**Transport Canada Data Administration Metadata Standards**

**TP 14435 E**

**Transport Canada Information Management**

**May 2005**

**Table of Contents**

* [1. Introduction](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n1)
	+ [1.1. Audience for the Transport Canada Data Administration Metadata Standards](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n1.1)
	+ [1.2. How to Use the Transport Canada Data Administration Metadata Standards document](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n1.2)
	+ [1.3. Scope of the Transport Canada Data Administration Metadata Standards](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n1.3)
	+ [1.4. Background](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n1.4)
	+ [1.5. Transport Canada Data Model](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n1.5)
* [2. Database Business Rules](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n2)
* [3. Application Business Rules](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n3)
* [4. TC Data Modeling Standard – Definition and Concepts](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4)
	+ [4.1. Data Definitions Standard](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.1)
		- [4.1.1. Data Definition Rules](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.1.1)
		- [4.1.2. Data Definition Guidelines](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.1.2)
	+ [4.2. Entity/Table](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.2)
	+ [4.3. Subtype](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.3)
	+ [4.4. Attribute/Column](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.4)
	+ [4.5. Data Subject Area Information Holdings Categories and Sub-Categories](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.5)
	+ [4.6. Data Subject Area Information Holdings Sub-Categories](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.6)
	+ [4.7. Relationship](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.7)
	+ [4.8. Constraints](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.8)
		- [4.8.1. Subtype Constraints](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.8.1)
		- [4.8.2. Column Constraints](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.8.2)
		- [4.8.3. Relationship Constraints](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.8.3)
		- [4.8.4. Inter-Relationship Constraints](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.8.4)
		- [4.8.5. Procedural Constraints](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.8.5)
	+ [4.9. Other Considerations](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.9)
		- [4.9.1. Abbreviations](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.9.1)
		- [4.9.2. Data and Process Modeling Tools Limitations](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.9.2)
		- [4.9.3. Data Modeling Tools Features](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n4.9.3)
* [5. Transport Canada Function and Process Models](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5)
	+ [5.1. Function Model and Decomposition](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1)
		- [5.1.1. Levels of Decomposition](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.1)
		- [5.1.2. Where to Stop at the Decomposition](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.2)
		- [5.1.3. Hint: Decomposition Limit via Naming](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.3)
		- [5.1.4. Hint: The 10-minute Rule](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.4)
		- [5.1.5. The "Right" Way vs. The "Wrong" Way](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.5)
		- [5.1.6. Data Model makes it Formal](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.6)
		- [5.1.7. Function Naming](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.7)
		- [5.1.8. Process Modeling Tools Limitations and Naming](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.8)
		- [5.1.9. Function Description Narrative](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.9)
		- [5.1.10. Associating Data with the Functions](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.1.10)
	+ [5.2. Business Process Models](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.2)
		- [5.2.1. Some important rules for Processes](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.2.1)
		- [5.2.2. Process Modeling](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.2.2)
		- [5.2.3. Dataflow Diagrams](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.2.3)
	+ [5.3. Process Modeling Tool's treatment of Functions and Processes](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n5.3)
* [6. Applying Transport Canada Data Models](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6)
	+ [6.1. System Information](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.1)
	+ [6.2. Structured Data – Data Models](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.2)
	+ [6.3. Unstructured Data – Format and Definitions](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.3)
	+ [6.4. Capture Data Models using Data Modeling Tools](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.4)
		- [6.4.1. Legacy Systems](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.4.1)
		- [6.4.2. New Applications](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.4.2)
		- [6.4.3. Data Models – Nominal and Minimal Set](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.4.3)
		- [6.4.4. Source System Data Models – Configuration Management](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.4.4)
	+ [6.5. Attribute and Data Element Definitions](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.5)
	+ [6.6. Attribute Naming Standard](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n.6.6)
	+ [6.7. Attribute Definition Standard](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.7)
	+ [6.8. Source System Data Elements](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n6.8)
* [7. Transport Canada Data Warehouse Models and Methodology](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7)
	+ [7.1. Integrated Data Warehouse Collector Data Model](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.1)
	+ [7.2. Data Warehouse Implementation Model](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.2)
	+ [7.3. Data Warehouse End User Business Output Product Data Models](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.3)
	+ [7.4. Data Warehouse – Transformation Mappings and Rules](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.4)
	+ [7.5. Data Warehousing Mapping Framework](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.5)
	+ [7.6. Data Warehouse Mapping Source System to Staging areas](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.6)
	+ [7.7. Data Warehouse Mapping Staging areas to Data Warehouse](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.7)
	+ [7.8. Data Warehouse Data Filtering, Data Transformation rules](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.8)
	+ [7.9. Data Mappings from Source to Transport Canada Data Warehouse (DW)](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.9)
	+ [7.10. End User Requirements – Data Mappings from Data Warehouse to Data Marts](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.10)
	+ [7.11. Data Warehouse Cleansing and Scrubbing Rules](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.11)
	+ [7.12. Data Warehouse Load Control Statistics](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.12)
	+ [7.13. Data Warehouse Audit and Control Metamodel](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.13)
	+ [7.14. Identifying Data Warehouse Loads](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.14)
	+ [7.15. Data Warehousing Collecting Load Statistics](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.15)
	+ [7.16. Data Warehousing Automating and Tracking Data Cleansing/Data Quality Issues](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.16)
	+ [7.17. Data Warehousing Additional User Friendly Support Tables](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.17)
	+ [7.18. Data Warehousing Individual Rules Disabled for Selected Runs](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.18)
	+ [7.19. Data Marts](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#n7.19)
* [Annex A – TC Metadata Model](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#annexa)
* [Annex B – Data Model Formalisms](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#annexb)
* [Annex C – Data Warehouse Audit and Control Tables](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#annexc)
* [Glossary and Acronyms List](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#glossary)

