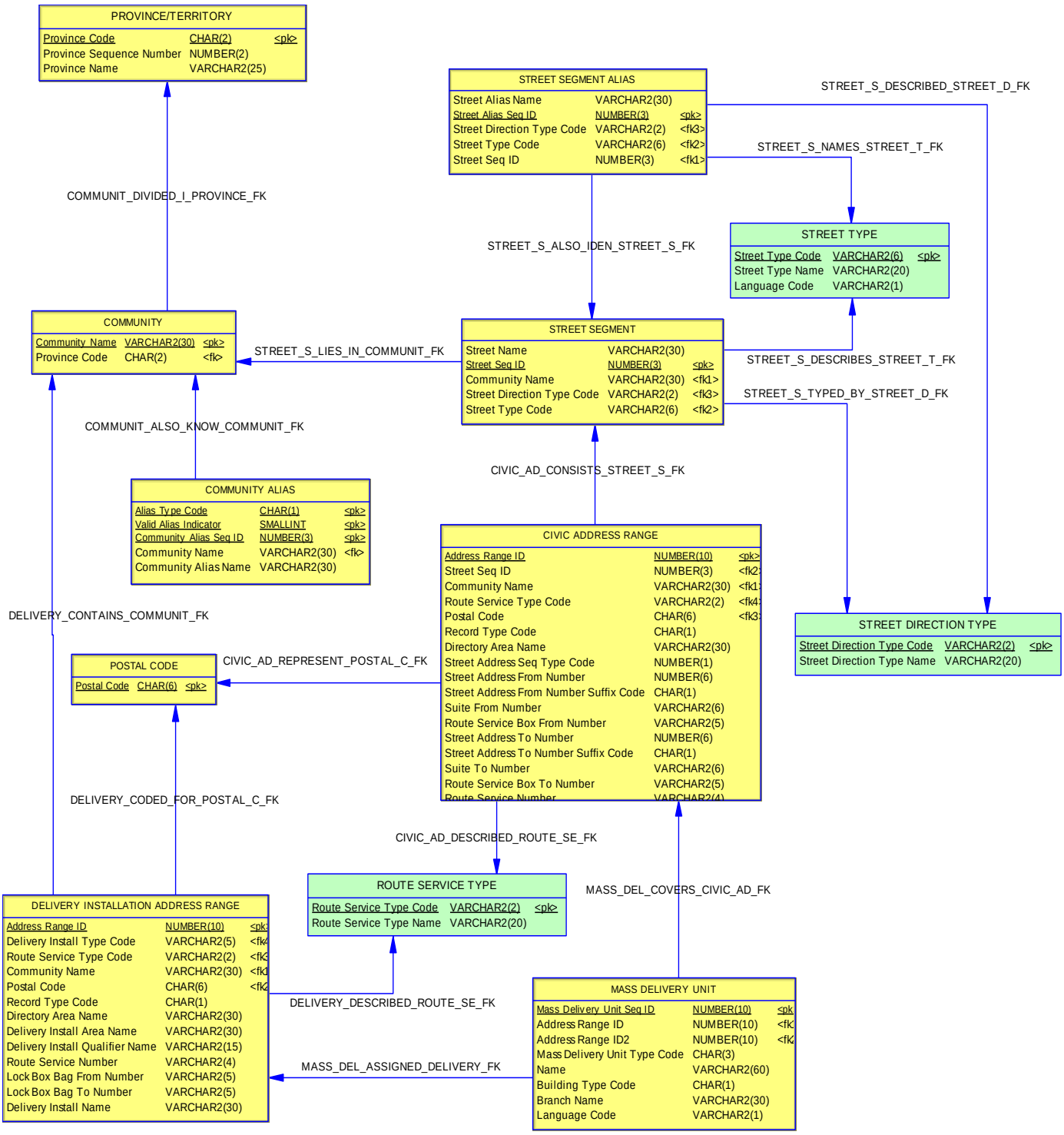


Figure 2-5: Sample Physical Data Model Diagram for a Relational Database Design

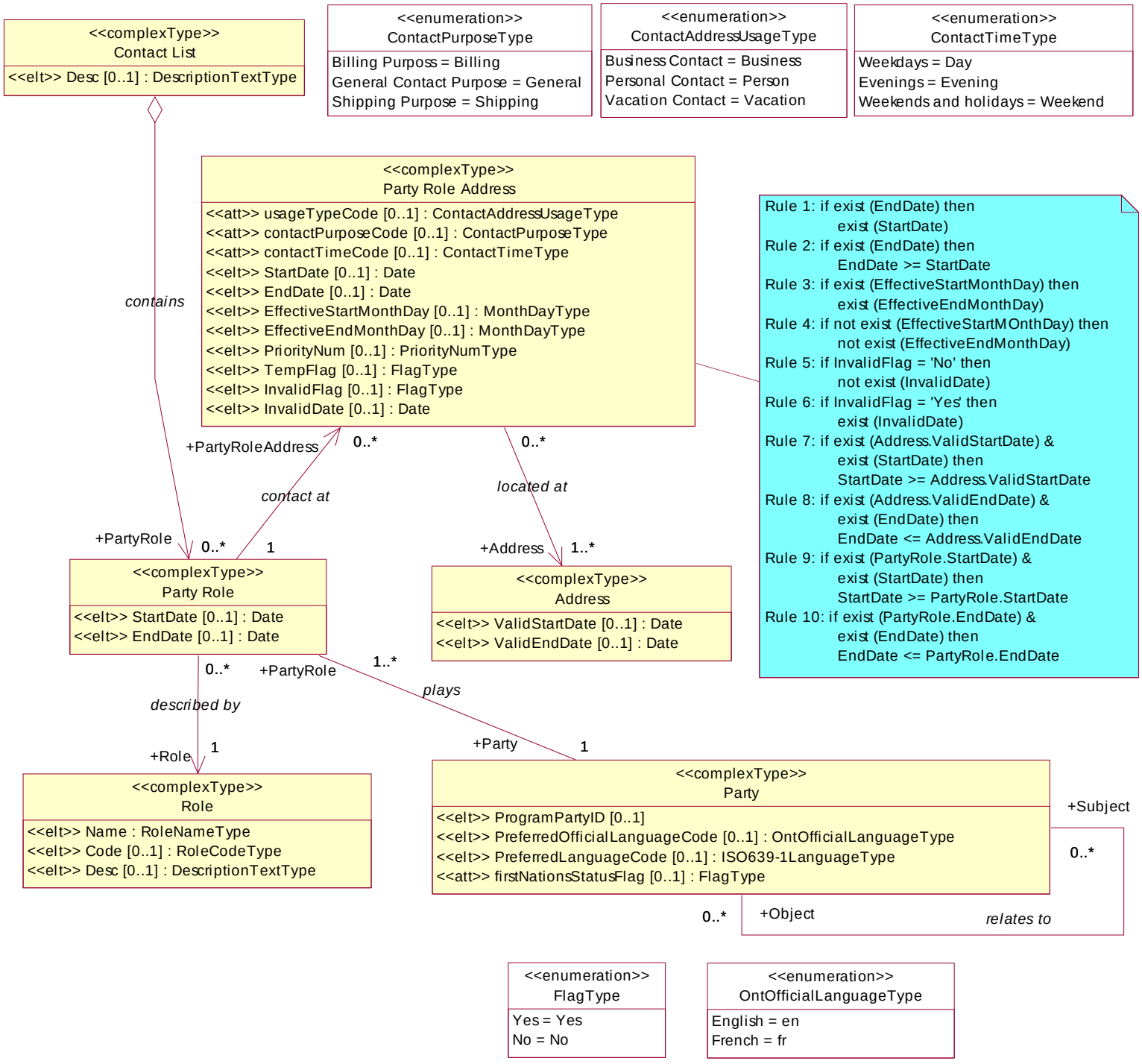


Figure 2-6: Sample Physical Data Model Diagram for an XML Schema Design

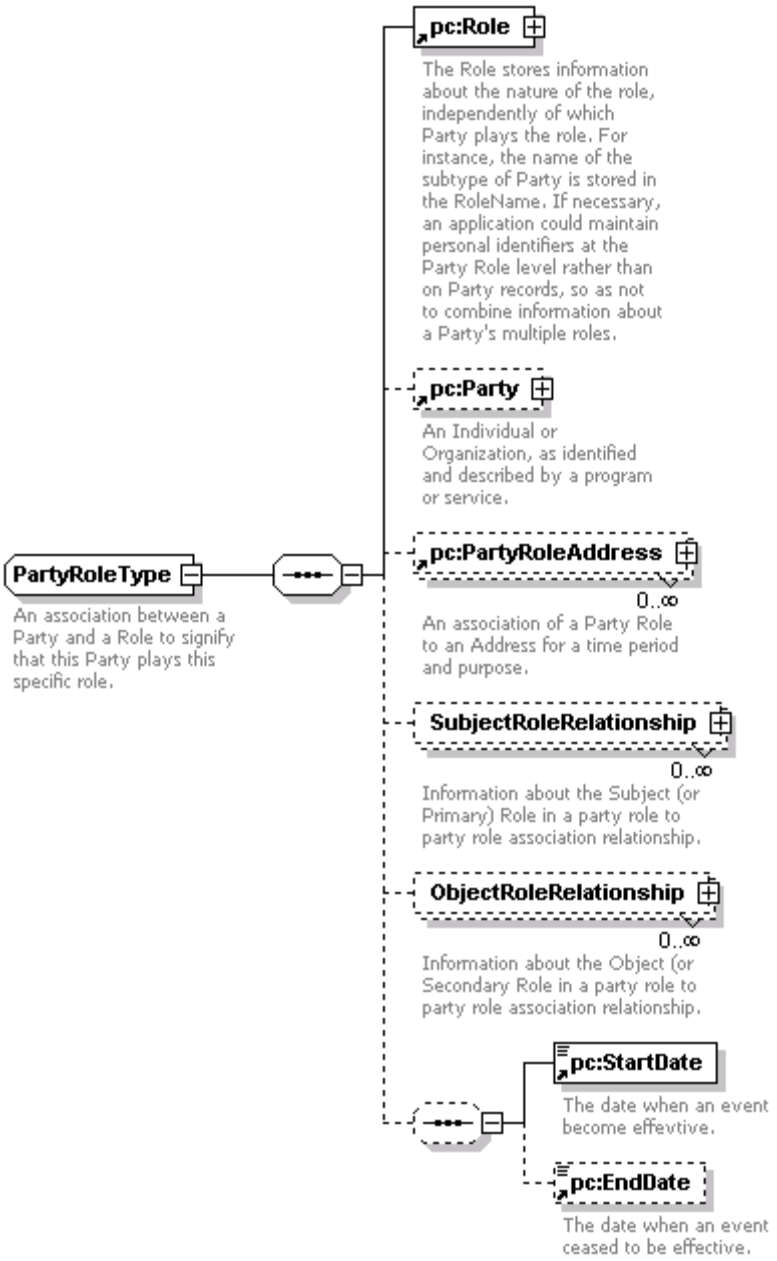


Figure 2-7: Sample Physical Data Model Diagram (in Tree Structure) for an XML Schema Design

2.1.5 Differences between Levels of Data Models

Table 2-2 outlines the major differences between the levels of data models:

Model Element	Conceptual Data Model	Conceptual Data Model for Acquired Solution	Logical Data Model	Physical Data Model
Representative attribute identified	Yes	N/A	N/A	N/A
Entities fully attributed	No	Yes	Yes	Yes
Business unique identifier	No	Yes	Yes	No
3rd Normal Form	No	No	Yes	No
Many-to-Many relationships resolved	No	Yes*	Yes	Yes
Attributive entities identified	No	No	Yes	Yes
Associative entities identified	No	Yes*	Yes	Yes
Constraints (Domains, RI) identified	No	No	Yes	Yes
Primary key (surrogate or candidate key) identified	No	No	Yes	Yes
Supertypes and subtypes resolved	No	No	No	Yes
Table / column structure identified	No	No	No	Yes
Indexes identified	No	No	No	Yes
Non-table objects (views, sequences, procedures, triggers, etc) identified	No	No	No	Yes

Table 2-2: Differences between Levels of Data Models

Note: In Table 2-2 a 'Yes' entry indicates that the model element is mandatory, while a 'No' entry indicates the model element is optional. A 'Yes*' entry means that the model element is mandatory subject to the condition when a Many-to-Many relationship represents additional business data requirements. In this case, this Many-to-Many relationship must be resolved and the additional data requirements must be shown as attributes in an identified associative entity.

2.1.6 Transformation between the Levels of Data Models

Figure 2-8 shows the levels of data models as they relate to the rows of the Enterprise Architecture Framework, and how they are transformed from one level to the next (conceptual to logical to physical), adding more detail and supporting traceability between the levels until the final model is implemented into the target technology. This also illustrates how project

artifacts are built by leveraging existing enterprise artefacts such as the Enterprise Reference Model.

Transformation Between the Levels of Data Models

	Phase	Row	Enterprise Step / Artefact		Project Step / Artefact	Description
Conceptual to Logical	Information Definition	R1	Enterprise Resource Types	Align	Project Resource Types	Common terminology for the information needs of the business
			Transform		Transform	
Conceptual to Logical		R2	Enterprise Conceptual Data Model	Align	Project Conceptual Data Model	Understanding how information is linked and defined (expressed as entities and relationships of the business)
			Transform		Transform	
Logical to Physical	Data Definition	R3	Enterprise Logical Data Model	Align	Project Logical Data Model	Complete and non-ambiguous data definition which fully supports data and process requirements
					Translation to target technology	
	Database / Datastore Design	R4			Project Physical Data Model	Translation of the LDM to the target technology (relational, XML) and the design of the optimized "container" (DBMS, XML Schema)
Physical to Functional	Database / Datastore Development	R5			DDL / XML Schema	Physical "container" (DBMS, XML Schema) to store, and/or transmit, data
	Operational Database / Datastore	R6			Functional Database / Datastore / XML Document	Data is persisted and consumed

Figure 2-8: Transformation between the Levels of Data Models

2.2 Data Modeling Notations

Data requirements are generally expressed in terms of data objects, their properties, and the rules that influence or relate these data objects. A data model is the key means by which data requirements are formally described.

The OPS supports both Entity Relationship (E/R) and Unified Modeling Language (UML) notation to model data in a manner that conforms to the best practices as described in the IMH. These notations differ essentially in the way they represent data requirements, and not in their expressive power. Both notations are briefly described and compared to help understand similarities and differences.

2.2.1 Data Modeling Using E/R Notation

E/R notation expresses business requirements schematically through an ERD (Entity Relationship Diagram) that shows entities and their relationships to other entities. There are many variations in E/R notation (e.g. Chen, IDEF, Barker). A sample ERD developed using the Barker notation is provided in Figure 2-9.

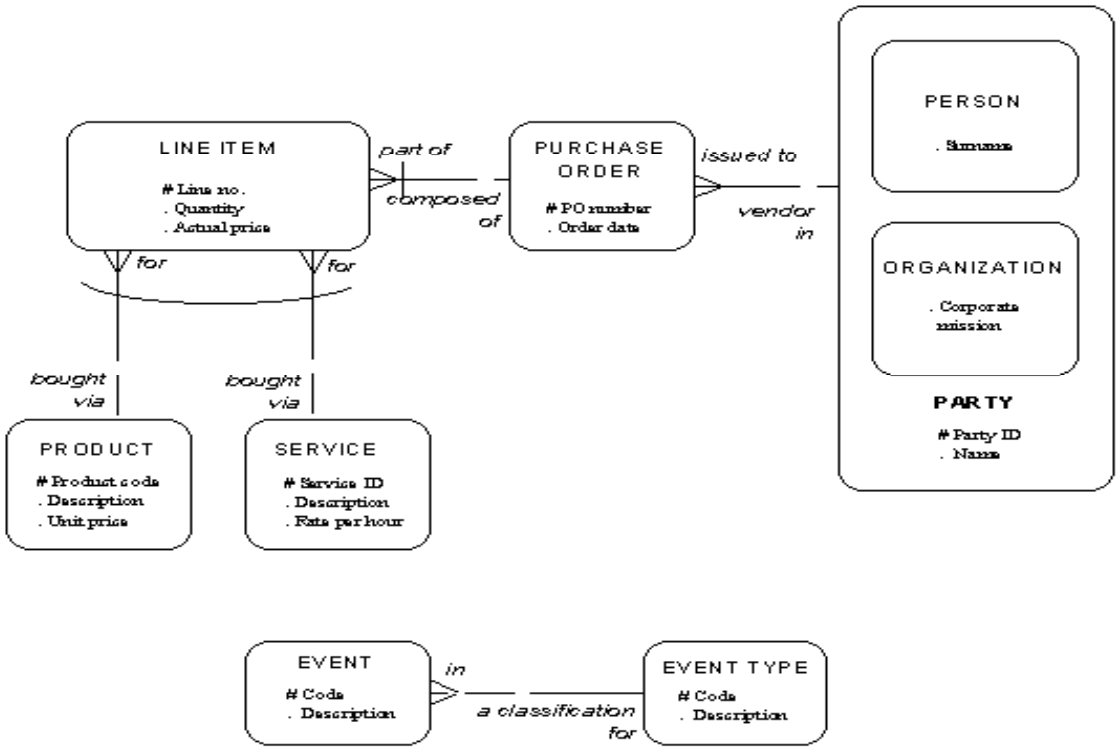


Figure 2-9: Sample Entity Relationship Diagram (ERD)

With the E/R notation:

- The cardinalities and optionalities of a relationship are the representations of the number of occurrences of an entity that may be related to an occurrence of another entity.
- Business rules related to data accuracy and integrity at the logical level are translated into and implemented at the physical level either as triggers and stored procedures on the database, or in the process logic written using some programming languages.

2.2.2 Data Modeling Using UML Notation

A conventional UML class diagram models both data and behaviour. Within the scope of the IMH the class diagram is used to represent data. All the classes that appear in the diagram are the entity (or persistent) classes, with their behaviour definitions suppressed. Class diagrams are used to model entity (or persistent) classes and their associations with other entity classes. A sample class diagram is provided in Figure 2-10.

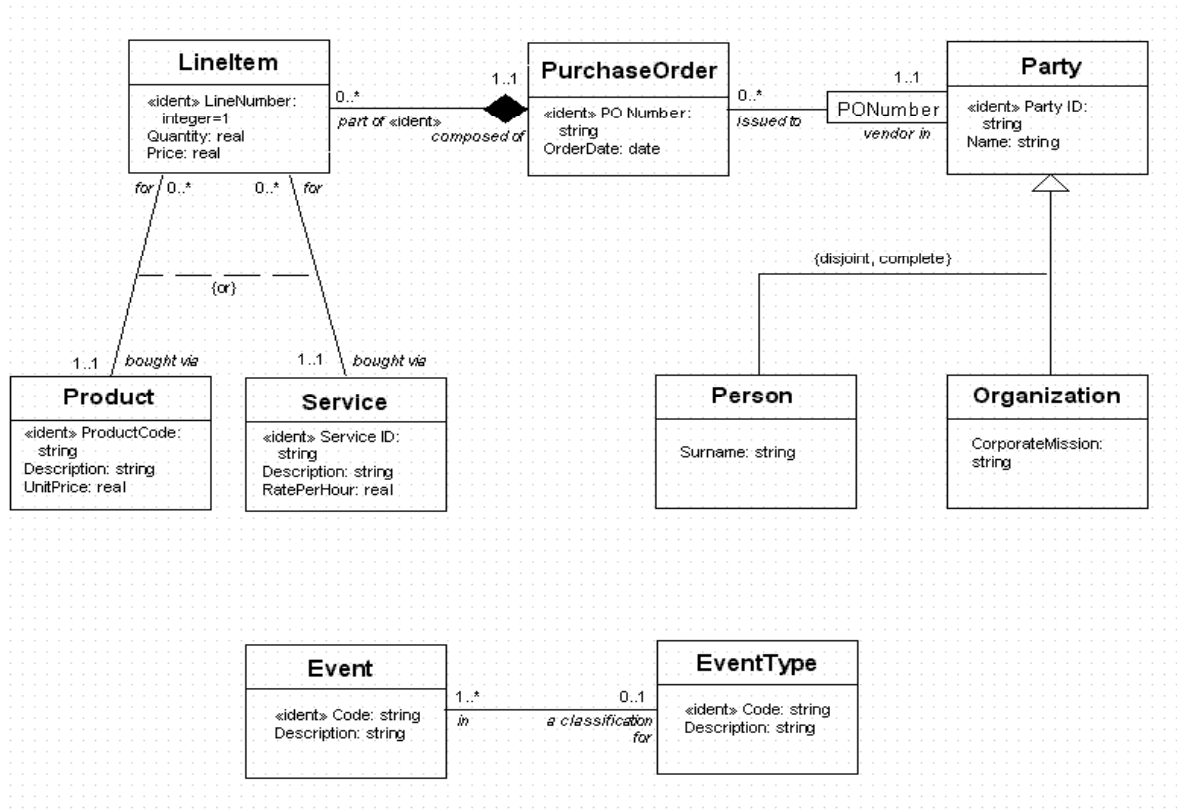


Figure 2-10: Sample UML Class Diagram

There are 3 types of class relationships:

1) **Association** - The relationship between two or more classes specifies connections between the instances of classes.
2) **Aggregation/Composition -** A special form of association that models a whole-part relationship.
3) **Generalization -** A specialization/generalization relationship.

With UML notation:

- The multiplicities of an association are the representations of the number of instances of a class that may be related to an instance of another class.
- Business rules related to data accuracy and integrity at the logical level are translated into and implemented at the physical level either as triggers and stored procedures on the database, or can be implemented by application functions.

2.2.3 Mapping E/R to UML Terminology

The notation of UML class diagrams corresponds to that of E/R diagrams. Where E/R diagrams show “entities” and “relationships”, UML class diagrams show “classes” and “associations”. In either case, these ultimately are definitions of the things that are of significance to the business. Table 2-3 provides a mapping of terminology between E/R and UML.

E/R Terminology	UML Terminology
Entity Relationship Diagram	Class Diagram
Entity	Class (stereotyped as <<Entity>>)
Instance of an Entity	Object
Relationship	Association
Cardinality/Optionality	Multiplicity
Supertype/Subtype	Generalization and Inheritance
Attributes	Attributes

Table 2-3: Mapping E/R to UML Terminology

2.2.4 Coexistence of E/R Data and UML Class Models

Although E/R data models and UML class models are developed using two different approaches and modeling notations, they have many similarities and complementary features, and they are able to coexist. Figure 2-11 illustrates how both E/R and UML modeling notations can be used to develop conceptual and logical data models.

In Figure 2-11, vertical arrows with solid line indicate that one data model can be directly created based on another data model using the same or similar modeling notation, while arrows with dashed line indicate that a data model developed using one modeling notation can later be transformed and converted into another data model of the same or different level using a different modeling notation.

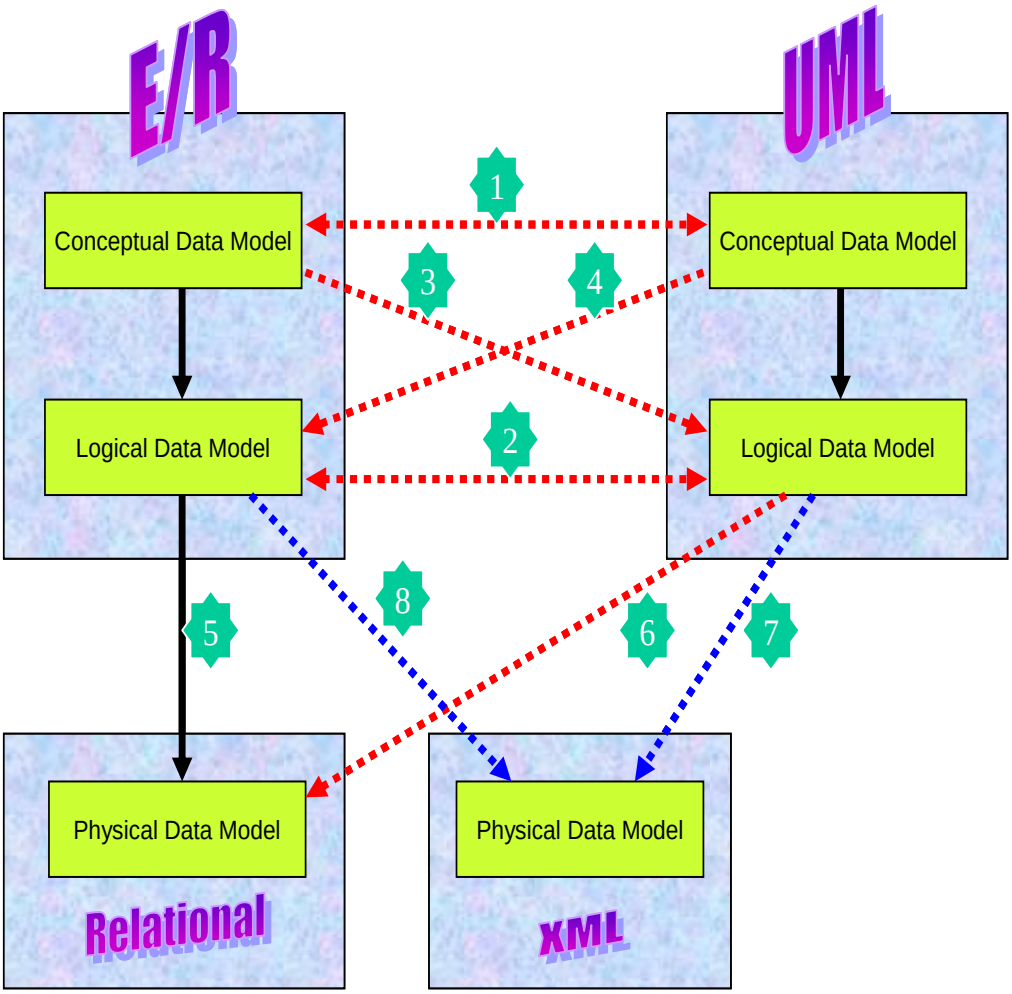


Figure 2-11: Coexistence and Switching Between E/R and UML Modeling

Figure 2-11 illustrates that there are 8 different points in the data solution design and development process in which change of modeling notation takes place. They are:

1) After completing a conceptual data model using either E/R or UML notation, the project develops another conceptual data model using the other modeling notation (path 1). This change may be triggered by the project's approach in doing business scope and requirements analysis.
2) After completing a logical data model using either E/R or UML notation, the project develops another logical data model using the other modeling notation (path 2). This change may be triggered by the project's decision on the target technology platform upon which the database solution will be implemented.
3) After completing a conceptual data model using E/R notation, the project switches over (path 3) to create a logical data model using UML notation. This change may be triggered by the project's decision to use an object oriented approach for application design and development.
4) After completing a conceptual data model using UML notation, the project switches over (path 4) to create a logical data model using E/R notation. This change may be triggered by the project's decision to follow structured system analysis and design approach for application development.
5) After completing the logical data model using E/R notation, the project decides to use the relational technology as a solution implementation tool, and develop a physical data model (path 5) describing the database solution design using a relational database management system.
6) After completing the logical data model using UML notation, the project decides to use the relational technology as a solution implementation tool, and develops (path 6) a physical data model describing the database solution design using a relational database management system.
7) After completing the logical data model using UML notation, the project decides to use an alternative solution implementation tool (e.g. XML) and develops (path 7) a physical data model to describe data solution design using the specific technology implementation tool.
8) After completing the logical data model using E/R notation, the project decides to use a non-relational based solution implementation tool (e.g. XML, Hierarchical, etc.) and develops (path 8) a physical data model to describe data solution design using the specific technology implementation tool.