**1. Introduction**

The assumptions used when reviewing the model is that the user requirements have been captured correctly and completely and are represented in the database design. It is not the role of the Data Management Infostructure Group to verify that the model meets the user’s requirements.

**1.1. Audience for the Transport Canada Data Administration Metadata Standards**

This document is technical in nature and defines the Transport Canada Data Administration Standards for the development of Data Models that will be used by Application Developers and Maintainers, Data Architects, Database Administrators, Data Administrators, Data Warehousing project teams etc.

**1.2. How to Use the Transport Canada Data Administration Metadata Standards document**

This Transport Canada Data Administration Metadata Standards document has been divided into 9 Sections for ease of use. Within the document, reference to a specific suite of tools has been generalized as much as possible. Please see the Transport Canada Data Administration Metadata Framework document for the background on the tool suite selected.

**Section 1: Introduction** lays the foundation for what the Transport Canada Data Administration Metadata Standards are trying to accomplish with a brief discussion of the Transport Canada Data Administration Metadata Framework and how it relates to the application development process. In addition, it includes the base rules for Data Modeling in Transport Canada.

**Section 2: Transport Canada Data Modeling Standard - Definitions and Concepts** defines the industry standards for Data Modeling terms and definitions (entity, relationships, constraints) as they are used in Transport Canada. In addition, the Standards and Guidelines for Data Definitions is documented.

**Section 3: Transport Canada Process Models** identify what Process Models are, where they are used, and provides helpful hints / guides and rules on how to create Transport Canada Process Models.

**Section 4: Transport Data Models** identifies the Data Modeling requirements for various types of Transport Canada systems (New Systems, Legacy Systems, Structured Data, Unstructured Data etc.) and the associated strategies and rules for the creation, capture and maintenance of the Transport Canada Metadata Repository.

**Section 5: Transport Canada Data Warehouse Models and Methodology** describes and standardizes the Data Model driven approach to Modeling, creation, staging, extract, transform and loading (ETL) of the Transport Canada Data Warehousing and Data Marts environments.

**Section 6: Annex A – TC Metadata Model** defines the TC Metadata Model using Data and Process Modeling Tools. The metamodel incorporates entities, attributes and relationships that constitute the Transport Canada metamodel.

**Section 7: Annex B – Data Model Formalisms** - Annex B is the base standard document of the Transport Canada Data Administration Metadata Standards and provides a template for naming standards and conventions. These standards are the descriptive properties for the Metadata objects in the Data Models for the tables and columns, including primary key constraints, unique constraints, check constraints, assertions, procedural constraints, default values, and derivation rules. The purpose of the Annex is to enable standardization of Table and Column descriptive properties within Data Modeling Tools, so that they are consistently represented.

**Section 8: Annex C – Data Warehouse Audit and Control Meta Model**

Extensions to Information Schema, lists the audit and control tables required for tracking and controlling the loads, data cleansing and quality enforcement processes.

**Section 9: Glossary and Acronym List** contains acronyms and the associated description.

**1.3. Scope of the Transport Canada Data Administration Metadata Standards**

Please note the Metadata Standards deals with Application Development, Application Maintenance, Information Needs Assessments, Information Management Plans, and Data Warehousing. Transport Canada will address other Metadata in concert with Treasury Board who are addressing Metadata standards in other disciplines such as Records Management, Library Management, Web Searches, Web Content Management, etc.

**1.4. Background**

There are four documents, which deal with Transport Canada Data Administration. They are:

1. The “[The Transport Canada Data Administration Metadata Framework](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-framework-632.htm)”;
2. This document the “Transport Canada Data Administration Metadata Standards”;
3. The “Safety and Security Metadata Strategy”;
4. [Data Administration – Quick Reference Guide](http://www.tc.gc.ca/eng/corporate-services/imit-reference-guide-633.htm).

The Transport Canada Data Administration Metadata Framework lays down the foundation for implementing Data Administration principles by putting in place a data modeling approach based on International Standards and common business processes. This results in the creation of a corporate data model that is stored in the corporate Metadata Repository. This Metadata Repository allows for information sharing, standardization, reusability, and data integrity.

The Transport Canada Application Management Framework (AMF) and the Macroscope System Development Life Cycle define how applications will be developed in Transport Canada. One of the key components of application development is Data Modeling. Data Modeling forms the basis for a significant amount of application development and maintenance activities.

The Metadata Standards are the standards that describe how Data Models are built at Transport Canada. Transport Canada has created a Metadata Repository using Data and Process Modeling Tools. These tools are used to develop the content of the Metadata Repository. The standards apply regardless of the Computer Aided Software Engineering product Data and Process Modeling Tools used.

The Metadata Standards is applicable across all information systems. For example, in the early stages of project initiation, user requirements and information needs analysis a conceptual model is created to identify the high-level data entities. As the project proceeds and more analysis is undertaken, a logical model is developed building on the work completed in the conceptual model. The logical model is further broken down into the physical data model and Database Schemas used to build Databases. The consistent use of the Metadata standards will result in improvements and efficiencies in resource utilization and overall costs. The standards are technology independent allowing for technology changes in the future.

The Metadata Standards are a living entity that will be expanded on as new requirements or information are identified (i.e. Records Management, Library, Web Searches etc.).

**1.5. Transport Canada Data Model**

The Transport Canada’s Metadata Repository is a representation of the information needs of Transport Canada.

There are two types of business rules, each having their own definition as follows:

**2. Database Business Rules**

Business Rules at the database level include cardinality, uniqueness, referential integrity and mandatory fields. These are applied to the database at the table level by primary key, foreign key, unique key and check constraints. The focus of the business rules at the database level is to ensure data quality, integrity and consistency.

**3. Application Business Rules**

Business Rules at the application level may include the database business rules and in addition includes rules governing business workflows, business components, business entities and screen flow navigation. Example: If the inventory levels for Product X go below 50 units, coordinate with the supplier to purchase 100 additional units in order to replenish the stock.

Note: any time business rules are referenced in this document it refers to primary keys, foreign keys, unique keys and constraints. The primary keys (foreign keys, unique keys and check constraints etc.) are implemented at the database level where feasible. Complex business rules can be at various application levels or at the database level (i.e. functions, procedures, packages, triggers etc.) should it be more practical/efficient to do so.

The Transport Canada Data Models will consist of Entities/Tables, Attributes/Columns, Relationships and Business Rules that represent the information needs and requirements for Transport Canada to perform its functions at the current time and into the future.

*The Transport Canada Metadata Repository shall include the Logical and Physical Data Models. Though not required, it may be of benefit to the application development team to generate the business flow, process flow and data flow models using the AllFusion Process Modeler. These business process models are compatible with the Logical and Physical Data AllFusion models as well as the model repository.*

There are other diagramming techniques for process modeling which can form the basis for other diagrams i.e. Use Case Diagrams, UML related Diagrams, VISIO Diagrams driving Biztalk services etc. These diagramming techniques can be used but are not compatible with the model repository.

Data Model normalization to at least to the 3rd normal form is mandatory. Application databases will be generated from the Metadata Repository only. The models represent an integrated set of functions, data, processes and business rules, without regard to boundaries of current systems, external interfaces or functional areas.

**4. TC Data Modeling Standard – Definition and Concepts**

The following sections document the Transport Canada Data Modeling definitions and concepts.

**4.1. Data Definitions Standard**

The Data Definitions Standard specifies rules and guidelines for constructing definitions for data elements and for formulating definitions for other types of data constructs such as tables, relationships, columns, object types (or classes), objects, segments, code entries, etc. Definitions shall be written to facilitate understanding by users, and by recipients of shared data.