2.3 Standard Properties of an E/R or a UML Class Diagram

An E/R or a UML class diagram must contain the following properties:

- **Title** - The identifying information that appears on the diagram. This should include the name of the diagram and the level of the model (i.e. conceptual, logical or physical).
- **Author** - The name of the person who last modified the diagram.
- **Description** - Free form text that describes the scope or purpose of the diagram.

- **Created** - The date/time the diagram was created.
- **Modified** - The date/time the diagram was last updated.
- **Content** - The appropriate data model level elements (i.e. entities, relationships, and entity attributes) should be shown as specified for each data model type.

2.4 Entity / Class

2.4.1 What Is an Entity / Class?

Entities / classes are collections of objects with common structure, common behaviour, common relationships and common semantics. They are used to identify persons, places, things, events or concepts of interest to the business. There must be more than one occurrence or potential for more than one occurrence of the entity / class.

2.4.2 Types of Entity / Class

Different types of entities / classes are required to provide a complete and accurate representation of an organization's data and to enable the analyst to use the ERD or the class diagram as a starting point for physical database design. An entity / class can be:

- **Fundamental -** The entity / class depends on no other entity / class for its existence.
- **Attributive** - The entity / class depends on another entity / class for its existence.
- **Associative** - The entity / class describes a connection between two entities / classes with an otherwise many-to-many relationship.
- S**upertype** / **Subtype** - The entity / class (the subtype) inherits the attributes of another entity (the supertype). In UML the term *generalization* is used to denote a derivation of the general class (the supertype) to the specialized class (the subtype).

2.4.3 Standard Properties of Entity / Class

An entity / class may contain the following properties depending on the level of the model.

- **Name** - A word or phrase that is used to identify an entity. See Entity naming standards for details.
- **Short Name** - A shorter name by which the entity / class may be called.
- **Plural Name** - A plural version name by which the entity / class may be called.
- **Description** - Free form text that describes the general features or characteristics of the object represented by the entity / class. This should include supporting **examples**, and possibly information about **what it is not**. The definition should not specify who uses the entity / class, how, where, or when.
- **Alias** - Alternative names by which the entity / class is commonly known (also known as a Synonym).

- **Primary Key** - A surrogate key or candidate key that is chosen as the unique identifiers for an entity. A primary key may consist of a single attribute/column or multiple attributes/columns. The primary key may be chosen from a Business Unique Identifier or system generated surrogated key.
- **Business Unique Identifier** - An identifier with business meaning by which an entity can uniquely be identified. Business unique identifiers may be in the form of identifying relationships and/or business attributes.
- **Data Steward** - The individual or department that has the responsibility for making classification and access decisions regarding the collection, transformation, use, access, disclosure, and disposal of information. The data steward is the owner and manager of the data and in particular of the entity / class.
- **Data Custodian** - The individual or department that has the responsibility for performing the intermediate information processing functions (process, communicate, store) related to information.
- **Data Source** - An indication of where the data (entity / class) originated. It may refer to an internal or external system or document.
- **Information Sensitivity and Privacy Classification (ISPC)** - All information resources must be classified according to their information sensitivity and privacy classification to ensure they receive the required protection. An entity / class must be classified by its owner (data steward) and clearly labeled to identify what safeguards are appropriate. Refer to the relevant information sensitivity and privacy classification standard for the classification levels.
- **Business Rule** - Free form text that describes a business condition under which data items are created, related and maintained
- **Volume** - A prediction of the initial number and maximum number of instances of an entity / class. This quantity may, for example, be used to predict storage requirements.
- **Volatility** - A prediction of the expected number of annual additions, changes and deletions to the instances of an entity / class. This quantity may, for example, be used to predict storage requirements.
- **Retention Requirements** - The maximum length of time for which instances of an entity / class must be retained for online data access and archived data storage. This may, for example, be used to predict storage requirements as well as to design the application in such a way to archive data when the retention period has been met. The terminology for this would be derived from the associated records retention schedule.

2.5 Attribute

2.5.1 What is an Attribute?

Attributes are properties that describe the characteristics of an entity / class. The attributes of an entity / class represent the information kept about the entity / class occurrence, which is relevant to the business operations and functions.

An attribute identifies elementary characteristics of the entity / class by associating one and only one data value with a single occurrence of that entity / class. Attributes are used to either identify the entity / class, to describe it in more detail or to relate two entities / classes.

2.5.2 Standard Properties of Attribute

An attribute may contain the following properties depending on the level of the model.

- **Name** - A word or phrase that is used to identify an attribute. See Attribute naming standards for details.
- **Description** - Free form text that describes the general features or characteristics represented by the attribute. This should include supporting **examples** (instances of values that would be consistent with this attribute's use), and possibly information about **what it is not**. The description should provide more information about the attribute than the name provided.
- **Alias** - An alternative name by which an attribute is commonly known.
- **Identifier Indicator** - An indication of whether this attribute is the identifier for the entity / class. At least one attribute (or relationship) on each entity /class must be either the identifier or part of the identifier. Every entity / class must have an identifier.
- **Optionality** - The indication that the attribute is either mandatory (required for every instance of an entity / class) or optional.
- **Information Sensitivity and Privacy Classification (ISPC)** - All information resources must be classified according to their information sensitivity and privacy classification. If the attribute requires a different information sensitivity and privacy classification from the entity / class, then it must be documented to ensure it receives the required protection. Refer to the relevant information sensitivity and privacy classification standard for the classification levels.
- **Data Type** - The format and domain information for an attribute is defined using its data type. The data type for an attribute must be global as it can be used by more than one attribute.
- **Size** - The length of an attribute.
- **Format** - The formats for dates, times, decimal numbers, etc.
- **Units** - If the attribute is a measurement in some consistent units then the units must be explicitly defined.
- **Domain** - An indication of the domain to which the attribute belongs.
- **Valid Values** - A specific, limited set of legitimate values which can be documented. This documentation should include both the values themselves as well as an indication of their particular meanings.
- **Default Value** - If relevant, initial value when created (system date for a date, time, attribute, flag default to “N”).
- **Validation Rules** - Beyond the domain definition, if relevant, a list of rules by which it can be determined that a proposed value for an attribute meets all criteria for legitimacy.

Some of the properties of attributes can also be defined as a group property called Domain. See section 2.6 for more details.

2.6 Domain

2.6.1 What is a Domain?

A domain is a named set of values (or range of values) of specified data type, length and format from which one or more entity attributes or data items draw their values. Domains are usually created in the data modeling process when data modelers come across situations, in which two or more data attributes share the identical set of data values.

The definition of a domain ensures that there is consistency across the data model with regard to how these attributes are defined (data type, length and format) and/or the values that are declared for them. Specifying a domain supports the ability to perform global changes at the attribute level without actually modifying every data attribute individually in a data model.

Domains must be defined and used with caution; over specified domains may lead to overhead in their management and use.

2.6.2 Standard Properties of Domain

A domain may contain the following properties depending upon the level of the model.

- **Name** - A word or phrase that is used to identify a domain. See Domain naming standards for details.
- **Description** - Free form text that describes the general features or characteristics represented by the domain. This should include supporting examples to clarify the meaning and use of the domain. The description should provide more information about the domain than the name provided.
- **Data Type** - The format for the attribute within the domain. The data type for a domain must be global so that it can be used by more than one attribute.
- **Size** - The average and maximum lengths for attributes in the domain. If required, this should include the number of decimal places.
- **Format** - The formats for dates, times, decimal numbers, etc.
- **Units -** The unit of measure for attributes in this domain (e.g. meters, kilograms, dollars)
- **Valid Values -** A specific, limited set of legitimate values that can be documented. It includes both the values themselves and an indication of their particular meanings. Do not list code values as allowed values. These values may end up in a database constraint, which is probably not the intent.
- **Default Value** - The default value(s) for the attributes in a domain (e.g. 0, ‘N’, ’SYSDATE’).

2.7 Relationship / Association

2.7.1 What is a Relationship / Association?

A relationship / association is a direct connection between two or more entities / classes, which is of interest to the business. A relationship represents a business rule. Every entity must have at least one relationship.

2.7.2 Standard Properties of Relationship / Association

A relationship / association may contain the following properties depending on the level of the model:

- **Name** - It is a verb phrase. Assign relationship names that are significant to the business or that are commonly understood in everyday language.

Note: It is a standard practice to state the relationship from the primary (parent) entity to the secondary (child) entity.

- **Cardinality** - Cardinality is the minimum and maximum number of instances in which an entity participates in a relationship.

There are 3 main types of relationship cardinality:

1) One-to-One
2) One-to-Many
3) Many-to-Many

- **Optionality -** Optionality helps identify whether or not an entity may participate in a relationship. An entity is then viewed as optional if the minimum number of instances in which it participates is zero (0).
- **Multiplicity -** In UML the term *multiplicity* covers both cardinality and optionality. Cardinality and optionality are conveyed in the form:

<lower limit>..<upper limit>

where the <lower limit> denotes the optionality (nearly always 0 or 1), and the <upper limit> denotes the cardinality. The <upper limit> may be either an asterisk (*) for the generic “more than one” or it may be an explicit number.

2.7.3 Relationship between Supertype and Subtype

A supertype can have multiple subtypes. The relationships between the supertype and subtypes can be inclusive or exclusive:

- Inclusive OR (can select one or multiple subtypes).
- Exclusive OR (can only select one subtype). Note: UML does not support Exclusive OR.
- AND (must select all subtypes).

2.7.4 Recursive Relationship

Sometimes, an entity/class has a relationship/association to itself. This does not mean that an instance of that entity/class has relationship/association to itself. Instead, it means that one instance of that entity/class has relationship/association to other instances of the same entity/class. For example, Employee reports to Manager (a manager is also an employee). This type of relationship is called a recursive relationship.

2.8 Data Model Metadata

Metadata is data that describes data. The information that defines each object in a data model, at each level, is/are all metadata.

When a modeler creates a data model diagram, he/she defines the way that data is represented conceptually, logically and physically in the organization. The modeler creates data entities, their attributes, and relationships between the data entities, names for the data, data domains, and other information that represents how data is defined for an enterprise, project, or subject area. All of this information is metadata.

A data dictionary is a centralized repository of metadata. Table 2-4 lists the metadata (objects and attributes) which are required to support information modeling standards in the creation and maintenance of data models.

Metadata should be captured once and be used or referenced many times when related data models are developed. For example, the metadata that is captured during the development of a conceptual data model should be inherited and used in the development of the corresponding logical data model. Similarly, the metadata defined during the development of the logical data model should be inherited and used in the development of the related physical data model.

Table Legend

Optional - **May** be included in the data model metadata if it will add useful information.

Mandatory - **Must** be included in the data model metadata.

Mandatory* - Must be included in the data model metadata for those data entities (or attributes) that are of significant interest to the business.

Mandatory - Must** be included in the data model metadata whenever it is known.

Mandatory* - Must** be included in the data model metadata for those data attributes that are required for:

- Data to be persisted in the database that requires precision and accuracy (i.e. minimum number of digits before and after the decision point in a number, the minimum number of characters in a text field, and/or special data value set);
- data to be migrated;
- data required to interface with other applications.

Please consult your cluster information architects or corporate information architects for clarification.

Table 2-4: Element Metadata for Different Levels of Data Models

Element	Element Property	Conceptual Data Model	Conceptual Data Model for Acquired Solution	Logical Data Model	Physical Data Model
Diagram	Title	Mandatory	Mandatory	Mandatory	Mandatory
	Author	Mandatory	Mandatory	Mandatory	Mandatory
	Description	Mandatory	Mandatory	Mandatory	Optional
	Created	Mandatory	Mandatory	Mandatory	Mandatory
	Modified	Mandatory	Mandatory	Mandatory	Mandatory
Entity / Table	Name	Mandatory	Mandatory	Mandatory	Mandatory
	Short Name	N/A	N/A	Optional	Mandatory
	Plural Name	N/A	N/A	Optional	Optional
	Description	Mandatory	Mandatory	Mandatory	Mandatory
	Primary Key (Candidate or Surrogate Key)	N/A	N/A	Mandatory	Mandatory
	Business Unique Identifier (Unique Key)	Optional	Mandatory**	Mandatory**	Mandatory**
	Alias (or Synonym)	Optional	Mandatory**	Mandatory**	Mandatory**
	Data Steward	Optional	Mandatory*	Mandatory	Mandatory
	Data Custodian	Optional	Mandatory*	Mandatory	Mandatory
	Data Source	N/A	Mandatory*	Mandatory*	Mandatory
	Information Sensitivity and Privacy Classification (ISPC)	N/A	Mandatory*	Mandatory	Mandatory
	Business Rules	Optional	Mandatory*	Mandatory**	Mandatory** (implemented as Table Constraints, stored procedures or database triggers)
	Volume	N/A	Mandatory*	Mandatory*	Mandatory
	Volatility	N/A	Mandatory*	Mandatory*	Mandatory

Element	Element Property	Conceptual Data Model	Conceptual Data Model for Acquired Solution	Logical Data Model	Physical Data Model
	Retention Requirements	N/A	Mandatory*	Mandatory*	Mandatory
Attribute / Column	Name	Mandatory (representative attributes only)	Mandatory	Mandatory (including class word)	Mandatory (including class word)
	Description	Mandatory (representative attributes only)	Mandatory	Mandatory	Mandatory
	Alias	Optional	Mandatory**	Mandatory**	Mandatory**
	Identifier Indicator	Optional	Mandatory**	Mandatory	Mandatory
	Optionality	Optional	Mandatory	Mandatory	Mandatory
	Information Sensitivity and Privacy Classification (ISPC)	Optional	Mandatory**	Mandatory**	Mandatory**
	Domain	Optional	Mandatory***	Mandatory**	Mandatory**
	Data Type	Optional	Mandatory***	Mandatory (not required if domain is used)	Mandatory (not required if domain is used)
	Size	Optional	Mandatory***	Mandatory (not required if domain is used)	Mandatory (not required if domain is used)
	Format	Optional	Mandatory***	Mandatory** (not required if domain is used)	Mandatory** (not required if domain is used)
	Units	Optional	Mandatory***	Mandatory** (not required if domain is used)	Mandatory** (not required if domain is used)
	Valid Values	Optional	Mandatory***	Mandatory** (not required if domain is used)	Mandatory** (not required if domain is used)

Element	Element Property	Conceptual Data Model	Conceptual Data Model for Acquired Solution	Logical Data Model	Physical Data Model
	Default Value	Optional	Optional	Mandatory** (not required if domain is used)	Mandatory** (not required if domain is used)
	Validation Rules	N/A	Mandatory*	Mandatory**	Mandatory** (Implemented as a check constraint, lookup tables, stored procedures or database triggers)
Domain	Name	N/A	Mandatory*	Mandatory	Mandatory (Implemented as a check constraint)
	Description	N/A	Mandatory*	Mandatory	Mandatory
	Data Type	N/A	Mandatory*	Mandatory	Mandatory
	Size	N/A	Mandatory*	Mandatory	Mandatory
	Format	N/A	Mandatory**	Mandatory**	Mandatory**
	Units	N/A	Mandatory**	Mandatory**	Mandatory**
	Valid Values	N/A	Mandatory**	Mandatory**	Mandatory**
	Default Value	N/A	Mandatory**	Mandatory**	Mandatory**
Relationship	Name	Mandatory	Mandatory	Mandatory	Mandatory (Implemented as a Foreign Key)
	Direction	Mandatory	Mandatory	Mandatory	Mandatory (Implemented as a Foreign Key)
	Cardinality	Mandatory	Mandatory	Mandatory	Mandatory
	Optionality	Mandatory	Mandatory	Mandatory	Mandatory
Table	Name	N/A	N/A	N/A	Mandatory

Element	Element Property	Conceptual Data Model	Conceptual Data Model for Acquired Solution	Logical Data Model	Physical Data Model
Constraint (e.g. primary key, foreign key, unique key, check constraint)	Definition	N/A	N/A	N/A	Mandatory
Index	Name	N/A	N/A	N/A	Mandatory
	Definition	N/A	N/A	N/A	Mandatory
Non-Table Objects (e.g. Views, Sequences, Procedures, Triggers, etc)	Name	N/A	N/A	Optional	Mandatory*
	Definition	N/A	N/A	Optional	Mandatory*

3. Data Modeling for Decision Support

Decision support systems require data modeling activities in addition to those described in Chapter 2.

This section presents two types of data models related to decision support; 1) a data model used for **integration** purposes and 2) a data model used for **presentation** purposes. These models should be developed using the Logical Data Model (LDM) of the operational data source as the foundational model.

Data modeling for **integration** purposes aims to capture snapshots of significant changes in data across time, and should be developed based on the LDM of the operational data source. The integration is an amalgamation of data from multiple sources which is transformed to align with the integration data model. The integration data model maintains consistency through the enforcement of enterprise business rules and metadata

Data modeling for **presentation** purposes aims to facilitate data analysis (slicing and dicing and rolling up facts and measurements by dimensions), and is based on dimensional modeling. To ensure consistency when restructuring data for analysis purpose, a dimensional model is 'derived' from the LDM created for integration purposes. The dimensional model is structured by adequately separating descriptive attributes (placed in dimension entities) from measurement / additive attributes (placed in fact entities).

3.1 Types of Data Models for Decision Support

Table 3-1 outlines the different data model types, dependent on the approach being used and the different levels of abstraction.

Table: 3-1: Types of Data Models for Decision Support

Levels of Abstraction	Data Model Type	Purpose	Data Lineage
Integration Level 1	Data Warehouse Logical Model	Integration: To define data requirements and to integrate data from model sources.	Developed using the LDM of the operational data source as the foundation model
Integration Level 2	Data Warehouse Physical Model	Integration: To optimize physical structures for the RDBMS being used for implementation.	Developed using the Data Warehouse Logical Model
Presentation Level 1	Fact and Dimension Matrix	Presentation: To define high-level information requirements in a dimensional manner	Developed based on decision support requirements and leveraging data definitions from the existing CDMs and/or LDMs of the operational data sources
Presentation Level 2	Logical Dimensional Model	Presentation: To define detailed information requirements in a dimensional manner	Developed using the Fact and Dimension Matrix and leveraging data definitions from the existing LDMs of the operational data sources
Presentation Level 3	Physical Dimensional Model	Presentation: To optimize physical structures for the RDBMS being used for implementation.	Developed using the Logical Dimensional Model

3.1.1 Fact and Dimension Matrix

The fact and dimension matrix represents a high-level view of business information requirements that are in scope for a decision support solution. The matrix illustrates different data analysis perspectives (i.e. dimensions) and may lead to the development of a data warehouse and/or data mart to support business intelligence initiatives.

The fact and dimension matrix documents the different fact groups, their dimensions, where these dimensions are shared, and the need for conformed dimensions. It suppresses technical details by including the fact groups and dimensions that have a business meaning and the important relationships between these fact groups and dimensions.

The fact and dimension matrix is independent of technology and database platform. It is used to document and communicate the scope of the data warehouse and/or data mart(s) supporting a decision support solution.

The fact and dimension matrix is a cross-reference table in which potential dimensions are shown in the columns and the fact groups in the rows. Intersections are marked where a dimension may exist for a fact group. The fact groups represent primary business processes of the organization. The rows of the matrix correspond to fact groups. Separate matrix rows are created if the data sources are different, the processes are different, or if the original matrix row represents some complex fact groups that need be sub-divided and tackled in more than one task. The columns of the matrix represent the common dimensions used across the enterprise.

The fact and dimension matrix should be developed by leveraging data definitions from the conceptual data models (CDM) and/or logical data models (LDM) of the operational data sources.

The fact and dimension matrix is delivered as part of the business architecture of a decision support system and helps to define and clarify the project scope. It fulfils the following objectives:

- It visually conveys the plan and scope for the data warehouse project.
- It helps prioritize which dimensions should be analysed and designed first for conformity given their prominent role in the data warehouse / data mart.
- It presents different views of business decision support information requirements.
- It extends areas of the CDM and/or LDM where additional details related to decision support type information requirements are needed.
- It clearly identifies the depth and breadth of potential business critical measures for an enterprise level reporting.

Format

The fact and dimension matrix is presented as a table and includes all of the mandatory elements. In most cases the definition will be found in the data model (CDM, LDM) that the fact and dimension matrix is being built on.

Impact of Not Developing This Matrix

The fact and dimension matrix demonstrates an understanding of the structure and content of the information requirements at a high level. The consequences of **not** developing this matrix include:

- No foundation to develop an overall data warehouse strategy.
- A danger of missing important information needs and their implications. Some key analytical requirements related to missing facts or dimensions may not be identified.
- An incomplete picture of an organization's needs which may result in erroneous recommendations regarding the development of some data warehouse areas.
- No assurance that business goals and objectives will be appropriately supported.
- Accuracy in estimating subsequent decision support related initiatives may be affected.
- An impact on requirements definition and project estimation activities when the scope and vision have not been clearly defined.
- The lack of basis for partitioning the organization's data and scoping subsequent data warehouse development.
- The construction of separate data marts that lack a framework to tie the data together. Stovepipe data marts deliver access to irreconcilable views of the organization and impact the development of a coherent warehouse environment. The effort required to retrofit these data marts into the warehouse architecture may exceed the effort to start the initiative again.

Audience

The primary audience for the fact and dimension matrix is business staff and is used for the purposes of review and approval. The fact and dimension matrix can also be used to communicate the scope of the data warehouse / data mart to senior business and IT management.

Scope

The fact and dimension matrix is a conceptual enterprise-wide view of the fact groups and their dimensions. For the purposes of this description, the term 'enterprise' can be the entire OPS, a cluster, a ministry, a ministry division, a program or an initiative crossing organizational boundaries. This matrix, in conjunction with the logical dimensional models for each fact group, is the basis for developing a data mart.

Mandatory Elements

- All in-scope fact groups and dimensions named and described.
- All candidate facts for the fact groups named and described. There should be no restrictions on whether these facts are consistent with each other.
- All dimension characteristics named and described. Include the dimension characteristics that are recognized by business staff (may or may not be business identifiers) and important to their understanding of a dimension's definition.
- Identification of potential conformed dimensions that can be determined across all fact groups.

Example

The following is an example of a fact and dimension matrix. The matrix shows the relationship between the possible fact groups and dimensions. Any dimension with more than one X implies that this dimension must be conformed across multiple fact groups.

	Fact Groups / Dimensions	Calendar (Date)	Time	Client (Party Role)	Internal Organization (Party Role)	Address (Location)	Geographic Location (Location)	Legislation / Policy / Standard (Authority)	Program	Product / Service (Service)	Access Channel	Service Provider (Party Role)	Agreement (Authority)	Driver (Party Role)	Vehicle (Resource)	Carrier (Party Role)	External Communication Subject (Event)	Internal Communication Subject (Event)	Examination Type (Event)	Inspection Type (Event)	License / Permit (Authority)	Investigation Type (Event)	Campaign (Program)	Violation Type (Event)	Inspector (Party Role)	Examiner (Party Role)	Customer Service Rep (Party Role)	Employee (Party Role)	Contractor (Party Role)	GL Account (Resource)	Asset (Resource)	Facility (Resource)	External Data (to be defined - e.g. Economic Trend, Stats Canada, other jurisdictions etc.)					
Driver	Driver Licensing	×	×	×	×	×	×			×	×	×	×	×			×	×	×		×	×		×		×	×	×		×		×	×						
	Driver Examination	×	×	×	×	×	×	×	×	×	×	×	×	×			×	×	×		×	×		×		×	×	×		×		×	×						
	Driver Performance	×	×	×		×	×	×	×				×				×	×	×		×		×	×	×	×	×		×				×						
	Driver Population	×	×	×		×	×			×				×			×	×			×		×			×	×						×						
	Driver Revenue	×	×		×		×	×	×	×	×	×	×	×			×	×	×		×	×		×			×	×		×		×	×						
Vehicle	Vehicle Registration	×	×	×	×	×	×	×	×	×	×	×	×		×	×	×	×			×	×					×	×		×		×	×						
	Vehicle Licensing	×	×	×	×	×	×	×	×	×	×	×	×	×			×	×		×	×	×		×	×		×	×		×		×	×						
	Vehicle Inspection Statistics	×	×		×	×	×	×	×	×	×	×		×			×	×		×	×	×	×	×	×		×	×		×		×	×						
	Vehicle Performance	×	×				×	×	×	×	×			×			×	×		×	×	×	×	×	×		×	×		×			×						
	Vehicle Population	×	×				×			×				×			×	×		×	×		×				×	×				×	×						
	Vehicle Revenue	×	×		×		×	×	×	×	×	×	×		×		×	×		×	×	×		×	×		×	×		×		×	×						
Carrier	Carrier Licensing	×	×	×	×	×	×	×	×	×	×	×	×			×	×	×		×	×	×		×			×	×		×		×	×						
	Carrier Performance	×	×	×		×	×	×	×	×	×	×				×	×			×	×	×	×	×	×		×	×		×		×	×						
	Carrier Inspection	×	×	×	×	×	×	×	×	×	×	×				×	×	×		×	×	×	×	×	×		×	×		×		×	×						
	Carrier Population	×	×	×			×			×						×		×			×			×			×	×					×						
	Carrier Revenue	×	×		×		×	×	×	×	×	×	×			×	×	×		×	×	×		×	×		×	×		×		×	×						
	Client Satisfaction (Customer Feedback)	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×		×	×						
Finance	Financial Planning / Management	×	×		×		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×						
Authorities	Legislation / Policy / Standards Effectiveness	×	×				×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×	×	×	×	×	×	×							
	Program Effectiveness	×	×				×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×		×							
	Campaign Effectiveness	×	×				×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×			×				×			×						
	Service Provider Performance	×	×				×		×	×	×	×	×	×	×	×	×		×	×				×	×	×		×		×		×							
	Product / Service Performance	×	×				×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×		×		×							
	Channel Performance Mgm't	×	×				×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×		×		×							
	Contract Management	×	×				×	×	×	×	×	×	×				×	×	×		×	×			×	×	×			×									
	Contact / Issues Management	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×		×							
	Publications	×	×		×		×	×	×	×	×		×	×	×									×	×	×													
HR	Employee Statistics	×	×		×		×	×	×	×	×	×	×			×	×	×	×		×	×		×	×	×	×		×		×							
	Position Statistics	×	×		×		×	×	×	×	×	×	×			×	×	×	×		×	×		×	×	×	×		×		×							
	Consultant Statistics	×	×		×		×	×	×	×	×	×	×			×					×	×						×	×		×							
Business Planning	Business Planning Effectiveness	×	×		×		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×							
Asset Management	Stock Management																																						
Communications Management																																							
Facilities Management																																							

Figure 3-1: Sample Fact and Dimension Matrix

3.1.2 Data Warehouse Logical Model

A data warehouse logical model represents the logical data structures to capture non-volatile data; in scope snapshot entities, their relationships, their 'primary' attributes, and some calculated attributes.

The data warehouse logical model is a logical data model (LDM) which has been enhanced or de-normalized with time stamping and the possible addition of some derived attributes. This model is independent from the physical environment (e.g. the relational technology), or any performance consideration.

Chapter 2 section 2.1.3 describes the standards relevant to the creation of a LDM which are also applicable to the data warehouse logical model.

3.1.3 Data Warehouse Physical Model

The data warehouse physical model is a description of the internal data structures used by the data warehouse. This model defines the physical implementation of the logical requirements using the selected database management system (RDBMS).

Chapter 2 section 2.1.4 describes the standards relevant to the creation of a PDM which are also applicable to the data warehouse physical model.

3.1.4 Components of a Dimensional Model

A dimensional model consists of the following elements:

- **Diagram -** A graphical representation showing dimension entities and fact entities, and the relationships between the two types of entities.
- **Fact Entity -** The primary fact entity in each dimensional model that is meant to contain measurements of the business. Every fact entity represents a many-to-many relationship and every fact entity contains a set of two or more relationships that join to their respective dimension entities.
- **Fact -** A business performance measurement, typically numeric and additive, which is stored in a fact table. Note: A fact can be considered as an attribute of a fact entity.
- **Dimension Entity -** An independent entity in a dimensional model that serves as an entry point or as a mechanism for slicing and dicing the additive measures located in the fact entity of the dimensional model. It provides context to the fact entity based on its descriptive attributes.
- **Dimension Attribute -** A property or characteristic that describes a dimension entity. The attributes of a dimension entity represent the information kept about the entity, which is relevant to the business.
- **Relationship -** An association that exists between two entities, usually between a dimension and a fact.
- **Hierarchy** - A series of dependencies between some attributes (e.g. day, month, year) within a dimension. Each hierarchy provides a 'valid' path to consolidate facts. Within a dimension several hierarchies may coexist (e.g. the time dimension could have the day-week and the day-month-year hierarchies).

3.1.5 Logical Dimensional Model

The logical dimensional model includes the details of the fact entities, facts, dimension entities, dimension attributes, and the relationships connecting the fact entities with the dimension entities.

The logical dimensional model is technology independent and can be implemented in a relational database management system (RDBMS) or multi-dimensional database. It is designed for query access to answer business questions. It is used to illustrate the capability of the data warehouse design to meet the decision support requirements. The logical dimensional model also illustrates the traceability from the high-level business information requirements, as presented in the fact and dimension matrix, to the detailed data mart design.

The logical dimensional model should be built from the fact and dimension matrix using the existing data definitions and inherent business relationships from the LDM(s) from the existing operational data sources in order to achieve consistency of data across the specific business domain

The logical dimensional model is used to identify the specific information requirements in the scope of the decision support initiative.

It is critical to have a clear understanding of the business requirements:

- To understand all of the business information that is required.
- To explain the contents of the data warehouse to the business staff as it relates to their analytical requirements.
- To optimize data query performance to support data analysis.

Format

The logical dimensional model includes one or more dimensional model diagram and supporting metadata of all the mandatory elements.

Impact of Not Developing This Model

The impacts of not developing the logical dimensional model include:

- Lack of understanding of analytical requirements
- Lack of understanding of the data elements and their meaning
- Sub-optimal performance of analytical queries and reports

Audience

The primary audience for the logical dimensional model includes the potential users of the data warehouse or data mart. On the business side this would be staff who will be analyzing data from the data warehouse or data mart. The logical dimensional model will also be of interest to IT staff who will implement the model and develop the ETL (Extract, Transform and Load) processes.

Scope

The scope of a logical dimensional model can be broad, such as enterprise (e.g., a series of interconnected data marts via conformed dimensions), or narrower in focus such as a subject area data mart (e.g., a star schema).

Mandatory Elements

- All in-scope fact entities named and described with fact grain specified.
- Data steward, data custodian, data source, information sensitivity and privacy classification, volume, volatility and retention requirements specified for each fact entity.
- Primary key (candidate key) identified for each fact entity.
- All facts named and described, with optionality, derived fact type, aggregation rules, formulas, constraints, transformations, data source, and domain specified.
- All in-scope dimension entities named and described.
- Data steward, data custodian, information sensitivity and privacy classification, retention requirements, volatility and volume information specified for each dimension entity.
- Primary key (surrogate key) and any unique identifiers (business identifiers) identified for each dimension entity.
- All dimension attributes named and described, with optionality, slowly changing policy, cardinality, data source, and domain specified.
- Validation information specified for each fact entity, fact, dimension entity, and dimension attribute.
- All domains named and described including specification of data type, length, and allowed values.
- All relationships named and with cardinality / optionality specified.
- Includes a mapping table that shows the traceability between the logical dimensional model and the data warehouse logical model.

Example

Figure 3-2 is an example of a logical dimensional model diagram for a single data mart that was developed using the fact and dimension matrix shown in Figure 3-1.

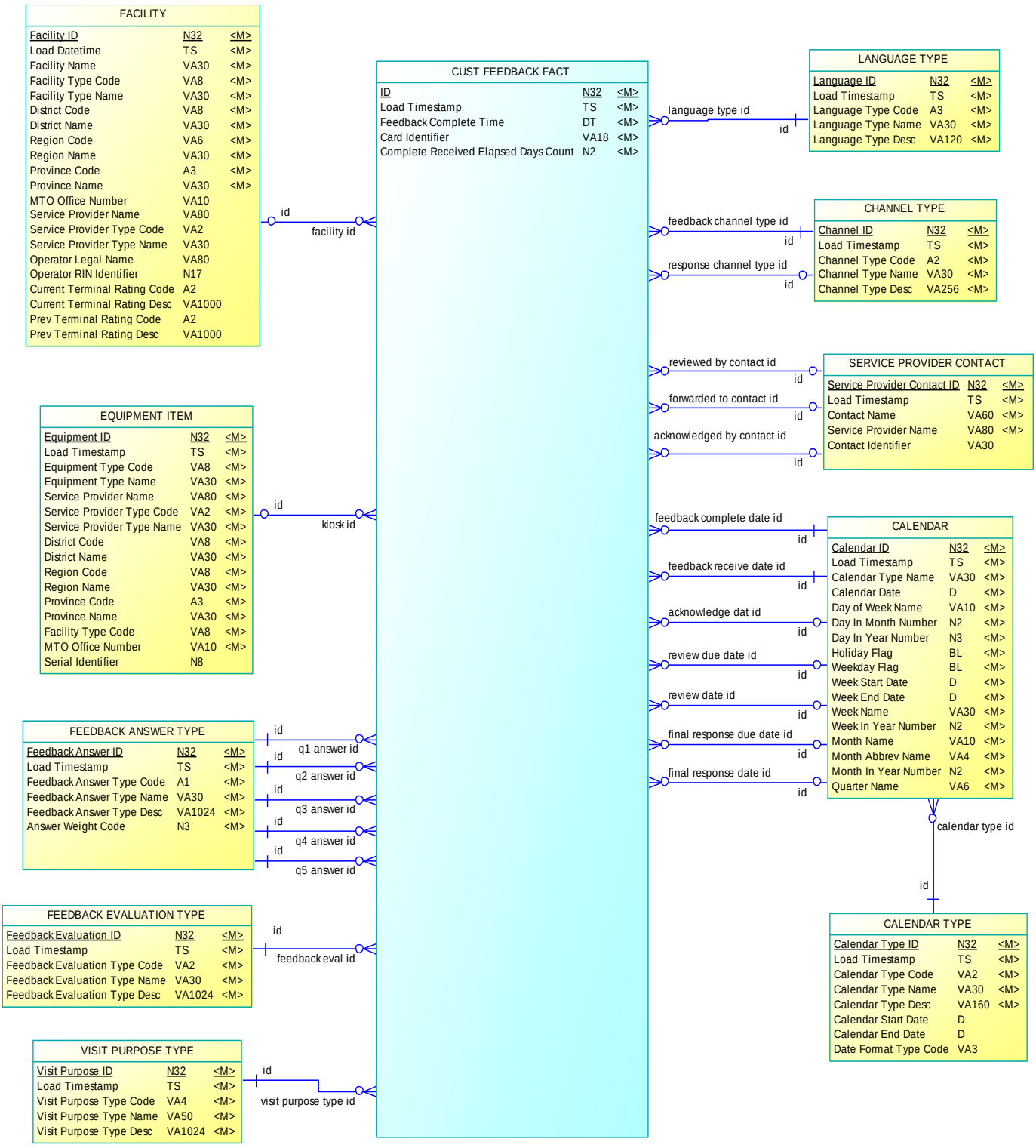


Figure 3-2: Sample Logical Dimensional Model Diagram

3.1.6 Physical Dimensional Model

The physical dimensional model is a description of the internal data structures used by the data mart. It is used to define the physical database objects for implementation specific to the selected RDBMS.

The physical dimensional model is composed of one or more table(s) with a multi-part key, known as a fact table, and a set of smaller tables which are called dimension tables. Each dimension table has a single-part primary key that corresponds exactly to one of the components of the multi-part key in the fact table. This characteristic star-like structure is often called a “star schema”.

The physical dimensional model should be built from the logical dimensional model.

The physical dimensional model may vary from the logical dimensional model in that the physical dimensional model may introduce database objects that do not contribute to the business information requirements of the decision support solution. These objects may be added in order to speed response times, ensure that the solution fits within the physical limitations of the computing environment, or improve maintenance turnaround for example.

The physical dimensional model is used to define the physical database objects for implementation specific to the selected RDBMS. It is also required to maintain traceability to the logical dimensional model and to support efficient change management.

Format

The physical dimensional model includes one or more dimensional model diagram and supporting metadata of all the mandatory elements.

Impact of Not Developing This Model

If the physical dimensional model is not developed it will lead to inconsistencies between the logical dimensional model and what has been physically implemented. This will also be detrimental to having an efficient change management process.

Audience

The primary audience for this model is the Database Administrator (DBA). The physical dimensional model contains all the information that is needed by a DBA to implement and maintain the database.

Scope

The scope of a physical dimensional model can be broad, such as an enterprise (e.g., a series of interconnected data marts via conformed dimensions), or narrower in focus such as a subject area data mart (e.g., a star schema).

Mandatory Elements

- All fact and dimension tables for the data mart named and described including a short name.
- Primary key (surrogate key) and unique keys (unique identifiers) identified for each dimension table.

- Primary key identified for each fact table.
- Columns in each table named (including use of class words) and described including specification of data type, length, allowed values and validation information (domains), and optionality.
- Relating columns for each relationship (foreign keys replace relationship name) are identified and named to replace relationship name.
- All table constraints (primary key constraint, foreign key constraint, unique key constraints, check constraints), identified and named.
- Primary key (surrogate) sequences identified, named and defined.
- Indexing technique to be used for fact and dimension tables (e.g. B-Tree, Bitmapped, Hash) identified.
- Indexes (primary key index, foreign key index, unique key index, clustering index) identified, named and defined
- Any required database views identified, named and defined.
- Any required aggregate fact and dimension tables identified, named, defined and described.
- Any particularly complex file relationships and access schemes are described.
- Major decisions about the data structures and the reasons for the decisions are documented; for example, the reasons why multiple fact tables may have been consolidated. This may include modifications to the model to address user access tool requirements.
- Data steward, data custodian, data source, information sensitivity and privacy classification, volume, volatility and retention requirements specified for each fact table.

Optional Elements

- A synonym for each table.
- Storage definitions and tablespaces identified, named and defined.
- Storage definitions and tablespaces assigned to database objects (tables and indexes).
- Views identified for multiple relationships between a single dimension and fact table.

Example

Figure 3-3 is an example of a physical dimensional model diagram for a single data mart that was developed based on the logical dimensional model shown in Figure 3-2.

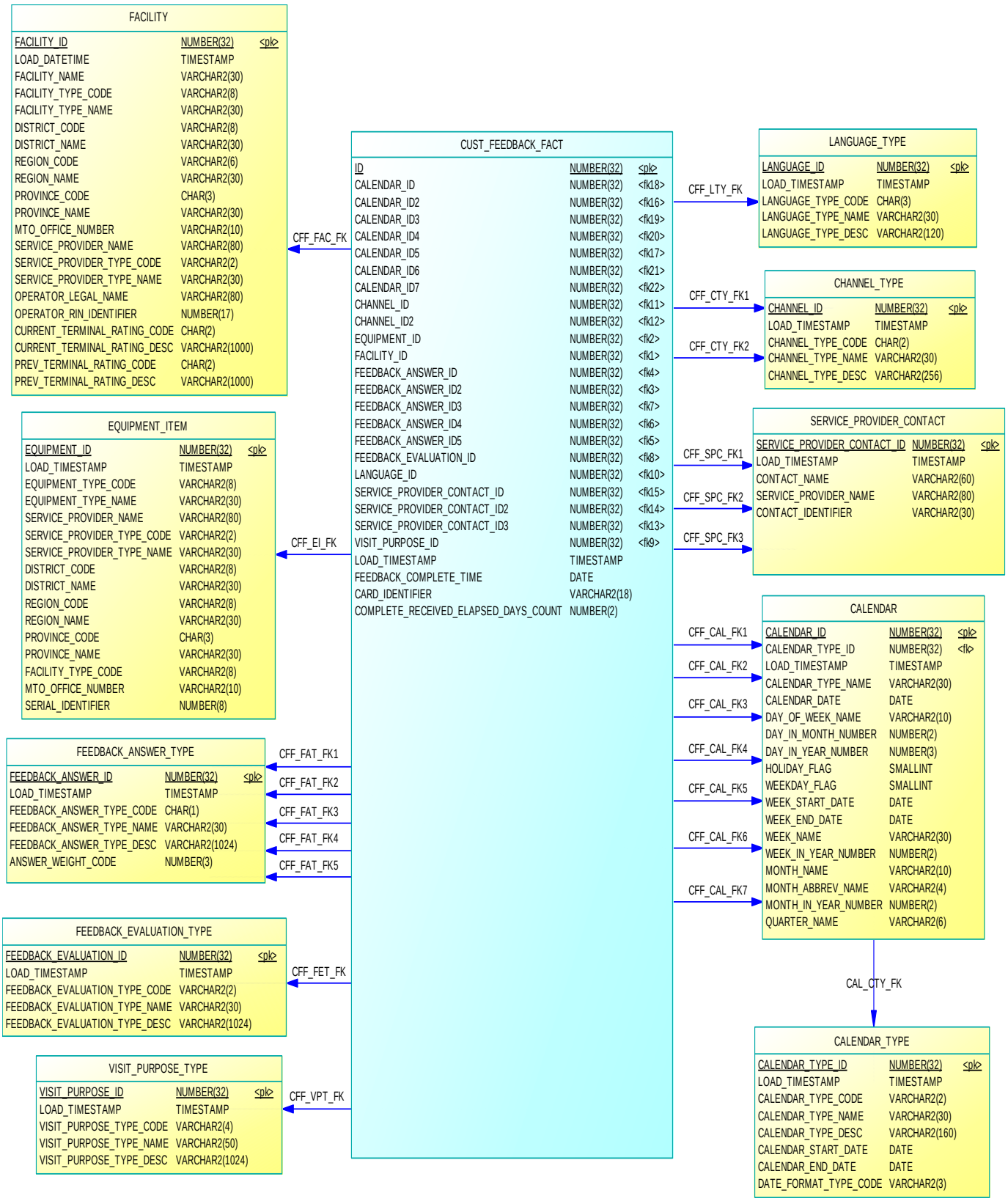


Figure 3-3: Sample Physical Dimensional Model Diagram

3.2 Dimensional Model Diagram

3.2.1 What is a Dimensional Model Diagram?

A dimensional model diagram is a schematic representation showing dimension entities and fact entities, and the relationships between the two types of entities. Like an ERD, a dimensional model diagram shows dimension entities and fact entities as rectangles. The fact entity, consisting of numeric measurements, is joined to a set of dimension entities filled with descriptive attributes. This characteristic star-like structure is often called a star schema.

3.2.2 Standard Properties of a Dimensional Model Diagram

A dimensional model diagram must contain the following properties:

- **Title** - The identifying information that appears on the diagram. This should include the name of the diagram and the level of the dimensional model (logical or physical).
- **Author** - The name of the person who last modified the diagram.
- **Description** - Free form text that describes the scope or purpose of the diagram.
- **Created** - The date/time the diagram was created.
- **Modified** - The date/time the diagram was last updated.
- **Content** – The diagram must contain at least one dimension entity and one fact entity.

3.3 Fact Entities

3.3.1 What is a Fact Entity?

A fact entity is the primary entity in a dimensional model where the numerical performance measurements of the business will be stored. A fact entity is modeled to capture the measurement data resulting from a business process.

Fact entities express the many-to-many relationships between dimensions in dimensional models. All fact entities have two or more relationships that connect them to dimension entities. A fact entity will be accessed via the dimension entities joined to it. A fact entity generally has a candidate key made up of a subset of the relationships to dimensions. Only a subset of the relationships is typically needed to guarantee uniqueness. The “grain” of a fact entity describes the level of detail (or level of aggregation) in the fact entity, and indicates whether it is transaction, periodic snapshot or accumulating snapshot.

3.3.2 Standard Properties of Fact Entities

All fact entities should have a standard description and properties appropriate with the level of model (logical or physical).

A fact entity can contain the following properties depending on the level of the dimensional model:

- **Name** - A word or phrase that is used to identify a fact entity. See Entity naming standards for details.
- **Description** - Contains the following sections:
  - **Definition** - If the fact entity name was abbreviated, include the full business name. The definition is free form text and should describe the general features or characteristics of the object represented by the fact entity.
  - **Examples** - Instances of objects that would belong to this fact entity.
- **Unique Identifier** - In a dimensional model, all identifiers should be surrogates. Keys from production systems or intelligent keys that contain business data (such as an office number that contains the region) should not be used. The identifier for a fact entity is a combination of the surrogates from the related dimensions.
- **Fact Grain** - Declaring the grain is equivalent to saying what is in an individual fact entity record. Typical choices for the grain of the fact entity could be expressed as follows:
  - Each license fee transaction is a fact record.
  - Each health insurance claim transaction is a fact record.
  - Each daily product sales total in each store is a fact record.
  - Each monthly account snapshot is a fact record.
  - Each line item on each order is a fact record.
  - Each line item on each shipment's invoice is a fact record.

In addition to declaring the fact grain, indicate the type of fact entity: transaction, periodic snapshot, or accumulating snapshot.

- **Alias** – An alternative name by which the fact entity is commonly known (also known as a Synonym).
- **Data Steward** – The individual or department that has the responsibility for making classification and access decisions regarding the collection, transformation, use, access, disclosure, and disposal of the fact entity. The data steward is the owner and manager of the data and in particular of the fact entity.
- **Data Custodian** - The individual or department that has the responsibility for performing the intermediate information processing functions (process, communicate, store) related to the fact entity.
- **Data Source** - The name of the source system which contains the data for the target fact entity. There may be more than one source system for a single fact entity.
- **Information Sensitivity and Privacy Classification (ISPC)** - All information resources must be classified according to their information sensitivity and privacy classification to ensure they receive the required protection. A fact entity must be

classified by its owner (data steward) and clearly labeled to signify what safeguards are appropriate. Refer to the relevant information sensitivity and privacy classification standard for the classification levels.

- **Volume** - A prediction of the initial number and maximum number of instances of a fact entity. This quantity may, for example, be used to predict storage requirements.
- **Volatility** - A prediction of the expected number of annual additions, changes and deletions to the instance of a fact entity. This quantity may, for example, be used to predict storage requirements.
- **Retention Requirements** - The maximum length of time for which instances of a fact entity must be retained for online data access and archived data storage. This may, for example, be used to predict storage requirements as well as to design the application in such a way to archive data when the retention period has been met. The terminology for this would be derived from the associated records retention schedule.

3.3.3 What is a Fact?

A fact is a business performance measurement, typically numeric and additive, that is stored in a fact table. Note: A fact can be considered as an attribute of a fact entity.

3.3.4 Standard Properties of Facts

A fact can have the following properties depending on the dimensional model level:

- **Name** - A word or phrase that is used to identify a fact. See Attribute naming standards for details.
- **Description** - Free form text that describes the general features or characteristics of the object represented by the fact.
- **Alias** – An alternative name by which the fact is known. This may become the label of the fact in data access tools and on reports.
- **Optionality** - The indication that the fact is either mandatory (required for every instance of a fact entity) or optional.
- **Information Sensitivity and Privacy Classification (ISPC)** - All information resources must be classified according to their information sensitivity and privacy classification. If the fact requires a different information sensitivity and privacy classification from the fact entity, then it must be documented to ensure it receives the required protection. Refer to the relevant information sensitivity and privacy classification standard for the classification levels.
- **Derived Fact Type** - An indication of the kind of derived fact, as per the following:
    - **Column -** An actual base fact. Basic information is pulled directly from a column in a fact table.
    - **Constraint** - A base fact with a predefined constraint. These are often created as building blocks for other calculations. For example, the total of Canadian sales units (sales units constrained on geography) is needed to create the final derived fact of the percentage of total Canadian sales units.
    - **Transformation -** A pre-defined process to translate a specific value within a fact entity to another value within the same fact entity. When this is supported

by a data access tool, it eliminates the need for a user to specify (at the time of query) both the specific value and the transformed value.

- **Calculation -** A straightforward calculation based upon one or more previously defined facts.
- **Column with Limits -** A semi-additive fact, one that cannot be aggregated across all dimensions. This requires specification of what the limits are for this fact.

- **Aggregation Rule** - The default aggregation rule for this derived fact. Sometimes the result of a calculation can be summed, but in other cases the components must be aggregated separately and the final derived fact recalculated. Every derived fact should have a default aggregation rule (sum, min, max, latest, semi-additive, special algorithm, and no aggregation).
- **Formula** – The specific mathematical formula used to create this derived fact (e.g. Sum (Dollar Sales)).
- **Constraints** – An indication of constraints that must be applied to the data in order to create this derived fact.
- **Transformations** – An indication of the specific translations that must be calculated when creating this derived fact. For example, the query specifies a reporting time period. A calculation may reflect prior period comparisons. This prior period must be based off the specific reporting period. In some cases, the user may be required to also enter the prior period specifically to apply to this derived fact.
- **Data Source –** An indication of where the data originated for the target fact. It may include the following level of detail:
  - **Source System** - The name of the source system which contains the data for the target fact. There may be more than one source system for a single fact.
  - **Source Table/File** - The name of the specific table or file which contains the data for the target fact. There may be more than one source table/file for a single fact.
  - **Source Column/Field** - The name of the specific column or field which contains the data for the target fact. There may be more than one source system for a single fact.
  - **Data Transformation** - Notes about any transformations that are required to translate the source information into the format required by the target fact. This is not intended to be a complete list of the specific business rules but a place to note any anomalies or issues that are known.
- **Data Type (Domain)** – The format and domain for a fact is defined using its data type. The data type for a fact must be global so that it can be used by more than one fact.
- **Size (Domain)** - The average and maximum lengths for the facts in the domain. If required, this should include the number of decimal places.
- **Format (Domain)** – The format for the fact (i.e.; date, timestamp, decimal number, etc.)
- **Units (Domain)** – If the fact is a measurement in some consistent units (e.g. meters, kilograms, dollars) then the units must be explicitly defined.
- **Valid Values (Domain)** - An indication of the specific, limited set of legitimate values for the fact. It includes both the values themselves and an indication of their particular

meanings. Do not list code values as allowed values. These values may end up in a database constraint, which is probably not the intent.

- **Default Value (Domain)** – If relevant, the default value(s) for the facts in the domain when it is created (e.g. system date for a date, time, attribute, flag default to "N").

3.4 Dimension Entities

3.4.1 What is a Dimension Entity?

Dimension entities are integral companions to fact entities. Dimension entities contain the textual and encoded descriptions (attributes) of the business.

3.4.2 Standard Properties of Dimension Entities

All dimension entities should have a standard description and properties appropriate with the level of model (logical or physical).

A dimension entity can contain the following properties depending on the level of the dimensional model.

- **Name** - A word or phrase that is used to identify a dimension. See Entity naming standards for details.
- **Description** - Contains the following sections:
  - **Definition -** If the entity name was abbreviated, include the full business name. The definition is free form text and should describe the general features or characteristics of the object represented by the entity.
  - **Examples -** Instances of objects that would belong to this dimension entity.
- **Unique Identifier** - In a dimensional model, all identifiers should be "surrogates". Keys from production systems or "intelligent" keys that contain business data (such as an office number that contains the region) should not be used. The identifier for a dimension entity should be a non-intelligent (system generated) unique identifier.
- **Alias** - Alternative names by which the dimension entity is commonly known (also known as a Synonym).
- **Data Steward** – The individual or department that has the responsibility for making classification and access decisions regarding the collection, transformation, use, access, disclosure, and disposal of information. The data steward is the owner and manager of the data and in particular of the dimension entity.
- **Data Custodian** - The individual or department that has the responsibility for performing the intermediate information processing functions (process, communicate, store) related to the dimension entity.
- **Data Source** - An indication of where the data for the dimension entity originated. It may refer to an internal or external system or document.
- **Information Sensitivity and Privacy Classification (ISPC)** - All information resources must be classified according to their information sensitivity and privacy classification to ensure they receive the required protection. A dimension entity must be classified by its owner (data steward) and clearly labeled to identify what safeguards are appropriate.

Refer to the relevant information sensitivity and privacy classification standard for the classification levels.

- **Volume** - A prediction of the initial number and maximum number of instances of a dimension entity. This quantity may, for example, be used to predict storage requirements.
- **Volatility** - A prediction of the expected number of annual additions, changes and deletions to the instances of a dimension entity. This quantity may, for example, be used to predict storage requirements.
- **Retention Requirements** - The maximum length of time for which instances of a dimension entity must be retained for online data access and archived data storage. This may, for example, be used to predict storage requirements as well as to design the application in such a way to archive data when the retention period has been met. The terminology for this would be derived from the associated records retention schedule.

3.4.3 What is a Dimension Attribute?

Dimension attributes describe the instances in a dimension entity. Dimension attributes are usually text fields and typically describe the characteristic of a tangible thing. For example, the brand and flavour of a product are well known attributes of the product and are probably displayed in a prominent way on the product packaging. The flavour of a product is not measured, it is known in advance. A dimension attribute can reside in one and only one dimension entity, whereas a fact attribute can be repeated in multiple fact entities.

3.4.4 Standard Properties of Dimension Attributes

A dimension attribute can contain the following properties depending on the level of the dimensional model (logical or physical).

- **Name** - A word or phrase that is used to identify a dimension attribute. See Attribute naming standards for details.
- **Description** - Contains the following sections:
    - **Definition -** Include the full business name of the dimension attribute. The definition of the dimension attribute is free form text and should describe the general nature of the characteristic represented by the dimension attribute. This section should also contain information about why this characteristic is of interest and for whom it is of interest (if these considerations are materially different for this particular attribute as compared to the parent entity as a whole).
    - **Examples -** Instances of values, which would be consistent with this dimension attribute's use. When this section is used, the examples should be accompanied by brief descriptions. This should be included only if the examples assist in clarifying the use or meaning of the dimension attribute.
- **Alias** - The label of the dimension attribute that is found in data access tools and on reports.
- **Slowly Changing Policy** - Dimension attributes are relatively static but do change slowly over time. Due to the long-term history of data in the data warehouse it is

important to have a strategy for dealing with the impact of changes to dimension attributes. Each dimension attribute in the dimension entity requires a strategy to handle change. There are four potential options available for dealing with slowly changing dimensions (Type 0, Type 1, Type 2, and Type 3).

- **Optionality** – An indication of whether the dimension attribute is mandatory (required for every instance of a dimension entity) or optional.
- **Information Sensitivity and Privacy Classification (ISPC)** - All information resources must be classified according to their information sensitivity and privacy classification. If the dimension attribute requires a different information sensitivity and privacy classification from the dimension entity then it must be documented to ensure it receives the required protection. Refer to the relevant information sensitivity and privacy classification standard for the classification levels.
- **Data Source** – An indication of where the data originated for the dimension attribute. It may include the following level of detail:
    - **Source System -** The name of the source system which contains the data for the target dimension attributes. Note: There may be more than one source system for a single dimension attribute.
    - **Source Table/File -**The name of the specific table or file which contains the data for the target dimension attributes. Note: There may be more than one source table/file for a single dimension attribute.
    - **Source Column/Field -** The name of the specific column or field which contains the data for the target dimension attributes. Note: There may be more than one source column/field for a single dimension attribute.
    - **Data Transformation -** Notes about any transformations that are required to translate the source information into the format required by the target dimension attribute. This is not intended to be a complete list of the specific business rules but a place to note any anomalies or issues that are known.
- **Data Type (Domain)** - The format and domain information for a dimension attribute is defined using its data type. The data type for a dimension attribute must be global as it can be used by more than one dimension attribute.
- **Size (Domain)** - Specifies the length of a dimension attribute.
- **Format (Domain)** – Specifies the formats for the dimension attribute (i.e.; date, timestamp, decimal number, etc.)
- **Units (Domain)** - If the dimension attribute is a measurement in some consistent units (e.g. meters, kilograms, dollars) then the units must be explicitly defined.
- **Valid Values (Domain)** – An indication of the specific, limited set of legitimate values for the dimension attribute. This documentation should include both the values themselves as well as an indication of their particular meanings. Do not list code values as allowed values. These values may end up in a database constraint, which is probably not the intent.
- **Default Value (Domain)** - If relevant, the initial value for the dimension attribute when it is created (e.g. system date for a date, time, attribute, flag default to “N”).

3.5 Domains

3.5.1 What is a Domain?

A domain in the context of a dimensional model is the same as a data model domain.

3.6 Relationships

3.6.1 What is a Relationship?

A relationship in a dimensional model represents the access path from dimension entity to fact entity. A fact entity will be accessed via the dimension entities joined to it. It serves to ensure that entities are related to each other in such a manner that the analysis and design based on the dimensional model can represent the business accurately.

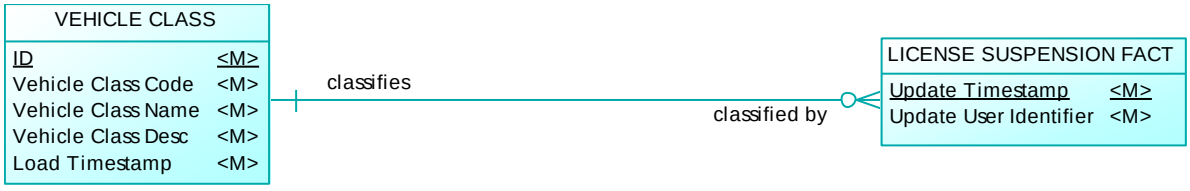
Relationship combinations such as inclusive OR, exclusive OR, AND are not used in a dimensional model.

3.6.2 Standard Properties of Relationships

A relationship may contain the following properties depending on the level of the dimensional model (logical or physical):

- **Name -** A verb phrase used to describe the relationship between the 2 dimension entities. Assign relationship names that are significant to the business or that are commonly understood in everyday language.
    - **Two names are required** - One at each end. One that describes the relationship from the perspective of the primary (parent) entity and another that describes the relationship from the secondary (child) entity perspective.
- **Cardinality** - Identifies the minimum and maximum number of instances in which a dimension entity participates in a relationship. Relationships in dimensional models are normally one-to-many (1:M).
- **Optionality** - Relationships in a dimensional model are usually optional (zero to many or 0:M).

For example:



The assumption in the example above is that each row in the LICENSE SUSPENSION FACT table must have a VEHICLE CLASS but not every VEHICLE CLASS must be represented in the LICENSE SUSPENSION FACT table.

3.7 Dimensional Model Metadata

Metadata is data that describes data. The information that defines each object within a dimensional model, at each level, is/are all metadata.

The data modeling tool data dictionary is used to capture metadata. A data dictionary is a centralized repository of metadata. Table 3-4 lists the metadata about the dimensional model that is maintained in the data dictionary.

Table Legend

Optional - **May** be included in the dimensional model metadata if it will add useful information

Mandatory - **Must** be included in the dimensional model metadata.

Mandatory - Must** be included in the data model metadata whenever it is known.

Please consult your cluster information architects or corporate information architects for more clarification.

Table 3-2: Dimensional Model Metadata

Object	Object Attribute	Fact and Dimension Matrix	Logical Dimensional Model	Physical Dimensional Model
Diagram or Matrix	Title	Mandatory	Mandatory	Mandatory
	Author	Mandatory	Mandatory	Mandatory
	Description	Mandatory	Mandatory	Optional
	Created	Mandatory	Mandatory	Mandatory
	Modified	Mandatory	Mandatory	Mandatory
Fact Entity (Fact Group)	Name	Mandatory	Mandatory	Mandatory (implemented as a Table)
	Short Name	N/A	Optional	Optional
	Plural Name	N/A	Optional	Optional
	Description	Mandatory	Mandatory	Mandatory
	Unique Identifier	N/A	Mandatory	Mandatory
	Fact Grain	N/A	Mandatory	N/A
	Alias (or Synonym)	Optional	Mandatory**	Mandatory**
	Data Steward	Optional	Mandatory	Mandatory
	Data Custodian	Optional	Mandatory	Mandatory
	Data Source	Optional	Mandatory	Mandatory

Object	Object Attribute	Fact and Dimension Matrix	Logical Dimensional Model	Physical Dimensional Model
	Information Sensitivity and Privacy Classification (ISPC)	Optional	Mandatory	Mandatory
	Volume	N/A	Mandatory	Mandatory
	Volatility	N/A	Mandatory	Mandatory
	Retention Requirements	Optional	Mandatory	Mandatory
Fact	Name	Mandatory (define candidate facts without class word)	Mandatory (including class word)	Mandatory (implemented as a Column, including class word)
	Description	Optional (define candidate facts)	Mandatory	Mandatory
	Alias	Optional	Mandatory**	Mandatory**
	Optionality	Optional	Mandatory	Mandatory
	Information Sensitivity and Privacy Classification (ISPC)	Optional	Mandatory**	Mandatory**
	Derived Fact Type	N/A	Mandatory**	Mandatory
	Aggregation Rule	N/A	Mandatory**	Mandatory**
	Formula	N/A	Mandatory**	Mandatory**
	Constraints	N/A	Mandatory**	Mandatory**
	Transformations	N/A	Mandatory**	Mandatory**
	Data Source	Optional	Mandatory	Mandatory
	Domain	Optional	Mandatory**	Mandatory**
	Data Type	Optional	Mandatory (not required if Domain is used)	Mandatory (not required if Domain is used)
	Size	Optional	Mandatory (not required if Domain is used)	Mandatory (not required if Domain is used)
	Format	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)

Object	Object Attribute	Fact and Dimension Matrix	Logical Dimensional Model	Physical Dimensional Model
	Units	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
	Valid Values	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
	Default Value	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
Dimension Entity or Dimension	Name	Mandatory	Mandatory	Mandatory (implemented as a Table)
	Short Name	N/A	Optional	Optional
	Plural Name	N/A	Optional	Optional
	Description	Mandatory	Mandatory	Mandatory
	Primary Key (Surrogate Key)	N/A	Mandatory	Mandatory
	Business Unique Identifier (Alternate Key, Natural Key)	N/A	Mandatory**	Mandatory**
	Alias (or Synonym)	Optional	Mandatory**	Mandatory**
	Data Steward	Optional	Mandatory	Mandatory
	Data Custodian	Optional	Mandatory	Mandatory
	Information Sensitivity and Privacy Classification (ISPC)	Optional	Mandatory	Mandatory
	Volume	N/A	Mandatory	Mandatory
	Volatility	N/A	Mandatory	Mandatory
	Retention Requirements	N/A	Mandatory	Mandatory
Dimension Attribute or Dimension Characteristics	Name	Optional (define representative attributes without class word)	Mandatory (including class word)	Mandatory (implemented as a Column, including class word)

Object	Object Attribute	Fact and Dimension Matrix	Logical Dimensional Model	Physical Dimensional Model
	Description	Optional (define representative attributes)	Mandatory	Mandatory
	Alias	Optional	Mandatory**	Mandatory**
	Slowly Changing Policy	Optional	Mandatory	Mandatory
	Optionality	Optional	Mandatory	Mandatory
	Information Sensitivity and Privacy Classification (ISPC)	Optional	Mandatory**	Mandatory**
	Data Source	Optional	Mandatory	Mandatory
	Domain	Optional	Mandatory**	Mandatory**
	Data Type	Optional	Mandatory (not required if Domain is used)	Mandatory (not required if Domain is used)
	Size	Optional	Mandatory (not required if Domain is used)	Mandatory (not required if Domain is used)
	Format	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
	Units	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
	Valid Values	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
	Default Value	Optional	Mandatory** (not required if Domain is used)	Mandatory** (not required if Domain is used)
Domain	Name	N/A	Mandatory	Mandatory (implemented as a Check Constraint)
	Description	N/A	Mandatory	Mandatory
	Data Type	N/A	Mandatory	Mandatory
	Size	N/A	Mandatory	Mandatory
	Format	N/A	Mandatory**	Mandatory**
	Units	N/A	Mandatory**	Mandatory**
	Valid Values	N/A	Mandatory**	Mandatory**

Object	Object Attribute	Fact and Dimension Matrix	Logical Dimensional Model	Physical Dimensional Model
	Default Value	N/A	Mandatory**	Mandatory**
Relationship	Name	N/A	Mandatory	Mandatory (implemented as a Foreign Key)
	Cardinality	N/A	Mandatory	Mandatory
	Optionality	N/A	Mandatory	Mandatory
Table Constraint (e.g. primary key, foreign key, unique key, check constraint)	Name	N/A	N/A	Mandatory
	Definition	N/A	N/A	Mandatory
Index	Name	N/A	N/A	Mandatory
	Definition	N/A	N/A	Mandatory
Non-Table Objects (Views, Sequences, Triggers, Procedures, etc)	Name	N/A	Optional	Mandatory**
	Definition	N/A	Optional	Mandatory**

4. Information Modeling Using XML Schema

4.1 Introduction to XML

XML (eXtensible Markup Language) is a plain text-based, platform independent markup language, which is readable by both humans and computers. These characteristics make XML a potential fit for universal information handling.

The OPS has been promoting XML as a standard for message-based business collaboration and data exchange, as the foundation and enabler for all web services, and to directly support the concept of Service Oriented Architecture (SOA).

The industry has overwhelmingly consolidated on a common approach to SOA, which is based on XML and web services, using Web Service Definition Language (WSDL) to describe the service interface and marking up the document payload in XML. The XML document payload is one part of the service that is described by an XML schema, either implicitly or explicitly.

XML schema is the metadata upon which the service is based as far as the data is concerned, and the WSDL is the metadata that controls the service as far as the technical implementation of a process is concerned. All other protocols in a SOA for security, transport, handshaking, and so on are XML-based. It is therefore highly recommended that metadata is captured and made accessible at a high level in XML schema format.

The World Wide Web Consortium (W3C) (http://www.w3.org.) specification for XML schema is comprehensive. More detailed information about XML Schema is available from the W3C site. The XML Schema Recommendations are divided into three parts:

1) An introduction to XML Schema concepts at www.w3.org/TR/xmlschema-0/
2) A definition of all of the structures used in XML Schemas at www.w3.org/TR/xmlschema-1/
3) A description of XML Schema data types at www.w3.org/TR/xmlschema-2/

The main objectives of this chapter are:

- To establish high-level best practices, guidelines and methods in XML schema design and implementation.
- To introduce information modeling as the approach to XML schema creation.

For the full list of the OPS standards for XML and XML-related specifications, see Government of Ontario Information Technology Standards No. 27 (GO-ITS 27), XML Family of Standards.

4.1.1 What is an XML Schema?

An XML schema is a set of rules that describe an XML document (called an XML instance document) by specifying:

- The data type of each element/attribute.
- The structure of instance documents (relationship between elements).

These rules are used by an XML parser to determine whether a given instance conforms to the particular vocabulary, and to thereby determine whether the XML instance is valid. In other words, an XML schema describes the structure of all possible XML documents and messages, each of which is a valid instance of that XML schema. Figure 4-1 shows an example of an XML schema and a valid instance of that XML schema.

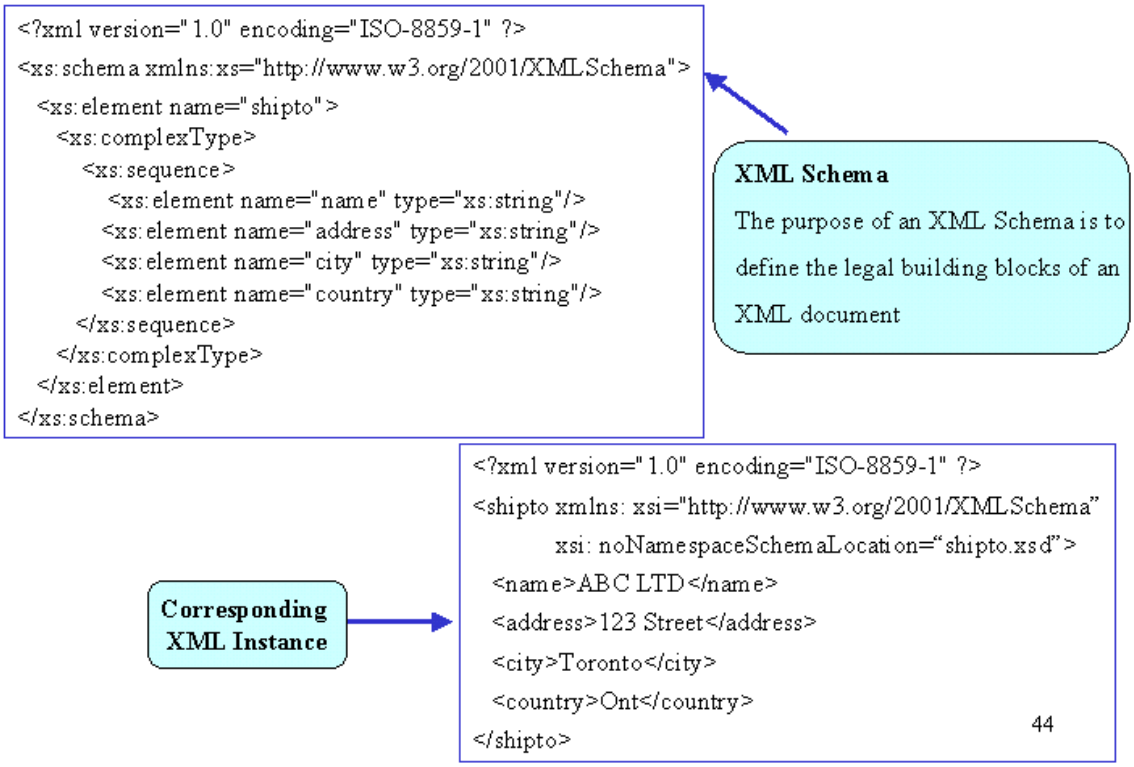


Figure 4-1: Sample XML Schema

An XML schema (as defined by the W3C XML Schema Recommendations www.w3.org/XML/Schema):

- Supports more than 44 data types.
- Allows creation of your own user-defined data types.
- Has familiar XML syntax.
- Models concepts of object inheritance and type substitutions.
- Supports complex and reusable content models.
- Supports namespaces.

4.1.2 XML Schema Basics

XML schema (or XML Schema Definition – XSD) is an XML-based language that defines the structure of information and associated semantics. Similar definitions in the relational database world would be database schemas, which define database tables and columns.

The general design principles for creating an XML schema are:

- Appropriate choice of names.
- Consistency in naming.
- Consistency in structure.
- Good documentation.
- Avoidance of unnecessary complexity.

As with database schemas which can be produced directly from a data model, XML schemas may also be produced from a data model. The model driven approach is described in Section 4.3.

XML schema supports the validation of XML documents:

- with or without any application specific codes;
- with the provision of default values for XML attributes and elements;
- with the creation of modular XML documents.

Previously the way to define the structure of information and associated semantics within an XML document was to use DTD (Document Text Definition). However, due to the advanced features and functionalities of XML schema over the DTD, the OPS standard is XML schema, (see GO-ITS 27).

4.1.3 XML Terminology

Table 4-1 provides a list of common XML schema components and descriptions. To assist readers who are not familiar with the terminology differences between XML concepts and data modeling concepts, a third column provides the equivalent meaning in data / object modeling terminology.

XML Schema Component	Component Description/Purpose	Equivalent Meaning in Data / Object Modeling Terminology
Element	A unit of XML data, delimited by XML tags, to represent and store data in a self-explanatory manner.	Entity or entity class attribute
Attribute	Used to provide additional information about an XML element. Therefore, an XML attribute represents metadata of the XML element.	Entity or entity class attribute
Annotation	A top-level element that captures schema comments and in-line	Definition, description, documentation, and note fields

XML Schema Component	Component Description/Purpose	Equivalent Meaning in Data / Object Modeling Terminology
	documentation. Each annotation element may have two sub-elements: documentation and appInfo.	within the logical and physical data models.
Documentation	A lower-level element that specifies information to be read by human readers. This element must go within an annotation element.	Definition, general description, documentation and note fields within the logical and physical data models.
AppInfo	A lower-level element that specifies information to be used by the application. This element must go within an annotation element.	Specific application and technical implementation notes and documentation within a physical data model.
Namespace	Used as a mechanism to avoid name conflicts (i.e. name collisions) of XML types, elements or attributes that have the same name but from multiple XML Schema definitions.	• Subject area in both logical and physical data models. • Tablespace in database definitions.
Target Namespace	The XML schema definition (xsd) file when it is created.	An assigned database tablespace
Default Namespace	The XML schema definition (xsd) file when it is created.	The default database tablespace
Import	Used to bring in definitions and declarations from one or many imported schemas with different target namespace(s) into the current schema.	Import data definitions from different subject areas of other data model files into the current model subject area.
Include	Used to bring in definitions and declarations from one or many included schemas with the same target namespace into the current schema.	Include data definitions of the same subject area from different data model files into the current model subject area.
Redefine	Used to bring in definitions and declarations from one or many external schema files with the same target namespace into the current schema and allow these brought in definitions and declarations to be modified at the same times.	Include data definitions of the same subject area from different data model files into the current model subject area with definition override option.
Primitive Data Type	A set of defined fundamental components that are used to create other, larger parts.	Primitive Data Type
Derived Data Type	Data types that are derived from the XML Schema primitive data types and serve as fundamental components that are used to create other, larger parts.	Built-in data type, or data type scalar function

XML Schema Component	Component Description/Purpose	Equivalent Meaning in Data / Object Modeling Terminology
Simple Type	An XML element that does not contain any other elements but data.	Data attribute domain based on a single primitive or derived data type. E.g. age, name, etc.
Complex Type	An XML element that may have other XML elements embedded within it.	• Entity or entity class. • Data attribute domain based on complex data types. For example, length that consists of a numeric reading for measure and a unit of measure.
User Defined Type	A customized data type that is based on or derived from any existing XML Schema simple or complex data type(s).	User defined data type profile
length, minLength, maxLength	A set of constraining facets used to specify constraints related to the length of XML elements or attributes.	Domain specifications on attribute sizes.
enumeration, pattern, whitespace	A set of constraining facets used to specify constraints related to the data values and patterns of XML elements or attributes.	Domain specifications on attribute data values.
maxInclusive, maxExclusive, minInclusive, maxInclusive	A set of constraining facets used to specify constraints related to the data value range of XML elements or attributes.	Domain specifications on attribute data value ranges.
minOccurs, maxOccurs	A way to specify the minimum and/or maximum number of times that an XML element is required to appear in a content model. If the value of this attribute is 0, then the element is optional.	• Multiplicity (in UML terminology) • Optionality and Cardinality (in ER modeling terminology)
Key or ID	Primary Key	Primary Key
Keyref or IDREF	Foreign Key	Foreign Key
Default	A way in XML schema to specify the default value.	Default
choice	Allows only one of the elements contained in the <choice> declaration to be present within the containing element.	Business rules related to the use of data elements expressed in a CASE statement or an IF ELSEIF statement.
Derived Type by Extension	XML schemas allow the derivation of new data types from some existing	Establish supertype and subtypes relationship through

XML Schema Component	Component Description/Purpose	Equivalent Meaning in Data / Object Modeling Terminology
	base types by extensions.	inheritance (in UML), or specialization of the generalization relationship (in both UML and ER).
Derived Type by Restriction	XML schemas allow the derivation of new data types from some existing base types by restrictions.	Apply data constraint rules to data attributes to restrict their values.
elementForm Default, attributeForm Default	XML schema attributes specified in the schema definition file to control if all schema instance files must qualify the namespace of all elements and/or attributes.	A switch that allows the specification of database name and table name as parts to fully qualify the names of table columns, and stored procedures.

Table 4-1: XML Schema Components and its Data/Object Modeling Equivalence

4.1.4 XML Schema Utilization

XML schema is used for the following:

- **Data Validation**

  One of the most common uses for schemas is to verify that an XML document is valid according to a predefined set of rules. A schema can be used to validate:

  - The structures of elements and attributes.
  - The order of elements.
  - The data values of attributes and elements, based on ranges, enumerations, and matching patterns.
  - The uniqueness of values in an instance. For example, health card numbers in all health card instances must be unique.

  Validating a document against schema requires the use of a special parser. The XML Schema Recommendation calls them schema validators.

  The validation can be accelerated if the information contained in an XML schema is rewritten using some efficient programming language (e.g. Java) constructs in terms of that language (e.g. an XML Schema element becomes a Java Object and an XML schema attribute becomes a Java Object's attribute). This procedure is called *binding*.

- **A Contract Specification With Trading Partners**

  Often, XML instances are passed between organizations. An XML schema may act as a specification of contracts with a trading partner. It clearly lays out the rules for contract document structure and what is required. Since an instance can be validated against an XML schema, the "contract" can be enforced using available tools.

- **System Documentation**

  XML schemas can document the data in an XML instance. Anyone who needs to understand the data can refer to the XML schema for information about names, structures, and data types of the items. Annotations can be added to any schema component to provide further documentation.

- **Augmentation of Data**

  Schema processing can also be applied at the instance level. It inserts default and fixed values for elements and attributes, determines subsequent data requirements based on the value of an element or attribute, and normalizes whitespace according to the data type.

- **Application Information**

  Schemas provide a way for additional information about the data to be supplied to the application when processing a particular type of document.

  This information in schemas can be used to generate code such as:

  - ***User interfaces for editing the information*.** For example, if you know that size is between 2 and 18, you can generate an interface that has a slider bar with these values as the limits.
  - ***Stylesheets to transform the instance data into a reader-friendly representation such as XHTML***. For example, if you know that the human-

readable name for the content of a number element is “Health Card Number” you can use this as a column header.

- ***Code to insert or extract the data from a database*.** For example, if you know that the health card number maps to the HCARD_NUM column on the HCARD table, you can generate an efficient routine to insert it into that column.

4.2 XML Schema Components

This section is an overview of the components, basic structure and features of XML schema.

An XML schema may consist of the following components:

- One or many XML schema namespaces,
- Zero or one target namespace,
- A default namespace,
- One or many primitive type definitions and derived type definitions,
- Zero, one, or many user defined simple or complex data types,
- One or many XML elements and attributes.

4.2.1 Namespace

The concept of XML namespace is introduced in XML schema design as a method to avoid name conflicts (i.e. name collisions) of XML types, elements or attributes from multiple XML schema definitions. XML namespaces are used to qualify and uniquely identify an XML element, attribute, or type of one XML schema definition from other element, attribute, or type with the same name but from different XML schema definitions.

An XML namespace label is always presented before the name of an element, attribute or type it qualifies and is separated from the name by the “:” sign. As shown in the example below, the namespace label “xsd” represents a namespace that qualifies the name of both element and type definitions:

```
<xsd:element name="Forename" type= "xsd:string"/>
```

4.2.1.1 Target Namespace

Each XML schema definition file should define a target namespace. This target namespace should be defined as a URL that uniquely qualifies the schema and all XML elements, groups of elements, attributes, group of attributes, and type definitions defined within the schema.

Each OPS organization such as a ministry, agency, cluster, or external service provider should have its own namespace. For ministries or agencies that have multiple applications or provide multiple services, each such application or service may have its own specific application schema. In this case, this specific application schema should have its own namespace. This provides a standard way to avoid name collisions between schemas that

may be embedded in other XML schema definitions. The namespace string will be a URL, and will be constructed by means of a hierarchical organization that corresponds to the owner group within the OPS organization structure.

The root URL string will be http://ns.orgName.org, where orgName represents the name of an organization, such as an OPS I&IT cluster, ministry or agency. The root URL string is followed by one or more level of working group qualifiers or project qualifiers (if necessary), a version specification, and a schema qualifier. The version specification will be defined in section 4.2.1.4.

For example:

targetNamespace=”http://ns.mgs.gov.on.ca.org/cdes/v2.0.0/address/”

targetNamespace=”http://ns.mto.gov.on.ca.org/esdi/v1.0.1/addresschange/”

The namespace URL may provide a document that points to the schema and specifications locations, but this is not guaranteed.

4.2.1.2 Namespace Design Style

A schema designer will usually face the following design decision challenges when a project needs to create many XML schemas: “Should I define one target namespace for all the schemas, or should I create one target namespaces for each schema, or should I have some schemas with no target namespace?”

XML namespaces can be defined within XML schema definition (.xsd) files in one of the following three ways:

- Homogeneous Namespace Design
- Chameleon Namespace Design
- Heterogeneous Namespace Design

Each namespace design style and its characteristics are fully explained in the supplementary document “Guide to XML Schema Design Styles and Transformation”.

Among the namespace design styles listed above, **Heterogeneous Namespace Design is the preferred design style for defining XML schema target namespaces** as it offers the flexibility and capability that allow each OPS organization and project to define its own specific business concepts while in the meantime to utilize and re-use some common definitions and data patterns.

4.2.1.3 Default Namespace

Each XML schema definition file should define a default namespace that is equal to the target namespace. In other words, the XML schema namespace should NOT be the default namespace.

Example:

xmlns=”http://ns.mgs.gov.on.ca.org/cdes/v2.0.0/address/”

The namespace prefix for the XML Schema should be **xsd** within each XML Schema definition file. For example:

xmlns:xsd=”http://www.w3.org/2001/XMLSchema”

4.2.1.4 Version of Namespace

The default and target namespaces defined in the XML schema definition file must include a version identification value. The value is a string composed the character **v** followed by the **version number**, **release number**, and **patch number**, with each number delimited by the symbol "**.**". Whenever a schema has a new version, a new release, or a new patch, the corresponding version number, release number, or patch number must be incremented accordingly.

The initial version of a schema will be version **v1.0.0**. For example:

targetNamespace = http://ns.mgs.gov.on.ca.org/cdes/v1.0.0/address/

For each new release, the initial patch number will be zero. For example:

targetNamespace = http://ns.mgs.gov.on.ca.org/cdes/v1.8.0/address/

Similarly, for each new version, the initial release number and patch number will start at zero. For example:

targetNamespace = http://ns.mgs.gov.on.ca.org/cdes/v2.0.0/address/

4.2.1.5 XML Fragment Example

The following XML fragment incorporates all of the guidelines discussed in section 4.2.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://ns.mgs.gov.on.ca.org/cdes/v1.0.0/address/"
  xmlns="http://ns.mgs.gov.on.ca.org/cdes/v1.0.0/address/"
  xmlns:xsd="http://www,w3.org/2001/XMLSchema"
          …
>
```

4.2.2 Qualified and Unqualified Schema Settings

When a schema definition file is created, the following two "switch" attributes of the schema element should be specified:

- ElementFormDefault
- attributeFormDefault

Although these two attributes are specified in the schema definition file, the impact of these settings is not obvious until an XML instance document is constructed.

When the **elementFormDefault** "switch" is set to **qualified,** all the elements defined in a global or local namespace in the XML instance files must be namespace qualified. When the **elementFormDefault** "switch" is set to **unqualified,** only the elements defined in the global namespace must be namespace qualified.

Industry best practices recommend using the first option where the elementFormDefault value is set to qualified. By fully qualifying the namespace, it is more obvious and clearer which element is from which schema.

Qualifying the namespace can also improve performance. Often, when processing an instance document, the namespace is required to determine how an element should be processed. If the namespace is hidden (or unqualified), then the application must look up the element in the schema definition file for the element, which results in slower performance.

The attributeFormDefault "switch" should always be set to unqualified, as recommended by industry best practices.

4.2.3 Element and Attribute Usages

In XML, both XML elements and attributes can be used to store information or represent some semantics in a self-explanatory manner. Making the data "self-explanatory" means that information should be contained in elements, each of which will start with an opening tag <element> and end with a closing tag </element>. All information that belongs to an element must be contained between the opening and closing tags of an element. For example,

```
<ElementName>Language Name string</ElementName>
```

Attributes are used to specify additional information on top of the data that falls between the opening and closing tags. It may help to think of attributes as a means of specializing generic elements to fit one's specific needs. An attribute for an element appears within the opening tag.

If there are multiple values an attribute may have, the value of the attribute must be specified. For example, the forename element has a namePart attribute whose values are: first, and middle. The syntax for including an attribute in an element is:

```
<ElementName attributeName="value">
```

In general, it is recommended to use elements for data that will be produced or consumed by an application and to use attributes for metadata. A good rule of thumb is to use elements for *nouns* and attributes for *adjectives.* For example, the information representing a person's language skills can be described in XML as follow:

```
<LanguageSkills canSpeak="Yes" canWrite="Yes"
    canRead="Yes">English</LanguageSkills>
```

In the example above, the only element represents the name of the language, while attributes are used to describe the proficiency of skill set with the Language. In the early days of Web services, there was a push to avoid attributes in schema definitions of messaging interfaces because of SOAP encoding issues. However, OPS have adopted a literal encoding[1] approach (doc-literal) that alleviates this issue.

[1] Literal encoding is also endorsed by Web Services Interoperability (WS-I).

4.2.3.1 Element Declarations

To declare an element:

1) Specify the element's name.
2) Define the allowable content, which is determined by the element's type (simple or complex).
3) Declare whether it is of local scope or global scope:
   - Local declarations are only valid in their specific context.
   - Global declarations can be reused throughout the XML schema instead of duplicating the same local declaration multiple times.

1) May optionally specify:
   - Cardinality- the minimum (0) and maximum (unbounded) occurrences of a specific element within a content model.
   - Default and fixed values for attributes and elements.

Elements can be declared and arranged in one of following 4 ways:

1) Sequence (elements must appear in the exact sequence, or order as specified)
2) Choice (specifies multiple declarations, only one of which may be used)
3) Group (refers to elements, which are grouped and reused together)
4) All (declares that elements within our content model may appear in any order)

4.2.3.2 Attribute Declarations

Attribute declarations are similar to element declarations. They can be of local or global type, be grouped together, and have default and fixed value.

4.2.4 Data Types in XML

The following data types are allowed in an XML schema:

- Built-in data types (string, byte, integer, time, date, Name, any URI, language, ENTITY, NOTATION…) There are 44 built-in data types in total.
- User defined data types (which are based on existing data type(s))

There are 2 broad categories of data types:

1) **Complex type** - A data type that may contain attributes or other elements.
2) **Simple type** - A data type that contains only text content. The text can be of many different types. It can be one of the primitive data types that are included in the XML schema definition (boolean, string, date, number, etc.), or it can be a user-defined type such as MinistryNameType, MinstryCodeType, etc. Restrictions (facets) can be added to a data type in order to limit its content, size, value range, or value sets, and you can even specify data matching patterns.

A user-defined data type can also be defined as named type or anonymous type:

- A named type is a *globally declared* data type that has a **name** attribute. Therefore a named type can be referenced anywhere within the XML Schema definition.
- An *anonymous type* is a data type that is nested in-line within the definition of an element. An anonymous type can usually be used when a type is only needed once within a schema.

As a general rule, data types should always be defined in the global namespace and then referenced by the local elements. This approach is supported by both the Venetian Blind and Garden of Eden Design styles. **Consequently, the use of named types is recommended over the use of anonymous types because the named types support the concept of reuse in the schema design**.

4.2.4.1 Simple Types

When appropriate, simple types should be used (and potentially restricted or extended) rather than creating a user defined complex data type. Restriction of a simple type reduces the possible values of the type while extension allows one to create a complex type with simple content that has attributes.

For example:

If a date value is needed, use

```
<xsd:element name="Date" type="xsd:date"/>
```

instead of

```
<xsd:complexType name="Date">
    <xsd:sequence>
        <xsd:element name="Month" type="xsd:integer"/>
        <xsd:element name="Day"   type="xsd:integer"/>
        <xsd:element name="Year"  type="xsd:integer"/>
    </xsd:sequence>
</xsd:complexType>
```

The correct simple type defined in the XML schema data model should also be used.

For example:

If a data value is needed, use

```
<xsd:element name="Date" type="xsd:date"/>
```

instead of

```
<xsd:element name="Date" type="xsd:string"/>
```

Every simple type is derived from another data type. Therefore, a simple type can also be called derived type. The type from which a simple type is derived is called a base type.

There are three derived types:

1) **Restriction type** - A derived type declared using restriction is a subset of its base type.

   Facet(s) is used to restrict a type. There are several constraining facets defined in XML schema.

   For example, the built-in numeric type *nonNegativeInteger* was created by setting the Integer's facet *minInclusive* to zero.

2) **List type** - A derived type that can be any type from the whitespace-separated list of other data types.
3) **Union type** - A union type occurs when different data types are combined into one type. Complex type definitions are used to declare sophisticated content models, which specify an element's allowable content.

4.2.4.2 Complex Types

Complex types should only be extended but not restricted. Extension involves adding extra attributes or elements to a derived type. Derivation by restriction of complex types should be avoided because the WXS specification is complex in this area and implementations are error prone.

For example:

Base Type

```
<xsd:complexType name="BaseAddress">
    <xsd:sequence>
        <xsd:element name="State" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

Derived Type

```
<xsd:complexType name="NewAddress">
    <xsd:extension base="BaseAddress">
        <xsd:sequence>
            <xsd:element name="City" type="xsd:string"/>
        </xsd:sequence>
    </xsd:extension>
</xsd:complexType>
```

4.2.5 Inheritance

The inheritance feature allows one to model supertype/subtype relations similar to the object oriented paradigm.

There are 2 primary models for creating inherited types:

1) **Extension** - The new types, derived by extension mechanism from the existing ones, will contain all of the original type information and will add new information.
2) **Restriction** - When restricting a data type, the base type is used to create the derivation by removing elements or attributes of a complex type or by constraining facets of a simple type.

An XML schema allows using inherited types in place of base types within the XML schema.

Using one type in place of the other is called *substitution*. Both type and element substitutions are possible.

4.2.6 Combining Definitions from Multiple Schema Documents

Instead of creating a single schema document, which can be difficult to use and update, XML schema Recommendation provides 3 mechanisms for combining and reusing separate XML schemas:

1) **Import element** - Allows the reuse of global declarations from other external XML schemas by *referring* to them.
2) **Include element** - Allows the reuse of global declarations from other external XML schemas by *including* them.

   The include mechanism only works on XML schemas designed using the Homogeneous or Chameleon Namespace Design approaches.

   The difference between the include element and the import element is that the import element allows references to schema components from schema documents with different target namespaces and the include element adds the schema components from other schema documents that have the same target namespace (or no specified target namespace) to the containing schema. In short, the import element allows you to use schema components from any schema; the include element allows you to add all the components of an included schema to the containing schema.
3) **Redefine element** - Allows the inclusion of a global declaration of simple type, complex type, group, or attribute group from another external XML schema and the modification of it in some way at the same time.

   The redefine mechanism only works on XML schemas designed using the Homogeneous Namespace Design approach.

4.2.7 Documenting XML Schemas

The W3C XML Schema Recommendations (www.w3.org/XML/Schema) provides several mechanisms to support the documenting of code:

- **Comments** - Since many XML processors will not support XML style comments, the use of standard XML style comments should be avoided even though this mechanism is useful for human readers of the XML schemas.
- **Attributes from other namespaces**
- ***Annotations*** - The primary documenting features introduced in the XML Schema Recommendation. It allows for the documentation of general information, as well as additional application specific information.

A standard best practice is to document the schema definition file content, elements, attributes, simple types, and complex types defined in the XML schema definition file. For example:

```
<xsd:simpleType name="SurnameType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">This is a simple type that defines the surname name of the Name
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value= "1"/>
        <xsd:maxLength value="35"/>
    </xsd:restriction>
</xsd:simpleType>
```


4.3 Model Driven Approach to XML Schema Design

This section briefly describes how to apply a data model driven approach in the design and development of XML schemas for persistent data.

The data model driven approach starts with a logical data model (LDM), which is developed from a conceptual data model (CDM). The LDM is then transformed into an XML schema physical data model (PDM) expressed in either a UML class diagram notation or a tree structure diagram notation. Finally, the XML schema PDM is used to generate XML schema definitions.

The LDM used as the starting point should be a normalized data structure with the Many-to-Many relationships resolved. The LDM constructs include entities (in ER modeling terminology) or entity classes (in UML class modeling terminology), primary key attributes, foreign key attributes, non-key attributes, domains for data values, optionalities and cardinalities (in ER modeling terminology) or multiplicities (in UML class modeling terminology). Since XML supports some object oriented (OO) characteristics such as inheritance, experience has shown that super-type/subtype structures can be left unresolved by referring back to the corresponding conceptual data model (CDM), or higher level analysis model. However, the use of more than two subtype levels is not recommended.

More details on how to apply the data model driven approach in the design and development of XML schemas can be found in the supplementary document “Guide to XML Schema Design Styles and Transformation”.

4.4 Transformation and Alignment of Data Models and XML Schemas in the OPS

This section briefly describes the transformation, generation and alignment of data models and XML schemas used in the OPS.

Figure 4-2 provides an overall picture of transformation and alignment between schema physical data models and XML schemas in the OPS.

The OPS is hierarchically organized. The highest level is OPS Corporate, under which are clusters and ministries. Each cluster, ministry or even corporate could have its own projects. Each organization could have its own CDM, LDM and XML schemas. LDMs are derived from CDMs, and LDMs could be used as a means of transforming into XML schemas. CDMs, LDMs and XML schemas should be aligned across the entire OPS. In Figure 4-2, horizontal arrows labeled with "A*" are for alignment and vertical arrows with "T*" are for transformation or generation.

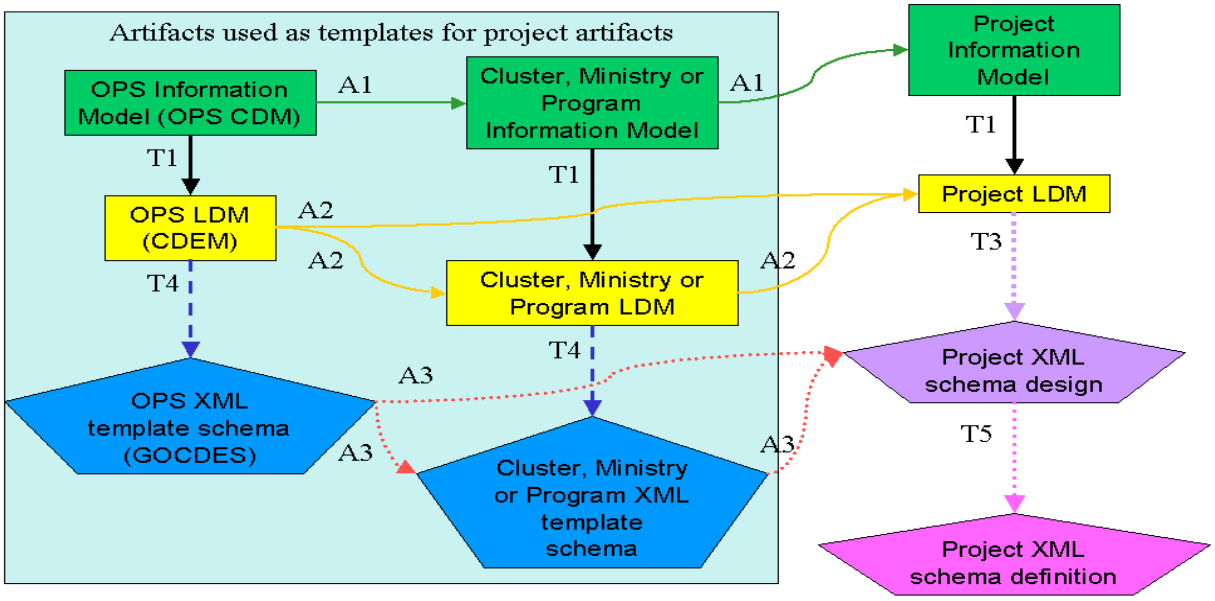


Figure 4-2: Transformation and Alignment of Data Models and XML Schemas in the OPS

Two types of schemas are supported:

1) **Architecture schemas** - Define common data types and elements for re-use elsewhere, like the OPS Common Data Elements XML Schemas (CDES) and cluster/ministry common schemas depicted in Figure 4-2.
2) **Message schemas**- Inherit architecture schemas and are used for application specific purposes such as message requests or responses. Most of the project XML schemas in Figure 4-2 belong to this category.

Figure 4-3 shows the levels of XML schemas in the OPS and their inherent structure.

This section only discusses the transformation from an LDM to XML schemas (T3 and T4 in Figure 4-2). T3 is the transformation from an LDM to a message schema and T4 is the transformation from an LDM to an architecture schema.

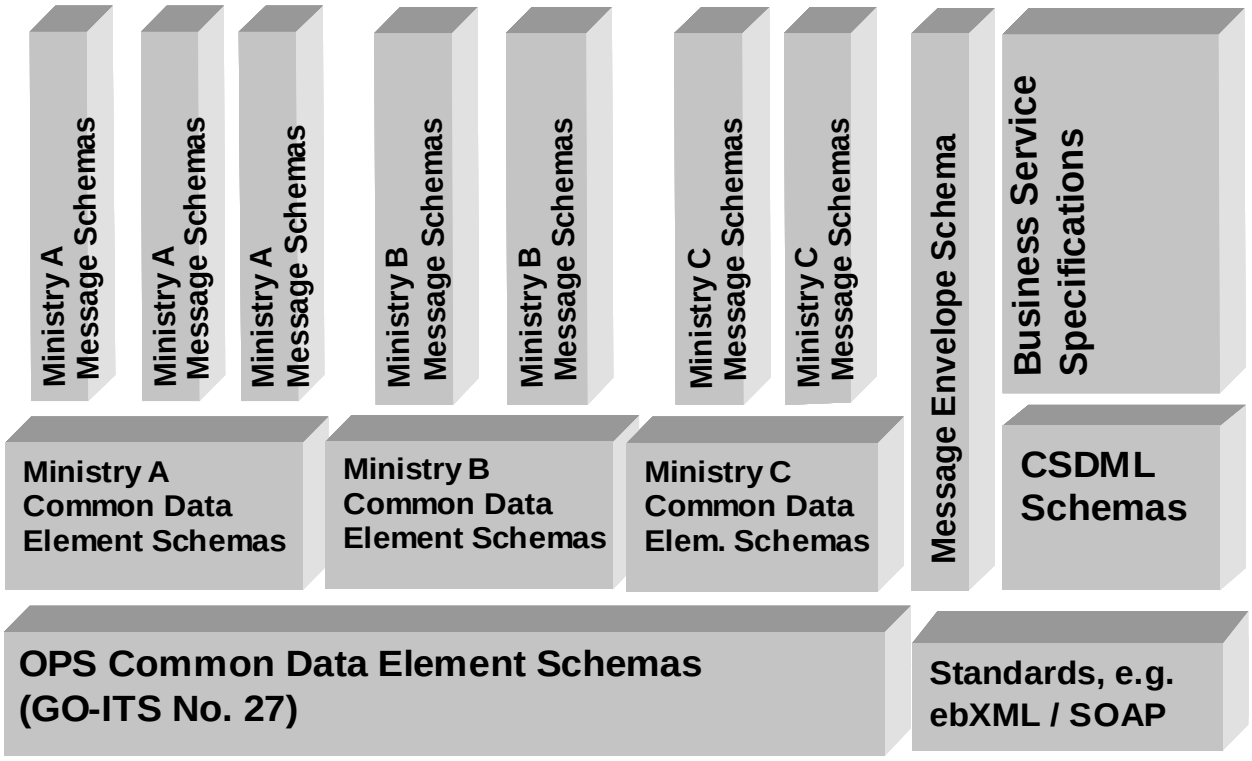


Figure 4-3: Example of Multiple Levels of XML Schemas in the OPS

Selection of an Appropriate XML Schema Design Style

Select a general schema design style that is appropriate to the intended usage of the schema. Please refer to the supplementary document "XML Schema Design Styles & Transformation" for a more detailed explanation of some popular schema design styles.

Architectural Schemas

Since type definitions are preferred over element definitions in architectural schemas, the Venetian Blind or Garden of Eden Design styles are the preferred style for those elements that will be globally available. Within this, some elements might be defined in other styles.

Message Schemas

A key aspect of message schemas is that they are easily readable and maintainable. (If a schema is generated from a metadata repository or some other database, maintainability of the final schema is no longer an issue. However, it still needs to be readable.)

Experience has shown that the Russian Doll Design style provides good readability since the format of the schema mirrors the format of the instance messages. It also has the benefit of simple scoping in which all definitions that are only used once are declared locally. This means that the same name can be re-used within the schema, leading to simpler element and attribute names.

In general, this model is preferred, although there is an argument for making more complex definitions global to simplify the layout of the schema. Although the element itself can be made global, defining a global complex data type instead retains the benefits of local scope of the element name.

4.5 OPS Common XML Schema Usage Rules

This section outlines some usage rules of the OPS XML Common Data Elements Schemas (CDES), which are a set of XML schema templates approved as GO-ITS 27.

The usage rules provided below indicate the mandatory and optional parts of the CDES standard, as well as recommendations on how to use the OPS CDES in OPS applications. Further usage rules are specified in the design documents for each specific subject area of the CDES.

OPS XML Schemas (OPSS) are schemas defined for projects and applications, based on the OPS CDES.

1) OPS XML Schemas (OPSS) that utilize concepts defined in the OPS CDES must use CDES schema components. The use of OPS CDES schema components in an OPSS is essential for standardizing and sharing common data element specifications across all OPS applications, and particularly, across shared transactional message schemas.
2) Use of the OPS CDES schema elements is recommended when both an element and a type (simple type or complex type) are defined for the same concept in the OPS CDES. That is because the use of the element not only guarantees the standardization of the data type structure, but also achieves the standardization of names. For example, use of the element CivicAddress should be considered first before the use of complex type CivicAddressType.
3) An OPSS may introduce new elements that make reference to the OPS CDES components when:
   a) The OPSS uses multiple references to the same OPS CDES component.
   b) The name of an OPSS element needs to express the application specific business meaning of the OPS CDES component being referenced.

   For example, an OPSS could introduce elements OldPhoneNumber and NewPhoneNumber that both make reference to the OPS CDES type TelephoneAddress.
4) If an OPS application requires extension or redefinition of an OPS CDES element or type, the following options should be considered (*listed in order of preference*):
   a) Extend an OPS CDES type. For example, extend Non Civic Physical Address to model Non–Ontario Physical Addresses.
   b) Extend a component that is referenced from an OPS CDES subtype. For example: Building Unit, which is referenced from CivicAddressType and RouteAddressType.
   c) Introduce a new schema element or type (simple type or complex type) by making reference to the existing OPS CDES components. For example, introduce a new schema element or type with a CHOICE construct that uses OPS CDES types.
   d) Add, remove or change data integrity rules.
   e) Redefine an OPS CDES schema element or type.
   f) Introduce a new component that does not make reference to any existing OPS CDES component.

   The above changes can be done only if an XML schema component allows extension (indicated in its specification). The new schema components should be defined within

an OPSS namespace that contains only OPSS extensions to the OPS CDES components.

5) Should an OPS application opt to use a JAXB tool that does not support some constructs of the W3C XML Schema specification then OPS CDES components may be modified to meet the restrictions of the selected JAXB tool. However, these modifications will be justified only if the semantics of the changed OPS CDES components remains the same.

5. Data Naming Standards

Data naming standards are one of the most important corporate standards that promote shared data. It would be impossible to achieve the goal of sharing information without data naming standards.

The naming standards introduced in this chapter apply to following data object types:

- **Conceptual and logical data objects** that describe the information. Examples are entities, classes, attributes, relationships, operations, and XML elements and attributes.
- **Physical data objects** that implement the logical objects. Physical objects should conform to the conceptual and logical standards except where restricted by product limitation. Examples are physical database objects, and XML schema elements.

5.1 Standardization of Data Elements to Conform to ISO / IEC 11179

5.1.1 Naming Conventions

The naming conventions follow the guidelines set out in ISO / IEC FDIS 11179-1/5. Data element names may consist of the following four types of component (the first three appear in order):

- Object Class Term
- Property Term
- Representation Term

Each of these may be preceded by an associated:

- Qualifier Term

Object Class Term: A component of a data element name which represents a set of ideas, abstractions or things in the real world, which has been represented in the data model as a data entity.

For example:

Warrant	in	Warrant Reason Description
Statute	in	Statute Short Title

Property Term: A component of a data element name which expresses a property of an object class.

For example:

Reason	in	Warrant Reason Description

Title	in	Statute Short Title

Representation Term: A component of a data element name which describes how its values are represented.

For example:

Description	in	Warrant Reason Description
Title	in	Statute Short Title

A controlled list of Representation Terms is maintained, such as Name, Date, Amount, etc.

The Representation Term may already be present in the Property Term. When this happens, one occurrence of the word is dropped to avoid duplication, as in the case of Title in the example above. Representation Terms map to a corresponding Domain.

For example:

Title	maps to	the Name domain

Qualifier Term: A component of a data element name which helps to make the data element name more specific. If necessary to uniquely identify a data element, Qualifier Terms may be attached to Object Class Terms, Property Terms, and Representation Terms.

For example:

Short	in	Statute Short Title
Long	in	Statute Long Title

5.1.2 Naming Convention Rules

The naming convention rules are based on Annex A of ISO / IEC FDIS 11179-5:1995.

Semantic Rules

- One and only one Object Class Term shall be present.
- One and only one Property Term shall be present.
- One and only one Representation Term shall be used.
- Qualifier Terms may be added as needed to describe the data element and make it unique within a specified context.
- Terms may be composed of more than one word.

Syntactic Rules

- The Object Class Term shall occupy the first position.
- The Property Term shall follow the Object Class Term and precede the Representation Term. If any word in the Property Term is already present in the Object Class Term, it should be removed.

- The Representation Term shall occupy the last position. If any word in the Representation Term is already present in the Property Term, it should be removed.
- Qualifier Terms shall precede the component qualified, which may be an Object Class Term, Property Term or Representation Term. The order of qualifiers shall not be used to differentiate data elements.

Lexical Rules

- Nouns and adjectives shall be in the singular form only.
- Name components and words in multi-word terms shall be separated by spaces.
- No special characters may be used. All words shall begin with a capital letter with the remaining letters in lower case.
- Acronyms may be permitted if they are widely used and universally understood in the business community.
- Abbreviations are not permitted.

Uniqueness Rule

- All data element names shall be unique within the dictionary.

Domain Rules

- Data elements using a particular Representation Term must be consistently associated with the same Domain.
- More than one Representation Term may map to the same Domain.

5.1.3 Definition Rules

Data element definitions shall:

- State the essential meaning of the concept precisely, unambiguously, and concisely.
- Match the data element name in terms of the implied scope of the concept.
- Avoid circularity by not using the components of the data element name.
- Use the same terminology and structure for related data elements.
- Not embed the definitions of other data elements.
- Not include rationale, functional usage, procedural, or domain information.
- Be unique within the dictionary.
- Be stated in the singular.
- Include explanations for any acronym used.

5.2 Purpose of Data Naming Standards

The purpose of data naming standards is to:

- Promote data sharing.
- Determine the “language” that will be spoken by the business and developers. The names communicate facts about the business in the data model.
- Enable the integration of the enterprise data models to the OPS Enterprise Architecture (EA).
- Enable the integration of project and application data models to the Enterprise Information Architecture.
- Facilitate the link between the logical and physical view. If a logical name is known, the physical name should be easily derivable.

These conventions are developed to ensure that the often disparate subject areas found within and across application specific data models have a consistent “look and feel”. It will enable users of the application specific and actual enterprise model to understand and reuse the contents with a high degree of accuracy. In addition, they provide a helpful framework for the development of new models. Therefore, a naming standard is of paramount importance. Although no naming standard is perfect, it is essential to find and implement a method that is compatible with the goals of the enterprise.

5.3 Data Naming Principles

The data naming conventions and standards should adhere to the following principles:

- Be precise and meaningful.
- Reflect the business concept, not just a concept of particular information system on a specific information technology platform.
- Be practical.
- Be self-documenting. By looking at the name, the reader should have a good idea what the name means without having to read the description. The name will have been derived from the business use or purpose.
- Be unambiguous. Different people from different areas reading the name at different times must have the same understanding of what the name means.
- Fit within the designated repository and data modeling tools.
- Allow for the implementation of software packages within the systems development project.

5.4 General Naming Standards

The data model is an abstraction of the business. As a general guideline, the data model elements (entities, attributes, relationships, etc) and XML schema tags should use English-language, full, and descriptive names that accurately convey the meaning of the associated business concepts.

Names should:

- Be meaningful.
- Use no special characters (/ * # $).
- Not begin with a numeric character.
- Not use abbreviation algorithms. These are methods of abbreviating names of entity (or entity class), attribute, and XML tags by applying a specific method such as “for names over six characters use the first letter, remove all following vowels and remove trailing ‘s’, ‘ing’, and ‘ed’”.
- Abbreviations and short forms may be used where product ‘naming’ limitations prohibit the use of a full word. A list of suggested abbreviations can be found in Guideline for Abbreviating Data Object Names.
- Be defined in the following way:
  - Each model element within the model element hierarchy (i.e. entity/entity class, attribute and data value in a conceptual or logical data model, owner, table, column and data value in a physical database model, and type, element/attribute and tag values in an XML schemas) should have its own appropriate name.
  - The name at each level of data model should be defined within the level above it.
  - Prefixes should not be used to qualify an entity or attribute name. This includes the use of application acronyms in table names and use of table name acronyms in column names.
  - Acronyms and abbreviated words may be used in entity and class names where product ‘naming’ limitations prohibit the use of a full word.

5.5 Conceptual & Logical Data Naming Standards

5.5.1 Entities

Entities will be named using the general naming standards provided in section 5.4 and conform to ISO / IEC 11179-5. In addition, entity names should have the following characteristics:

- Must be unique across all cluster/ministry models. If a chosen name already exists, either the two entities should be combined or one of them should be renamed.
- Must use business terminology. Where different groups of users have different terminology, select the more widely used or accepted term to name the entity. Include any synonyms to that terminology in the entity definition.
- Must be meaningful. The name should reflect the thing or concept that the entity represents, rather than any medium on which it is handled.
- May contain acronyms or abbreviations, but only when these acronyms or abbreviations are fully recognizable by the business communities. A list of suggested abbreviations can be found in Guideline for Abbreviating Data Object Names.
- Must contain only singular nouns (for example: Party, Organization).
- Must use mixed case or uppercase characters. If using mixed case then capitalize the first letter of each word in the entity name (for example: Party Role).
- Must use a space as the delimiter between the words in the name (for example: Party, Party Role, Party Type).
- Must use only alphanumeric characters.

5.5.2 Attributes

Attributes will be named using the general naming standards provided in section 5.4. In addition, attribute names:

- Must be unique within its parent entity.
- Must use business terminology.
- May contain acronyms or abbreviations, but only when these acronyms or abbreviations are fully recognizable by the business communities. A list of suggested abbreviations can be found in Guideline for Abbreviating Data Object Names.
- Must contain only singular nouns (for example: First Name, Last Name).

- Must use mixed case or uppercase characters. If using mixed case then capitalize the first letter of each word in the attribute name (for example: First Name, Last Name).
- Must use a space as the delimiter between the words in the name (for example: First Name, Last Name, Birth Date).
- Must use only alphanumeric characters.
- Must end with a class word that is used to indicate the type of data represented or stored. Use the list of class words provided in Appendix B: Class Words.

5.5.3 Entity Classes

Entity Classes will be named using the same naming standards provided in section 5.5.1 Entities.

5.5.4 Attributes for Entity Classes

Entity Class Attributes will be named using the same naming standards provided in section 5.5.2 Attributes.

5.5.5 Domains

Domains will be named using the same naming standards as provided in section 5.5.2 Attributes.

5.5.6 Relationships

The relationship between two entities is directional. Sometimes, the relationship is bi-directional. Both directions of each relationship must be named separately. The relationship should be named with a clear, concise phrase that is meaningful to the business user and should use the following rules:

- Must use only lowercase characters (for example: belongs to).
- Must use a space as the delimiter between the words in the phrase (for example: consists of).
- Must use only alphanumeric characters.
- Must be a maximum of 30 characters in length.
- Note: Naming supertype / subtype relationships is optional.

Naming relationship is often difficult because it isn't always easy to come up with meaningful names; some guidelines that may help are listed below.

- Use active, rather than passive, verbs that describe the relationship.
- Be meaningful. The name should convey the business rule that exists between the entities; it should provide a clue for devising a good name. Names like

"related to" or "associated with" do not add enough meaning to the name to be useful when communicating the relationship to others.

- Start with the entity that is most likely to take some action. Suppose we are considering the relationship between courses and students. While "One Course may contain one or more Students", the relationship of "One Student must be contained by one or more Classes" is tenuous. A much more meaningful name for the relationship is derived from the student end: "One Student must take one or more Course", and "One Course may be taken by one or more Students".
- Try naming the active relationship direction first. For most relationships, there is an active and a passive direction for the relationship; "One Instructor may teach one or more Courses" is the active direction whereas "One Course must be taught by one and only one Instructor" is the passive. It is usually easier to derive a meaningful relationship name by starting with the active side.
- Eliminate "be" verbs from the name. Words like "is" are implied and should not be shown in relationship names.

5.5.7 Associations for Entity Classes

An Association is a connection between two classes (or itself) in a UML class model. To name an association, use the same naming standards provided in section 5.5.6 Relationships.

A Generalization in UML notation plays the same role as a Supertype / Subtype relationship. Use the same naming standards provided in section 5.5.6 Relationships.

5.6 Physical Data Naming Standards

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

5.6.1 Tables

Tables are named using the general naming standards provided in section 5.4. In addition, table names:

- Must be unique within the database schema. When it is corporate data it should be unique within the cluster/ministry.
- Must have a unique short name. The short name is used in conjunction with the physical naming standards to name database objects such as foreign keys,

packages, and triggers. Refer to Appendix A1 for a suggested name abbreviation algorithm.

- Must use business terminology, as much as it can be accommodated in the target environment.
- Must be named the same as the logical (entity/class) name, subject to DBMS naming limitations (for example, length, reserved words).
- Must be singular.
- Must be in uppercase or mixed case, subject to the supportability of database management software used. If using mixed case then capitalize the first letter of each word in the table name (for example: Party_Role).
- Must use an underscore (_) as the delimiter between the words in the name to be consistent with traditionally accepted SQL syntax (for example: DEVICE_TYPE, Claim_Invoice).
- Must not exceed 30 characters in length (subject to DBMS naming limitations).
- Where the target DBMS is Oracle, table names:
  - Must not exceed 30 characters in length.
  - Must start with a letter of the alphabet.
  - May include letters, numbers and underscores.
  - Must be unique within the database, and cannot be an Oracle reserved word (e.g. column, create, data, date, select, etc.).
- Where the target DBMS is DB2 on a UNIX/Intel platform, table names:
  - Must not exceed 128 characters in length.
  - Must start with a letter of the alphabet or a valid accented letter (such as ö).
  - May include letters, numbers, symbols (@, #, $), underscores and valid accented letters (such as ö).
- Where the target DBMS is DB2 on an OS/390 platform, table names:
  - Must not exceed 30 characters in length. A fully qualified table name (remote location + dot + table owner + dot + table name) can be longer but for purpose of data modeling we should be concerned with the unqualified name.
  - Must not contain accented letters.
  - May include letters, numbers, symbols (@, #, $), underscores

Some DBMS prefix table names with some special word when using certain DBMS features. For example, Oracle replication prefixes snapshot logs with MLOG_, and DB2 DPROPR has something similar. Be aware of the DBMS features being used when naming tables so that the special prefixes can be used without changing the table name.

5.6.2 Columns

A column contains values of attributes (fields) of an entity or class modeled by a table. Each column has a name and specific characteristics (including data type and constraints). A column name is unique within a table.

When a column references the primary key of another table (the column is a foreign key), the column name must have the same name and data type as the column in the referenced table. Additionally, a foreign key constraint should be defined on the column to ensure data integrity.

Columns will be named using the general naming standards provided in section 5.4. In addition, column names:

- Must be unique within the parent table.
- Must be named the same as the logical (attribute) name, subject to DBMS naming limitations (e.g. length, reserved words).
- Must be singular.
- Must use uppercase characters or mixed case (subject to the DBMS naming limitations). If using mixed case then capitalize the first letter of each word in the column name (for example: Birth_Date).
- Must use underscores (_) as the delimiter between words in the name (for example: FIRST_NAME, LAST_NAME, Birth_Date). This is used in place of a blank to create the impression of a multi word name to humans but a single word name to the relational database system and application software.
- Must not exceed 30 characters in length (subject to DBMS naming limitations).
- Must end with a class word that is used to indicate the type of data stored. Use the list of class words defined in Appendix B.
- Where the target DBMS is Oracle, column names:
  - Must not exceed 30 characters in length.
  - Must start with a letter of the alphabet.
  - May include letters, numbers and underscores.
  - Must be unique within the table.
  - Must not be an Oracle reserved word (such as column, create, data, date, select).
- Where the target DBMS is DB2 on a UNIX/Intel platform, column names:
  - Must not exceed 30 characters in length.
  - Must start with a letter of the alphabet or a valid accented letter (such as ö).
  - May include letters, numbers, symbols (@, #, $), underscores and valid accented letters (such as ö).
- Where the target DBMS is DB2 on an OS/390 platform, column names:

- Must not exceed 30 characters in length. A fully qualified table name (remote location + dot + table owner + dot + table name) can be longer but for purpose of data modeling we should be concerned with the unqualified name.
- Must not contain accented letters.
- May include letters, numbers, symbols (@, #, $), underscores.

Foreign Key Column Names - Where the foreign key is a system generated primary key from the parent table (i.e. ID), change the name of the column to the following format:

(parent table_name)_ID (e.g. ORGANIZATION_ID, Person_ID, Facility_ID, Location_ID)

Where there is more than one foreign key relationship between the same two tables, resolve duplicate column names by:

- Inventing a meaningful name (in compliance with the naming standards) for the foreign key column that is different from the other column name. For example, use a ‘parent’ or ‘child’ prefix for hierarchical structures, use the code table logical name for the class code table key, *entity_name*_ID.
- Changing the name of the conflicting column in the source entity.

5.6.3 Table Constraints

Table constraint names:

- Must be unique within the database schema.
- Must be named according to the principles set out in Table 5-2 below.
- Must not exceed 30 characters in length where the target DBMS is Oracle.
- Must not exceed 30 characters in length where the target DBMS is DB2.
- Must use uppercase characters.
- Must start with a letter of the alphabet.
- May include letters, numbers and underscores.

Constraint Type	Constraint Format	Example
Check constraint	[table short name]_CHK[n] n = 1, 2, 3, ...	ORGN_CHK1 ORGN_CHK2 ORGN_CHK3
Foreign key	[child table short name]_[parent table short name]_FK[n] n = 1, 2, 3, ... Capture the relationship to the parent table. Only need the numeric n if there are multiple relationships between the same two tables.	ORGN_CODE_FK1 ORGN_CODE_FK2 PRSN_CODE_FK
Primary key	[table short name]_PK	ORGN_PK PRSN_PK
Unique key	[table short name]_UK[n] n= 1, 2, 3, ...	ORGN_UK1 ORGN_UK2

Table 5-2: Table Constraint Naming Standards

Note: ORGN is the short name for the table “ORGANIZATION”. PRSN is the short name for the table “PERSON”.

5.6.4 Indexes

Indexes are a database structure used to optimize the execution speed of queries, enforce referential integrity, and to facilitate the ordering of data based on the contents of the index’s field or fields. Indexes may be generated by a user, or internally by the DBMS. A single column index has only one column in the index key. A concatenated index, also known as a composite index, is created using multiple columns in a table.

Indexes must be named.

Index names:

- Must be unique within the database schema.
- Must use the name of the related constraint as shown in Table 5-3 below.
- Must not exceed 30 characters in length where the target DBMS is Oracle.
- Must not exceed 30 characters in length where the target DBMS is DB2.
- Must use uppercase characters.
- Must start with a letter of the alphabet.
- May include letters, numbers and underscores.

Constraint Type	Constraint Format	Index Format	Example
Foreign key (not usually indexed unless performance is an issue)	[child table short name]_[parent table short name]_FK[n] n = 1, 2, 3, ... Capture the relationship to the parent table. Only need the numeric n if there are multiple relationships between the same two tables.	[constraint name]_I	ORGN_CODE_FK1_I ORGN_CODE_FK2_I PRSN_CODE_FK_I
Primary key	[table short name]_PK	[constraint name]_I	ORGN_PK_I PRSN_PK_I
Unique key (not usually indexed unless performance is an issue)	[table short name]_UK[n] n= 1, 2, 3, ...	[constraint name}_I	ORGN_UK1_I ORGN_UK2_I
Non-unique index (not usually used unless performance is an issue)	[table short name]_NU[n] n= 1, 2, 3, ...	[constraint name}_I	ORGN_NU1_I ORGN_NU2_I

Table 5-3: Index Naming Standards

Note: ORGN is the short name for the table “ORGANIZATION”; PRSN is the short name for the table “PERSON”.

5.6.5 Non-Table Objects

Non-table objects:

- Must be named according to the principles set out in Table 5-4 below.
- Must not exceed 30 characters where the target DBMS is Oracle.
- Must not exceed 30 characters where the target DBMS is DB2.
- Must use underscores (_) as the delimiter between words.
- Must start with a letter of the alphabet.
- May include letters, numbers and underscores.

Table 5-4: Non-Table Objects Naming Standards

Non-Table Object	Name Format	Example
Cursors	Cursor name should be a descriptive name followed by the abbreviation cur: [descriptive_name]_CUR	INSERT_ORGANIZATION_CUR
Functions A function is a named block of code that can be stored locally in the application or remotely in the server database. A function can only return a single value.	Since a function returns a value, it should be named as if it were a variable that already contained the returned value F_[return value].	F_NEW_NAME F_CODE_VALUE F_ERROR_MESSAGE
Package (ORACLE) A package is a PL/SQL object that groups PL/SQL procedures and functions. Packages consist of two parts: a package header and a package body. The package header defines the functions or procedures in the package. The actual code is stored in the package body.	Packages represent a collection of procedures and/or functions related to a specific table or subject/action. Their names should reflect the table name or subject/action. [table name]_PKG [subject | action] _PKG A package name must be unique within a database schema. Ideally, the package will also be unique within the database instance. For source control the	COMMON_PKG ERROR_PKG FACILITY_PKG ORGANIZATION_PKG LOCATION_PKG PERSON_PKG

Non-Table Object	Name Format	Example
	package body should be separated from the package header and named as follows: [table name]_PKG_BODY [subject | action]_ PKG_BODY For source control the package body should be separated from the package header and named as follows: [table name]_PKG_HEAD [subject | action]_PKG_HEAD	
Procedures Stored procedures are named blocks of code. Stored procedures are best used to for calculating results that cannot readily be derived using SQL.	Procedures represent an action, so their names should be a verb plus a subject. In many cases, the subject will be a table name: P_[verb]_[subject]	P_UPDATE_CONTACT P_GRANT_RIGHTS
Sequence	[column name]_SEQ	PERSON_ID_SEQ ORGANIZATION_ID_SEQ LOCATION_ID_SEQ
Triggers Database triggers are named blocks of code attached to a table. Trigger procedures are executed when a specified database-related event (update, insert or delete) takes place against a specified table in the database. Triggers may be activated before or after the event. They can be used to augment	Trigger Names must contain the table name suffixed by the trigger actions and statement level. Actions: I—insert U—update D—delete Statement levels: BS—before statement BR—before row AR—after row AS—after statement Unique to the table with	PERSON_IUBR ORGANIZATION_IUBR LOCATION_IUBR LOCATION_IBS LOCATION_IAS

Non-Table Object	Name Format	Example
referential integrity, enforce additional security, enforce business rules consistently or enhance the available auditing options.	which the trigger is associated. Ideally unique within the database schema.	
Variables	Variables should be named for what they represent (e.g. table column). Where a variable does not represent a table column then a descriptive name should be chosen. A data type descriptor should prefix the variable to indicate the type of values it contains. Data type descriptor: N—numeric D—date V—text (varchar)	D_EFFECTIVE_DATE N_PERSON_ID V_ORGANIZATION_NAME
Views A view can be thought of as a mask overlaying one or more tables, such that the columns in the view are found in one or more underlying tables. To the end user of a database, views should appear identical to base tables. Views do not use physical storage to store data. Views can be used to limit access to certain columns of a table, present calculated values, or to ease end user access to joined tables.	A view represents an amalgamation of tables. A descriptive name should be chosen for the view. For relational views: [descriptive name]_VW [table name]_VW For object views: [descriptive name]_OV [table name]_OV Must be unique within the database schema	EMPLOYEE_DEPARTMENT_VW MESSAGE_OV

5.7 XML Schema Naming Standards

The purpose of the XML schema is to show how the data and metadata will be described, structured, managed, interchanged, and presented. XML schemas vary from the traditional logical and physical models in that the XML schema introduced some new XML specific constructs and objects. These new objects and constructs are created for defining, validating and processing XML documents required by the business and applications. Each XML schema objects and constructs are enclosed between the open and close tag pair.

XML schema tag names will follow the naming standards provided for Entities and Attributes with some additional naming standards as described in this section.

5.7.1 XML Tag Naming Standards

Rule	Description	Example
Attribute	Attributes should be defined using lower camel case, i.e. attribute names have their first letter in lower case with each subsequent word, phrase, or acronym capitalized.	<degreeDiscipline = “Chemistry”>
Element	Elements should be defined using upper camel case, i.e. element names have their first letter uppercase with each subsequent word, phrase, or acronym capitalized.	<PostalCode>
Type	Types should be defined using upper camel case and appended with the term “Type”.	<PostalCodeType>
Complex Type	Complex types should bear the name of the XML elements whose content model they represent and append the name with the term “Type”.	<CivicAddressType>
Simple Type	Simple types should bear the name of data dictionary domain they represent and append the name with the term “Type”.	<DescriptionTextType>
Acronyms	The use of acronyms is discouraged in general. However, the use of some acronyms that are widely understood and accepted in the business communities is permitted. In this case, all acronyms must be shown in uppercase.	<UserID> <SIN>
Illegal	Illegal characters cannot be used (e.g.: / * #	NOT allowed:

Rule	Description	Example
Characters	$, etc.). Recommended characters in a tag name are basically limited to letters and underscores.	<BirthDate/Time> Allowed: <BirthDateTime>, <Birth_Date_Time>
Similar Names	Use the similar tag names with elements in a similar child structure.	<ContactAddress> <HomeAddress> <WorkAddress>
Plural Names	Use plural tag names only for collections.	<CreditCards>
Name Size	Element and attribute name sizes have no limitation. However, the names must be meaningful to the business and concise.	<CustomerRelationshipInformation>
Prefixes	Element and attribute names may optionally include in a prefix word when appropriate. See the Tag Name Prefixes table for the proposed representation types.	<HasFirstNationStatus> as an element, <canSpeak> and <canRead> as attributes.
Suffixes	Element and attribute names should incorporate suffixes from the proposed list of representation types (a subset of class words in Appendix B that have also been adapted in ebXML) when appropriate. See the Tag Suffixes table for the proposed representation types.	<StartDate> <BilledAmount>
Keys / KeyRefs or ID / IDREFS	All XML keys/keyRefs or ID/IDREFS are named using the XML attributes naming standard.	

5.7.2 XML Tag Name Prefixes

Representation	Representation Type	Description
Is	Boolean	An enumerated list of two, and only two, values which each indicates a Condition such as on/off; true/false, etc. Example: <isIncorporated>
Are	Boolean	An enumerated list of two, and only two, values which each indicates a Condition such as on/off; true/false, etc.
Can	Boolean	An enumerated list of two, and only two, values which each indicates a Condition such as on/off; true/false, etc. Example: <canSpeak>
Have	Boolean	An enumerated list of two, and only two, values which each indicates a Condition such as on/off; true/false, etc. Example: <haveCanadianCitizenship>
Has	Boolean	An enumerated list of two, and only two, values which each indicates a Condition such as on/off; true/false, etc. Example: <hasFirstNationStatus>

5.7.3 XML Tag Name Suffixes

Representation Type	Description
Amount	A number of monetary units specified in a currency where the unit of currency is explicit or it may be implied.
Boolean	An enumerated list of two, and only two, values which each indicates a Condition such as on/off; true/false etc. The Boolean value is expressed in XML schema using Prefixed Tag Name phrases.
Code	A character string that represents a member of a set of values.
Date	A day within a particular calendar year. Note: Reference ISO 8601
DateTime	A particular point in the progression of time. Note: This may incorporate dependent on the level of precision, the concept of date.

Representation Type	Description
Identifier	A character string used to identify and distinguish uniquely one instance of an object within an identification scheme.
Measure	A numeric value that is always associated with a unit of measure.
Name	A word or phrase that constitutes the distinctive designation of a person, object, place, event, concept etc.
Number	A numeric value that is often used to imply a sequence or a member of a series.
Quantity	A number of non-monetary units. It is normally associated with a unit of measure.
Rate	A ratio of two measures.
Text	A character string generally in the form of words.
Time	The time within any day in public use locally, independent of a particular day. Reference ISO 8601:1988

6. Quality Assurance

6.1 Quality Measures[2]

A model should fully convey its intended message and only its intended message. Otherwise, a model does not reliably speak for or about the business. It doesn't communicate effectively, and is of reduced quality and value.

The following measures can be used to test the quality of a model:

- **Accuracy -** Do the definitions reflect the essential, inherent meaning of a business concept?
- **Clarity -** Is the meaning clear?
- **Completeness -** Have all of the model objects been included with the appropriate level of documentation?
- **Conciseness -** Are the definitions straight to the point and information not repeated?
- **Consistency -** Does one piece of information not contradict another?
- **Alignment -** Does the model provide for business and architectural alignment?

Data model checklists have been provided in the "Information Architecture Review Questionnaire". These checklists can be used by project teams to perform a quality review of their data models to ensure that they are consistent with the contents of the GO-ITS 56.3 Information Modeling Standard Appendix A – Information Modeling Handbook (IMH) and that the data models meet the six quality measures identified above. These checklists are used during the architecture review process at cluster architecture review boards and at ACT and ARB.

6.1.1 Accuracy

A model must accurately reflect the business area it represents. Accuracy requires that each assertion in the model truly reflects the business intent. Passing the test of accuracy requires combing through the model and asking the business: "Is this right?"

Examples of 'accuracy' checks include:

- **Definitions -** Does the definition of each object (entity, attribute, relationships, etc) reflect the true meaning of the business concept it represents?

[2] The material in this section is based on "Measuring the Quality of Models", John C. Claxton and Peter A. McDougall, The Data Administration Newsletter, October 2000.

- **Relationships -** Does the relationship show the right business rule between two business concepts? Any disagreement between the model and the business intent creates a contradiction. This indicates that the model does not capture and reflect the business.
- **Domain of Values -** Does each domain of values correspond to the business scope? For example: Types of Client Needs are different within the business of 'Social Services' versus that of the 'Education' business. A domain that does not align with the required business scope for the concept is inaccurate.

6.1.2 Clarity

A model must state its message in clear and unambiguous terms. A model lacks clarity when the same statement has two or more interpretations. The following guidelines can eliminate ambiguity in a model:

- Use short and direct statements for descriptions.
- State the business requirement with statements that can only have one interpretation.

6.1.3 Completeness

A complete model provides a thorough picture of the business and fully documents the business area it represents. Readers have everything they need to understand what the model communicates.

Examples of 'completeness' checks include:

- **Scope -** Do the entities depicted fully reflect the scope of the business being modeled?
- **Structural -** Have all of the objects (diagram, entity, attributes, relationships, etc.) that are required at this level of model been included?
- **Semantic -** Have all of the objects (diagram, entity, attributes, relationships, etc.) been described to the extent that is required at this level of model?

6.1.4 Conciseness

A concise model does not repeat information or have overlapping data. Benefits of a concise model include:

- Easy to find specific information about the business.
- Easy to maintain, both in development and long-term.

6.1.5 Consistency

A consistent model does not have statements that conflict with other statements from the same model. A consistent model provides a uniform view of the business.

Consistency (diagramming, naming, defining, etc.) across all models will help project teams to find and understand the required information quickly.

6.1.6 Alignment

A key model quality consideration is to develop a model according to the business needs and key organizational standards. These two characteristics of model quality are respectively termed business alignment and architectural alignment.

- **Business Alignment** - A business aligned model reflects the business it is meant to represent by using common business terminology and incorporating business related standards where they exist.
- **Architectural Alignment** - An architecturally aligned model leverages data concepts and relationships from existing relevant reference data models (e.g. BACM, CDEM Party, Case Management, etc.) as a foundation for building the data model.

6.2 Quality Assurance Process

Quality assurance is the certification that the model adheres to a set of standards practiced by an organization (e.g. IMH) to ensure that the model meets the test of the six quality measures (accuracy, clarity, completeness, conciseness, consistency and alignment).

It is recommended that quality assurance review occur at different stages in a project, these include:

- At project initiation.
- At important project milestones; including architecture governance checkpoints.
- At project implementation.

Appendix A1 - Guideline for Abbreviating Data Object Names

Sometimes there is an upper limit for naming data objects in various types of data models. For this reason, abbreviations and acronyms are often used to shorten names to comply with a name length restriction, or to be used to append to other names, such as Class Words.

The following guidelines have been provided to derive an abbreviated data object name, to help ensure that new abbreviations are consistently derived across the OPS.

Rules for Forming Name Abbreviations

In general words less than seven (7) characters in length are not abbreviated

Rules for Abbreviating Single Word Names

1. Check the table to make sure that a standard abbreviation doesn't already exist
2. Check that a standard acronym doesn't already exist.
3. Remove all special characters (-, &, ', etc.).
4. Remove all vowels, except the leading (or leading two).
5. Replace multiple consonants with a single consonant.
6. Truncate consonants after the sixth letter.

For example:

The word 'Watercraft' would be abbreviated as 'Wtrcft'

Rules for Abbreviating Multiple Word Names

1. Apply the abbreviating rules for single word names for each individual word in a multiple word name.
2. Alternatively, for names with more than two words, use the first letter of each word if it seems appropriate.

For example:

The words 'Standard Deviation' would be abbreviated as 'Std Devtn'

The words 'Reduced Instruction Set Computer' would be abbreviated as 'RISC'.

Other General Rules

If the resulting abbreviated word conflicts with a popular acronym, it is better to avoid the acronym and create a new abbreviation.

For example, the abbreviated word ‘Tel’ is commonly known to represent the word “Telephone” you should avoid using ‘Tel’ for the word Telemetry. Instead, the word ‘telemetry’ would be abbreviated as ‘Tlmtry’.

In another example, the acronym ‘SSL’ is commonly understood to mean ‘Secure Socket Layer’, so it would not be advisable to use ‘SSL’ to mean ‘Security Standard Layer’. The words ‘Security Standard Layer’ would be better abbreviated as ‘Scrty Std Lyer’.

Commonly Used Abbreviated Words

Over the years, a set of commonly used/known abbreviations have been in use by many business systems across the OPS for technical, business or both purposes. More details on these commonly used word abbreviations can be found in the supplementary document “Guide to Abbreviating Data Object Names”.

Appendix B - Class Words

A class word is a word or term that classifies the content and role of a piece of data described by a data attribute (which can be an entity attribute, an entity class attribute, a column name, an XML schema element or XML schema attribute). A data attribute may be further defined by a range of values (domain).

A class word is defined by its name and description. A class word is used to indicate the type of data that the data attribute represents or stores. Data attribute names should end with a class word.

Class Word Usage

- Must be used when developing logical and physical level data models.
- Assign a class word to every data attribute.
- Assign class words according to the nature of the data being defined.
- **Assign class words from the correct class word category.** Class words are selected from one of six different class word categories, as follows:
  1. **Chronology** – A value that indicates a date, a point in time, a sequence.
  2. **Description** – A string of characters that is not treated as 'fielded' data. It is used primarily for the purpose of storing free-form text.
  3. **Label** – A data item used for naming or identifying an item (e.g. a person, a row in a table, etc.).
  4. **Measurement –** A value that indicates dimension, capacity, amount, performance or a count.
  5. **Multimedia** – A data item that indicates a multimedia item (video, image, sound, file).
  6. **Spatial Coordinate** – A value that indicates a geospatial position on the surface of the earth.

Class words assigned to each of these categories may only function in the role defined by the category name. This standard allows a data user to know the role an attribute plays in the database by noting the category of the class word used in the attribute name.

Table B-1: List of Standard Class Words

Category	Class Word	Abbreviation	Definition	Data Type
Chronology	Date	DATE	An alphanumeric string used to represent the calendar date when an event occurs. The ISO8601 standard format is CCYY-MM-DD.	DATE
Chronology	Day	DAY	An alphanumeric string used to identify the day an event occurs. The ISO8601 standard format is DD.	DATE
Chronology	MonthDay	MTHDAY	An alphanumeric string used to represent a specific day of the month when an event occurs; such as the third of May. The ISO8601 standard format is MM-DD.	DATE
Chronology	Time	TIME	An alphanumeric string used to represent an instant of time when an event occurs. The ISO8601 standard format is hh:mm:ss.	TIMESTAMP
Chronology	Timestamp	TS	An alphanumeric string used to represent an instant of time when an event occurs. Timestamp is used when a high precision of time is required. The ISO8601 standard format is CCYY-MM-DD-hh:mm:ss:n.	TIMESTAMP
Chronology	Year	YEAR	An alphanumeric string used to represent a calendar year when an event occurs. The ISO8601 standard format is CCYY.	DATE

Category	Class Word	Abbreviation	Definition	Data Type
Chronology	YearMonth	YEARMTH	An alphanumeric string used to represent a specific month in a specific year when an event occurs. The ISO8601 standard format is CCYY-MM.	DATE
Description	Description	DESC	An alphanumeric string used to describe a data element. Note: Where the name of the attribute will be singular 'DESCRIPTION' then a suffix of DESC is not required.	STRING
Description	Text	TEXT	An unconstrained string of characters, or any freeform comment or notes field. Text, unlike name, description, and address, does not have any specific pre-determined purpose.	STRING
Description	Title	TITLE	A descriptive or general heading.	STRING
Label	Code	CD	An alphanumeric string which has a domain with a larger set of values than 'Flag' or 'Indicator'. It is used to uniquely identify status, or state of activity, or instances of data. The class word 'Code' is usually used in the name of an attribute to indicate one of finite outcomes, states, status, or set of data values.	STRING
Label	Flag	FLG	An alphanumeric string which represents one of only two possible values, such as Yes/No, On/Off, True/False. The class word Flag is usually used in the name of an attribute to indicate one of two possible outcomes.	STRING

Category	Class Word	Abbreviation	Definition	Data Type
Label	Identifier or Key	ID	A numeric or alphanumeric string used to represent a system-generated primary key.	INTEGER or STRING
Label	Indicator	IND	An alphanumeric string which usually has a domain of two or more values. The class word Indicator is usually used in the name of an attribute to indicate one of very few possible outcomes, states, status, or set of data values.	STRING
Label	Name	NAME	A character string which gives the name or title of a business object. It is usually a commonly used, descriptive name or title, but is not to be confused with a short description. It is usually a proper name or title, such as Person Name, Product Name, Article Name, etc.	STRING
Label	Number	NUM	A number which is not used for the purpose of measuring a quantity or serving as a counter, but which is a numeric value.	DECIMAL
Label	Prefix	PREFIX	An alphanumeric string which represents the affix occurring at the beginning of a word (e.g. name prefix, partition prefix).	STRING
Label	Sequence	SEQ	A numeric string used to indicate an application generated sequence number (e.g. sequence of line items within an Order).	INTEGER
Label	Suffix	SUFFIX	An alphanumeric string which represents the affix occurring at the end of a word (e.g. name suffix, street number suffix).	STRING
Label	User ID	USER	An alphanumeric string used in a physical data model to capture a user ID for auditing purposes.	STRING

Category	Class Word	Abbreviation	Definition	Data Type
			A Binary Large Object (e.g. an electronic user identifier such as a photo ID, an electronic signature, a digital signature) used in a physical data model. Must be accompanied by an Internet Media Type attribute (MIME/IANA Type and Subtype) to indicate the type of media.	BLOB
Measurement	Amount	AMT	A number used to represent the measurement of monetary value, e.g. dollars and/or cents ($99, $99.99). This can also be referred to as cost or price. An amount field may be specified as an integer, or may include decimal positions, may be positive or negative, or may be in various units of measurement, depending upon the specific domain associated with the attribute.	DECIMAL
Measurement	Area	AREA	A number used to represent the two-dimensional size of a defined part of a surface, typically a region bounded by a closed curve. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL
Measurement	Count	CNT	A number used to keep track of the count of an item.	INTEGER
Measurement	Depth	DPTH	A number used to represent the measurement of the depth of an object, or the depth below water level, etc. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL

Category	Class Word	Abbreviation	Definition	Data Type
Measurement	Duration	DUR	A number used to represent a span of time. Must be accompanied by a unit of time measured in seconds, minutes, hours, days, weeks, months, or years.	DECIMAL
Measurement	Factor	FACTOR	A number by which a given quantity is multiplied or divided in order to indicate a difference in measurement.	DECIMAL
Measurement	Height	HT	A number used to represent the measurement of the height of an object, or height above ground level, etc. When height is measured above sea level, it will be denoted as Elevation. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL
Measurement	Length	LGTH	A number used to represent the measurement of the length of an object. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL
Measurement	Percent	PCT	A number which represents percentage (Numerator/ denominator * 100). The definition must specify what the numerator and denominator are. The two numeric values must have the same units, so the percentage does not have a unit.	DECIMAL
Measurement	Perimeter	PERIMTR	The length of a closed path that surrounds an area. Must be accompanied by a Unit	DECIMAL

Category	Class Word	Abbreviation	Definition	Data Type
			of Measure attribute or unit in the definition.	
Measurement	Quantity	QTY	A number which represents how much there is or how many there are of something that one can quantify, such as: Order Quantity, etc.	DECIMAL
Measurement	Rate	RATE	A numeric ratio with associated units. A numeric field representing a measurement's change per unit of time, e.g. Growth Rate.	DECIMAL
Measurement	Weight	WT	A number used to represent the measurement of the weight of an object. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL
Measurement	Width	WDTH	A number used to represent the measurement of the width of an object. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL
Multimedia	Audio	AUDIO	A binary object used in a logical data model to represent a (series of) sound(s).	BLOB
			A Binary Large Object (e.g. audio, video) used in a physical data model. Must be accompanied by an Internet Media Type attribute (MIME/IANA Type and Subtype) to indicate the type of media.	
Multimedia	File	FILE	A string used in a logical data model to capture large documents (e.g. Word document, Spreadsheet).	BLOB

Category	Class Word	Abbreviation	Definition	Data Type
			A Binary Large Object (e.g. spreadsheet, audio, image, video) used in a physical data model to represent a file which contains a spreadsheet, audio or video file or an image. Must be accompanied by an Internet Media Type attribute (MIME/IANA Type and Subtype) to indicate the type of media.	
			A Character Large Object used in a physical data model to represent a file such as a word document. Must be accompanied by an Internet Media Type attribute (MIME/IANA Type and Subtype) to indicate the type of media.	CLOB
Multimedia	Image	IMAGE	The binary representation of a picture, or a video file used in a logical data model.	BLOB
			A Binary Large Object used in a physical data model to represent an image such as a picture, audio or video. Must be accompanied by an Internet Media Type attribute (MIME/IANA Type and Subtype) to indicate the type of media.	
Spatial Coordinate	Latitude Longitude	LATLONG	A pair of numbers that determine a point location on the surface of the earth comprising a latitude component and a longitude component. **Latitude:** The angular distance north or south from the earth's equator measured through 90 degrees	DECIMAL

Category	Class Word	Abbreviation	Definition	Data Type
			Longitude: The arc or portion of the circumference of the earth at or parallel to the equator expressed either in degrees or in time relative to the prime meridian, which runs through Greenwich, England. Must be accompanied by a Unit of Measure attribute or unit in the definition.	
Spatial Coordinate	Universal Transverse Mercator	UTM	A set of 3 component numbers; Easting, Northing and Zone, that together determine a point location on the surface of the earth. One of many methods of displaying the earth, which is a globe, as a rectangular surface, specifically dividing the earth's surface into rectangular grids 6 degrees east-to-west and 8 degrees north-to-south, assigning any one location an easting and a northing value relative to the true lines of latitude and longitudes in the centre of each grid cell. Must be accompanied by a Unit of Measure attribute or unit in the definition.	DECIMAL

Appendix C – Using Template Models

C.1 What Are Template Models?

A template model is a data model, normally expressed as a logical data model, which has been developed to document common data requirements which reflect a broader enterprise perspective. Template models provide a starting point to be used when modeling business data requirements for a specific project. Using template models supports alignment and common context across the organization, thereby enabling better data integration, consistent language, enforcing standards, etc.

A new data model, at a project or enterprise level, should be developed leveraging (making use of) existing template models wherever possible. A modeler selects from one or more template models whose model elements, either as a whole or in part, are applicable to their project scope, and then proceeds to modify the model by adding more detail (subtypes, attributes, relationships, etc), or removing irrelevant elements from the model, based on the project requirements. Sections C.3 and C.4 provide more details about the rules of using a template model.

Across the OPS there are a number of template data models which, if applicable, must be leveraged:

1. **Common Data Elements Model (CDEM):** An enterprise data model expressed as a logical data model that provides standardized definitions of information that is common to all mandates of the Government of Ontario. The following CDEM data subject areas logical data models have been developed:
    - Party - detailed data elements for Individuals, Organizations, their roles, relationships, names and characteristics
    - Address - detailed data elements for physical, mailing, telephone, email and net addressing, in Canada, the US and internationally
    - Privacy and Security - covers IT security and its support for information privacy, including identification, authentication, authorization, system resources and system events.
2. **Enterprise Models:** A data model developed to represent the data architecture of a specific enterprise (where enterprise can be defined as Ministry, Ministry division, I&IT cluster, or a specific program).
3. **Reference Models:** A data model that represents information about a common business. For example:
    - GO-ITS 56.2 OPS Case Management Reference Model
    - GO-ITS 56.5 OPS Grants Management Reference Model
    - Canadian Service Description Markup Language (CSDML) Model.

C.1.1 Using the CDEM Template Model

Figure C-1 shows how the CDEM template model can be leveraged as a starting point for the development of either an enterprise model or a project model.

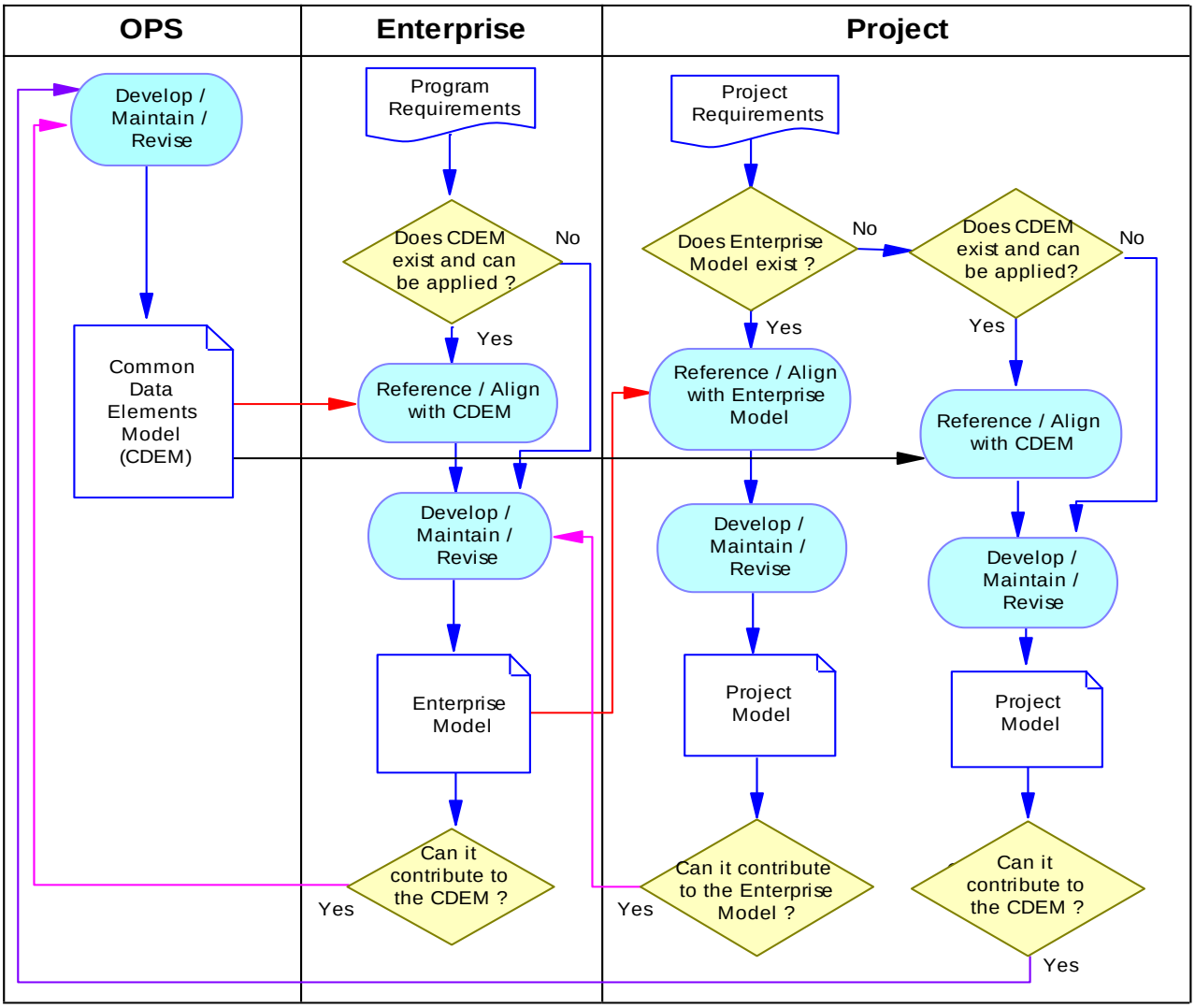


Figure C-1: Using the CDEM Template Model

An enterprise (Ministry, Ministry Division, Ministry program area, etc) model should be developed by leveraging relevant elements from the CDEM template model. When the enterprise model is complete, new elements may be used to extend the CDEM template model. This way the enterprise model is aligned with the CDEM and the CDEM continues to evolve as the OPS data architecture grows.

A project[3] level model should be developed by leveraging relevant elements from an enterprise model, if the enterprise model exists. When the project level model is complete, new elements may be used to extend the enterprise model which in turn may then be applied to the CDEM template model.

Alternatively, if an enterprise model does not exist, the project level model should be developed by leveraging (making use of) relevant elements from the CDEM template model. When the project level model is complete, new elements may be used to extend the CDEM template model. This way the project model is aligned with the CDEM and the CDEM continues to evolve as the OPS data architecture grows.

The premise here is that if OPS enterprise models are developed by leveraging relevant elements from the CDEM template model, and then subsequently project models are developed by leveraging relevant elements from a relevant enterprise model this will result in all project models, across the OPS, having a common set of standardized element definitions, which are inherited indirectly from the same CDEM template model. This will lead to the standardization of data element definition across all enterprises (Ministries, programs, etc) within the OPS, which in turn will make the goal of integrating data and information easier, more effective, and more efficient.

C.2 General Template Model Usage Rules

The following rules will ensure that target models[4] are consistent or aligned with template models. The rules make template models useful and flexible, while defining reasonable restrictions on the target models.

- The target model can and should follow best practices for data modeling, as documented in the IMH. Target models should not include elements from the template model that are not relevant to the project's scope.
- A template model or subject area may have documented usage rules that state exceptions to these rules.
- The usage rules are written as if a modeler is creating a new target model based on a template model. The same rules apply if a different modeling process is used; for example, a target model is being aligned after it has been developed.

In the usage rules, the modeler is the person who creates or modifies a target model.

The modeler starts by making a copy of the template model to create a new target model. The models have a loosely coupled relationship; the template model can be updated independently of the target model. The modeler selects the elements from the template model that support their project's information requirements. The modeler then adds more detail (subtypes, attributes, relationships, etc) to these elements; subject to restrictions identified in the template model usage rules.

[3] A project can be at any enterprise level, e.g. ministry, division or cluster.

[4] A target model is the term being used for an enterprise or a project level data model under development

Detailed usage rules for logical level template models are provided in section C.3.

C.3 Usage Rules for Logical Level Template Models

The usage rules for logical level template models are more restrictive in order to enable better data integration. In particular:

- Model elements that have been declared as mandatory in the template model may not be made optional in the target model.
- Attributes that have been declared as mandatory in the template model must be used in the target model.
- Specifications of attribute data type, length and allowed values must not be substantively changed in the target model.
- Unique business identifiers that were identified in the template model must be used in the target model.
- When related entities are selected from a template model their relationships must also be used in the target model.
- Relationship cardinality in a template model cannot be changed in the target model:
  - One-to-Many relationships may not be made Many-to-Many.
  - One-to-One relationships may not be made One-to-Many.

More detailed usage rules on aligning a logical level target model with a logical level template model are discussed in more detail in this section.

C.3.1 Naming and Definition Rules

Modelers must not rename a template model element in their target model. Instead:

- Change the terminology in the target model to match the template's names and definitions.
- If the modeler's enterprise uses a synonymous term, record it as an alias.

  For example the organization may use the term STAKEHOLDER which is equivalent in meaning to the entity PARTY in the CDEM Address. The term STAKEHOLDER would then be recorded as an alias for the entity PARTY in the target model.

Modelers must not substantively change the meaning of a template model element:

- The template model element's definition must be the basis for the target model element’s definition.
- Notes and explanatory material may be removed if they do not apply to the project scope. The modeler may add further notes and explanation, especially to document how the target model is aligned with the template model.

- The meaning of an “Other” subtype may change, depending on which subtypes are included in a target model (see section D.3.3 Subtype Usage Rules).

A template model element (entity, attribute, etc) must not be used for anything with a substantively different definition. Homonyms must be resolved.

For example, a template model defines the entity DRIVER as “a person who drives a vehicle”. The target model may not therefore use the term DRIVER for anything else, such as “A screwdriver” or “A motivating factor”, even if vehicle drivers are not within the scope of the target model.

To resolve homonyms, change the name of the target model element that does not agree with the template model definition. For example, use SCREWDRIVER or MOTIVATING FACTOR. Do not simply add an organization or application name as a prefix or suffix (for example, ABC DRIVER).

Where the enterprise has existing terminology that conflicts with the template model element names:

- Create subtypes, related entities or additional attributes to *capture the differing definitions* in the target model.
- Document the differences between model elements that might be confused, in all relevant definitions.

Application user interfaces (application screens, paper forms, telephone voice prompts) can still use the terminology best understood by the business user. This may be terminology from the template model, the target model, or other business terms. Application documentation should show the mapping from user interface terminology to the data model.

C.3.2 Entity Usage Rules

Modelers may include or omit any entity from the template model in their target model, to meet their business scope and requirements. There are exceptions related to subtypes (section C.3.3 Subtype Usage Rules) and parents of identifying relationships (section C.4.4 Unique Identifier Usage Rules).

Modelers may create any entity. It is preferable to create subtypes of entities found in the template model, rather than create new entities, if possible. Additional attributes may be added to the template entities, through the normalization process.

Entities can be simplified under the following circumstances:

- Simplified subtype hierarchy (section C.3.3 Subtype Usage Rules).
- One-to-One relationships (section C.3.4 Collapsing One-to-One Relationships).
- Only the identifier is in scope (section C.4.4 Unique Identifier Usage Rules).
- Only one occurrence in scope (section C.4.4 Unique Identifier Usage Rules).

C.3.3 Subtype Usage Rules

A set of subtypes at the lowest selected level of the hierarchy may be converted to a *discriminator attribute* in the logical data model if no attributes of the subtypes are in use.

If an “Other” subtype is defined, it is always the out-of-scope or undefined remainder of subtypes. Thus, its meaning can vary from the template model to the target model. See examples below.

Constraints

The Entity Usage Rules have these constraints:

- If a template subtype is included, all of its supertypes must be included or collapsed (see Collapsing Supertypes).
- If some, but not all, subtypes are selected from the template model (at a single level in a subtype hierarchy), the target model must either have an 'Other' subtype for the remainder, or a note that explains that not all of subtypes from the template model are included.
- Assuming subtyping is exclusive, a subtype cannot be added at the same level as template subtypes, except if it defines part of the template's “Other” subtype.

Collapsing Supertypes

Collapsing supertypes in logical data models means that the modeler has selected a subtype from the template model and used it in a target model without its supertypes. If this is done:

- The subtype entity name must be used, not the supertype name. The description for the subtype entity should list the template supertype in order to identify the alignment with the template model.
- All attributes and relationships of the supertypes are inherited by the selected subtype (but they can be omitted from the target model, if allowed by the applicable usage rules).

The following is an example of subtypes in a template model (the supertype is BUSINESS):

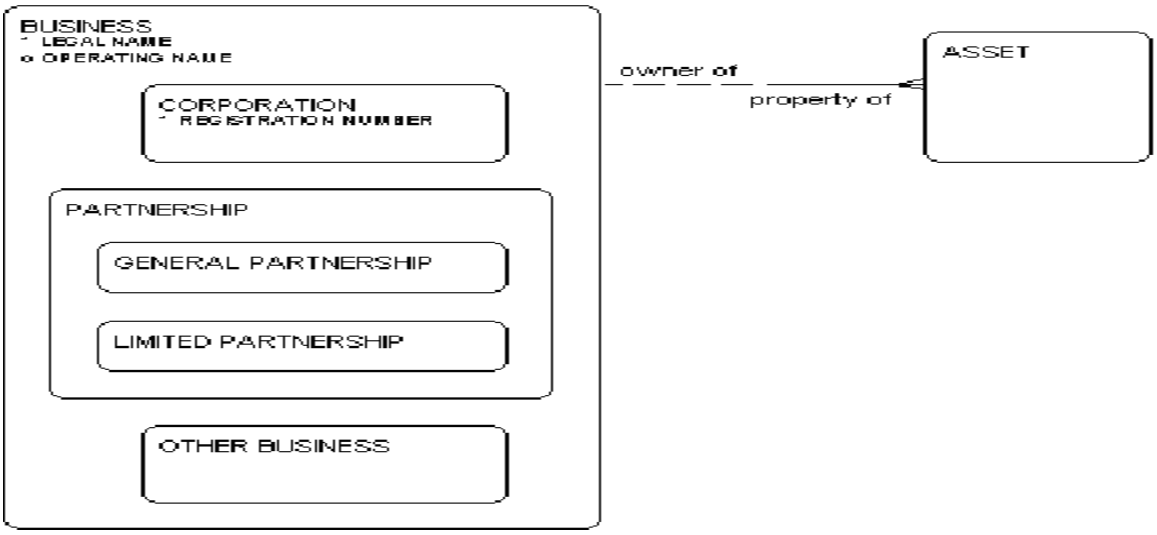


Figure C-2: Example 1 - Subtypes in a Template Model

OTHER BUSINESS is defined as “a BUSINESS other than a CORPORATION or a PARTNERSHIP.” The dashed line indicates an optional relationship (“a BUSINESS *may be owner of one or many* ASSETs”.)

The following is one possible target model representation:

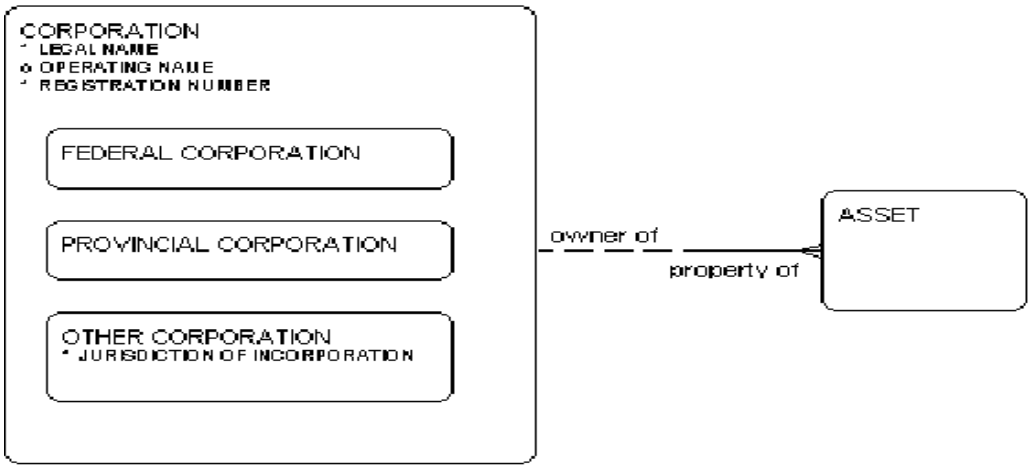


Figure C-3: Example 1 - Target Model 1

Only ASSET and the CORPORATION subtype were in scope for the target model. More detail about CORPORATIONs was required. Attributes and relationships of the supertype BUSINESS became properties of CORPORATION. The definition of CORPORATION includes the statement “CORPORATION is a subtype of BUSINESS as defined in the <name of template model>”.

The following is another example of a target model:

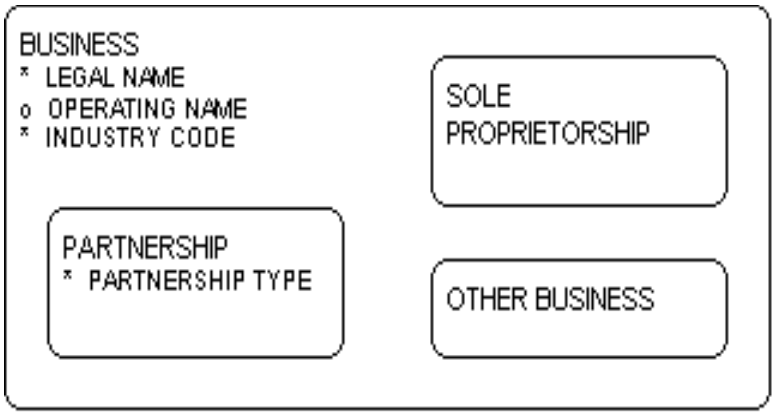


Figure C-4: Example 1 - Target Model 2

The attribute INDUSTRY CODE was added to the supertype BUSINESS. ASSET was omitted since it was out of scope for the target model. The subtypes of PARTNERSHIP had no attributes or relationships, so they were converted to the discriminator attribute named PARTNERSHIP TYPE.

This project was not interested in making use of CORPORATIONs but was interested in SOLE PROPRIETORSHIPs. Thus the description of OTHER BUSINESS now states “A BUSINESS other than a PARTNERSHIP or a SOLE PROPRIETORSHIP May be a CORPORATION, which is defined in the <*name of template model*>.”

C.3.4 Collapsing One-to-One Relationships

The following rules apply to identifying and non-identifying relationships:

- Modelers may wish to retain One-to-One relationships for clarity.
- Completely optional One-to-One relationships must not be collapsed, since either entity can exist independently of the other.
- Entities may be collapsed if, in the target scope, they have a mandatory or contingent (optional becoming mandatory) One-to-One relationship. To do this:
  - If the relationship is identifying, collapse the entity into the parent (independent) entity.
  - If the relationship is fully mandatory, collapse the entity into the entity with more business significance. Attributes from the collapsed entity retain their optionality.
  - If the relationship is contingent, collapse the entity into the independent entity. All of the attributes from the collapsed entity will be optional; additional business rules may be required to manage the optionality.

- Do not change the name of the entity being kept (collapsed into). The collapsed entity name and original relationship names must be reflected in the definition of the entity being kept.

C.4 More Usage Rules for Logical Level Template Models

The following rules apply when aligning a logical level target model with a logical level template model.

C.4.1 Non-Identifying Relationship Usage Rules

- If two entities with a non-identifying relationship are included in a target model, their relationship found in the template model must normally be included.
  - The template model usage rules may specify some exception relationships, which should only be included if they are of interest to the business.
  - Whether a relationship end is optional or mandatory does not determine whether the relationship must be included in target models.
- Modelers may create new relationships.
- Modelers may reduce, but not increase, the cardinality of relationships. For example, a relationship that is One-to-Many in the template model may be One-to-One in a particular business situation.
- Modelers may make an optional relationship mandatory but not vice versa.
- A non-identifying relationship in the template model must not be changed to be identifying in the target model. This would be equivalent to replacing a unique identifier (see section C.4.4 Unique Identifier Usage Rules).
- One-to-One relationships may be left as is, or the entities may be collapsed together (see section C.3.4 Collapsing One-to-One Relationships).

Notwithstanding the above rules, modelers may resolve Many-to-Many relationships. This involves creating two identifying relationships and an associative entity. Mandatory ends of the relationship from the template model must be resolved to mandatory ends of the identifying relationships. Optional ends may remain optional or be changed to mandatory in the target model. For example:

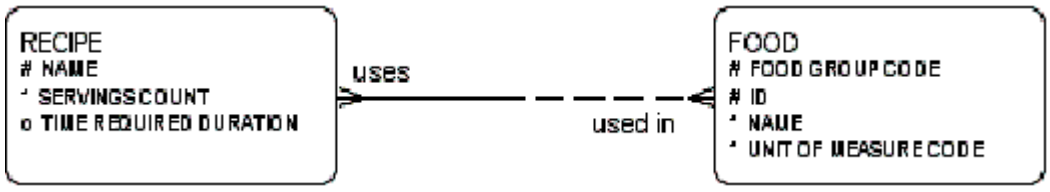


Figure C-5: Example 2 - Template Model

The template model shown in Figure C-5 has a Many-to-Many relationship. The target model adds interesting detail by resolving it, as shown in Figure C-6. The relationship optionality is not changed: RECIPEs must use FOODs, but FOODs need not be used in RECIPEs.

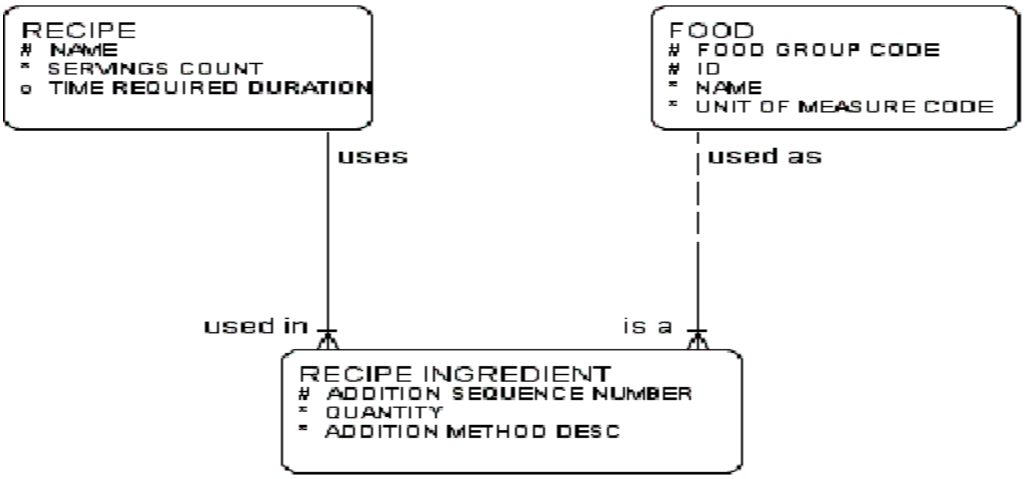


Figure C-6: Example 2 - Target Model

C.4.2 Non-Identifying Attribute Usage Rules

Modelers may create new attributes in their target models.

If an attribute always has the same value in the target application, either:

- Include the attribute in the target model. This may be informative, and it may be necessary if the attribute is likely to be interfaced to other systems, **Or**
- Document the attribute name and value in the entity definition, and omit the attribute, (even if it is mandatory).

Attributes found in the template model are designated as either mandatory or optional to show whether the attribute information is an essential descriptor of an entity. The template model does not determine whether a particular enterprise or project requires certain information, so many attributes are left optional.

- Modelers normally must include mandatory template attributes. Any exceptions will be specified in the template model usage rules.
- Modelers normally may include or omit optional template attributes. Any exceptions will be specified in the template usage rules.
- Attributes should not be included unless the information is essential to the enterprise or project. If this principle conflicts with the mandatory/optional nature of an attribute, and the template model usage rules, follow the template's support procedure.
- Modelers may make an optional attribute mandatory but not vice versa.

Before leaving an attribute optional in a data model, consider whether there are business rules that should be modeled or documented, to define when the attribute is mandatory and when it is not used. Related entities or subtypes may be required.

Application designers may simplify by combining multiple template attributes into one field on a form, screen, etc. The system must automatically or manually parse the resulting data into template-compliant attributes.

C.4.3 Domain Usage Rules

Attributes in the logical level template model may have specifications for data type, length, allowed values and validation rules. These may be specified as properties of the attribute, or they may be part of a global domain. The following rules apply to the logical level data model, whether the domain specification is global or local to the attribute. Note that domain specifications may be adjusted in physical models for implementation reasons.

- If an attribute in the template model has a domain specification, that domain must be used in the target model.
- If the template does not specify attribute domains, the modeler may provide their own domain specifications.
- Modelers must not substantively change the data type of a domain. (Changes are permitted for technical reasons, like using data types available in a particular DBMS, or changing CHAR to VARCHAR).
- Modelers must not substantively change the allowed values of a domain:
    - Do not add or delete allowed values. This may affect the meaning of the entire attribute.
    - If the template model does not specify an allowed value for “unknown”, “not applicable”, etc., it must not be added, because this could effectively make a mandatory attribute optional.
    - Allowed values may be modeled and stored as short codes or longer descriptive meanings, as long as the original meaning from the template is substantially preserved. There must be a documented mapping from the template model to the target model’s allowed values, if any changes are made.
    - Allowed values may be indicating allowed values in the definition.
    - Allowed values may be implemented in any suitable fashion, such as lookup tables or one large code table.
    - The attribute or domain definition may refer to a central repository providing the allowed values; this practice is recommended.

Validation instructions specified in an attribute or domain definition are recommended as a check on data quality. They may be changed in a target model.

Some attributes have standardized domains. For example, a Canadian postal code is always 6 characters plus a space, in an ANA NAN pattern. For other attributes, the template model may recommend a length (for example, individual first names can have

any length, so the template length is arbitrary). Attribute and domain definitions must specify “Length is arbitrary” if the domain is not standardized.

- Modelers must not change the maximum length or decimal places of a standard domain. (This is the default, if the attribute definition does not state “Length is arbitrary”.)
- The maximum length and decimal places of an arbitrary domain are recommended. See the section below re Transition. Database size should not be a deciding factor in setting attribute lengths in a logical model. Use variable length data types where appropriate.
- Modelers may change the average length of domains, if it can be predicted from existing data.

Transition to Standard Lengths

Where the template model sets an arbitrary attribute length, there will be transition problems when interfacing to a system using a different length. New systems may receive or send data to multiple existing systems.

If the source attribute from a new or legacy system is longer than a destination attribute, some data will get truncated in the destination database. It is not obvious whether a new system's attribute length should match the longest possible source, or the shortest possible destination.

It may be tempting to accommodate a new system's major data source or destination. But can we assume that the source data will always be in its current format? Will the new system never be used as a source for another database? What new formats might need to be accommodated?

As systems adopt the template standards, there will be less need to accommodate different formats. Note: The Common Data Elements Model (CDEM) errs on the long side when setting arbitrary attribute lengths.

When interfacing a new system to an existing database, determine if any destination attributes are too short to accommodate source attributes. Determine if any source data values are actually long enough to be truncated. If so, here are some implementation options:

- Allow display, but not input, of longer values from an existing system with a longer attribute length.
- If the new system is collecting data that will be sent to an existing system with a shorter attribute length, there are two options:
    1. Restrict data input to the shorter length in the application user interface. Store that data in the standard-length attribute.
    2. Collect new data using the standard attribute length. If appropriate, warn users that data will be truncated by the destination system.
- Apply matching algorithms only to the first N characters, where N is the shortest attribute length.

C.4.4 Unique Identifier Usage Rules

The logical level template model shows unique business identifiers when they are common across the enterprise. They may be identifier attributes or identifying relationships, and they are designated as part of an entity's unique identifier.

Sections C.4.1 Non-Identifying Relationship Usage Rules and C.4.2 Non-Identifying Attribute Usage Rules do not apply to unique identifiers. Section C.4.3 Domain Usage Rules provides rules about the domains of attributes, which do apply to attribute identifiers.

For identifying relationships, if a child entity is included in the target model, its parent entity must be included to include the entire unique identifier of the child.

Modelers must **not** replace a business identifier in the template model with a different business identifier in their logical level target model. This will ensure that the OPS uses common identifiers for common concepts.

- If the enterprise or project requires a legacy identifier, it may be added as a non-identifying attribute.
- Business identifiers may be replaced with artificial keys in the physical data model.
- If the template model does not include a unique business identifier, the modeler must provide one in their target model.
- Identifiers are mandatory attributes or mandatory relationship ends. They must not be changed to optional in the target model.
- The non-identifying end of an identifying relationship may be changed from optional to mandatory, but not vice versa.
- The cardinality of an identifying relationship may be reduced from many to one, but not increased.
- One-to-One relationships may be left as-is, or the entities may be collapsed together (see section C.3.4 Collapsing One-to-One Relationships).
- If an identifying attribute always has the same value in the application, either:
    - Include all parts of the identifier in the target model. This may be informative, and it may be necessary if the attribute is likely to be interfaced to other systems, or
    - Document the attribute name and value in the entity definition, and omit the identifying attribute. Part of the composite identifier is now visible only in the definition.
- If only one occurrence of the parent entity (in an identifying relationship) is in scope, either:
    - Include the parent entity in the model. (This may be informative, and it may be necessary if the entity is likely to be interfaced to other systems.), or

- Document the parent entity name, attribute names and values in the child's entity definition, and omit the parent entity. Part of the composite identifier is now visible only in the definition.

For example, the definition of RECIPE in the target model may read "The <*template model*> has an identifying relationship from RECIPE to RECIPE CATEGORY. In this application, there is only one RECIPE CATEGORY. Its identifier CATEGORY CODE = 03 is part of the identifier of RECIPE. The value of CATEGORY NAME is "Dessert".

Non-identifying attributes of a parent entity may have been omitted according to the rules in section C.4.2 Non-Identifying Attribute Usage Rules. If only the unique identifier of the parent entity remains, it may be collapsed into the child entity, becoming an identifying attribute.

For example, in this template model, CAMPERs are identified partly by the CAMP they go to:

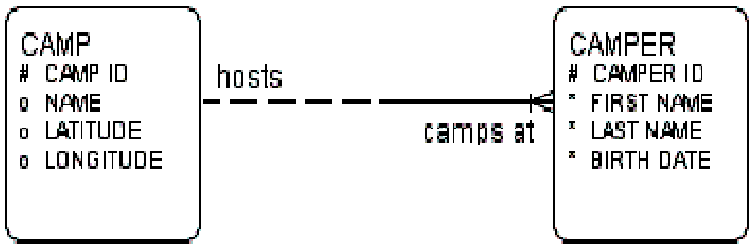


Figure C-7: Example 3 - Template Model

The target application is interested in CAMPERs but not CAMPs. In the target model the identifier from CAMP will be collapsed into CAMPER:

Figure C-8: Example 3 - Target Model

Appendix D – References

Reference Document	Date of Issue
BC Forest Service, Information Management Group. *Data Modeling Guide – Guide S7*. Ministry of Forests, Information Management Group http://www.for.gov.bc.ca/his/datadmin/s7.pdf	April 2000
Booch, Grady and James Rumbaugh and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley	1999 [ISBN 0201571684]
Claxton, John C. and Peter A. McDougall. *Measuring the Quality of Models*. The Data Administration Newsletter (TDAN.com)	October 2000 Issue 14.0
Carlson, David. *Modeling XML Applications with UML*, Addison-Wesley.	2001 [ISBN 0-201-70915-5]
Han, Jiawei and Mickeline Kamber. *Data Mining: Concepts and Techniques*, Chapter 2. Data Warehouse and OLAP Technology. Morgan Kaufmann Publishers	August 2002
Hay, David C. *A Comparison of Data Modeling Techniques*. Essential Strategies, Inc. http://www.essentialstrategies.com/publications/modeling/compare.htm	October 1999
Hay, David C. *Data Model Patterns: Conventions of Thought*. Doreset House	1996 [ISBN 0-932633-29-3]
IBM. *Class Diagram, Work Product Description*. IBM Global Services	January 2000 Version 3.0 [APP 302]
IBM. *Business Object Model, Work Product Description*. IBM Global Services	January 2000 Version 3.0 [APP 107]
IBM. *Enterprise Information Model, Work Product Description*. IBM Global Services	January 2000 Version 3.0 [ARC 307]
IBM. *Logical Data Model, Work Product Description*. IBM Global Services	January 2000 Version 3.0 [APP 110]
IBM. *Subject Area Model, Work Product Description*. IBM Global Services	April 2000 Version 2.0 [BUS 331]
Imhoff, Claudia. Galemmo, Nicholas. Geiger, Jonathan. *Mastering Data Warehouse Design: Relational and Dimensional Techniques*, Wiley	August 2003 [ISBN-10: 0471324213]

Reference Document	Date of Issue
Inmon, W.H. *Building the Data Warehouse* (third edition), Wesley	March 2002 [ISBN-10: 0471081302]
iSERV Ontario, *CORPORATE SECURITY Information Security Classification Policy*	January 2002 Version DRAFT
ISO/IEC 11179 Information Technology Metadata Registries (MDR) Standards, 3rd Edition, http://metadata-standards.org/11179/	March 2006
Kimball, Ralph. Reeves, Laura. Ross, Margy. Thornthwaite, Warren. *The Data Warehouse Lifecycle Toolkit, Expert Methods for Designing, Developing, and Deploying Data Warehouses*. Wiley Computer Publishing.	1998 [ISBN 0-471-25547-5]
Kimball, Ralph and Margy Ross. *The Data Warehouse Toolkit, The Complete Guide to Dimensional Modeling*. Wiley Computer Publishing.	2002 [ISBN 0-471-20024-7]
Ministry of Environment, *Environet Data Modeling Standards*.	March 2001
Ministry of Government Services, *Government of Ontario Information Standard No. 27 - XML Family of Standards, Version 1.2*, http://www.gov.on.ca/MGS/en/IAndIT/STEL02_047303.html	March 2005
Ministry of Government Services, *Government of Ontario Architecture Standard No. 56 – OPS Enterprise Architecture Principles and Artifacts, Version 1.4*, http://www.gov.on.ca/MGS/en/IAndIT/STEL02_047303.html	December 2009
Ministry of Government Services, *Guide to Abbreviating Data Object Names, Version 1.0,* http://intra.net.gov.on.ca/iit/services/enterprise-architecture/domains/#information	January 2011
Ministry of Government Services, *Guide to Architecture Transformation in EA Framework, v1.0*	March 31, 2006
Ministry of Government Services, *Guide to Data Model QA, Version 1.0,* http://intra.net.gov.on.ca/iit/services/enterprise-architecture/domains/#information	July 2012
Ministry of Government Services, *Guide to XML Schema Design Styles and Transformation, Version 1.0,* http://intra.net.gov.on.ca/iit/services/enterprise-architecture/domains/#information	December 2009
Ministry of Government Services, *Information Architecture Review Questionnaire, Version 1.0,*	July 2012

Reference Document	Date of Issue
http://intra.net.gov.on.ca/iit/services/enterprise-architecture/domains/#information	
MTO. *Active Visual Process*. EDS Systemhouse Inc.	1993-2000 EDS Systemhouse Inc. Release 1.5
Muller, Robert J. *Database Design for Smarties, Using UML for Data Modeling*. Morgan Kaufmann Publishers, Inc.	1999 [ISBN 1-55860-515-0]
Naiburg, Eric J. and Robert A. Maksimchuk. *UML for Database Design*. Addison-Wesley	July 2001 [ISBN 0-201-72163-5]
OASIS CIQ International Standards (xAL, xNL, xNAL, xCIL, etc.), Version 2.0, http://www.oasis-open.org/committees/ciq/ciq.html	2004
Rational Software White Paper *Mapping Object to Data Models with the UML*.	March 2000
Reingruber, Michael and William W. Gregory *The Data Modeling Handbook*, John Wiley & Sons, Inc.	December 1994 [ISBN 0-471-05290-6]
Silverston, Len. *The Data Model Resource Book,* John Wiley & Sons, Inc.	2001 [ISBN 0-471-38023-7]
UML and XML Schema, a research paper by Nicholas Routledge, Linda Bird and Andrew Goodchild, Thirteenth Australasian Database Conference (ADC2002), http://crpit.com/confpapers/CRPITV5Routledge.pdf	Australian Computer Society, Inc., 2001 and Thirteenth Australasian Database Conference, 2002
W3C XML Schema Standard Recommendation Version 1.0, http://www.w3.org/TR/xmlschema-1/	October 28, 2004
XML Schemas: Best Practices, Tutorial on XML Schemas, xFront, http://www.xfront.com/BestPracticesHomepage.html	February 17, 2003