**4.1.1. Data Definition Rules**

**A data definition shall:**

1. Be stated in the singular
2. State what the concept is, not only what it is not
3. Be stated as a descriptive phrase or sentence(s)
4. Contain only commonly understood abbreviations
5. Be expressed without embedding definitions of other data or underlying concepts

**4.1.2. Data Definition Guidelines**

**A data definition should:**

1. State the essential meaning of the concept
2. Be precise and unambiguous
3. Be concise
4. Be able to stand alone
5. Be expressed without embedding rationale, functional usage, domain information, or procedural information
6. Avoid circular reasoning
7. Use the same terminology and consistent logical structure for related definitions.

**4.2. Entity/Table**

An entity/table is something of interest to the Transport Canada business. It is also the name given to a group of related data attributes/columns. All Entities/Tables must have a name and a definition. Each core entity/table is defined once in the repository, and may be used in more than one subject area.

Entity/Table names shall be singular meaningful nouns. For example the Entity 'EMPLOYEE' is defined as the set of all employees in the organization.

**4.3. Subtype**

An Entity/Table is a Subtype of another Entity/Table when all it’s instances are a subset of the instances of the other Entity/Table. A Subtype Entity/Table inherits all the properties of its Super-type.

A Subtype must be defined as an Entity/Table. Furthermore the link between the Subtype and the Supertype must be defined (normally by an identifying attribute in the subtype).

Example of a Entity:

‘PERSON’ is a subtype of ‘PARTY’ and inherits the attributes of ‘PARTY’.

Example of a Table:

‘AC001\_PERSON’ is a subtype of ‘TC001\_PARTY’ and inherits the attributes of ‘TC001\_PARTY

**4.4. Attribute/Column**

An Attribute/Column is a class of data ascribed to an entity/table. It may also be known as a data element. Attributes/Columns define properties of a single entity/table.

The logical model does not require the table name prefix or underscores between words. Only attribute and verb phrase relationship displays are required on the logical model. This view can be used for model reviews with business clients.

Physical models will convert the spaces contained in the logical to an underscore. The physical model requires each table to start with a prefix from the category and subcategory tables followed by a three-digit number. Synonyms can be entered in the model for each table.

If during the modeling exercise it becomes clear that an Attribute/Column defines properties of more than one Entity/Table, then that Attributes/Column shall be modeled as an Entity/Table. (Note that this is part of the process of Normalization (removing repeating groups, part key dependencies, data dependencies) which will be conducted to derive the Logical/Physical Models.)

Attributes/Columns must have a definition. It is recommended that an example be given for each attribute.

All attributes/columns must have a data type in the Logical/Physical model. However it is not mandatory to enter a data type for the initial modeling process.

Attribute/Columns names must be a singular meaningful name.

Example:

‘Surname’ and ‘Given Name’ are examples of Attributes of the Entity ‘PERSON’. ‘Jones’ and ‘Smith’ are examples of values for Attribute ‘Surname’.

**4.5. Data Subject Area Information Holdings Categories and Sub-Categories**

A Subject Area Category is a high-level data classification in Transport Canada that represents a specific business need. An examination of over 200 Transport Canada information holdings resulted in the *classification* of a list of information holding categories that were involved. The documentation of the Data Model concepts, and in particular Entities and Attributes, must indicate to which Subject Area(s) they apply.

Table 1 lists the set of information holding categories.

| **Table 1 – Holding Categories**  |
| --- |
| **Category**  | **Information\_holding\_cateogory\_txt**  | **Definitions**  | **Remarks**  |
| A  | Administration  | Tables containing TC items. Ex. Users, Buildings, Organizations, HR Positions.  | (HR, Personnel, Organization)  |
| B  | Bulletins  | Federal Transportation Bulletins eg. Dangerous Goods, Security notifications.  |    |
| C  | Regulatory Compliance  | Processes or procedures involving conformity for fulfilling official rules, regulations and requirements.  | (Inspection, Maintenance)  |
| E  | Emergency & Response  | Emergency Preparedness, Situation Center.  | (Emergency Response Plan)  |
| F  | Financial  | TC financial Systems Ex. IDFS, SMS, BIRM.  |    |
| G  | Investigation  | TSB accidents, incidents, TC special investigations.  | (Accidents, Incidents, Bird Strikes)  |
| K  | Specifications  | Technical specifications like weight, height.  | (MMEL, Aircraft Details, container specs)  |
| L  | Training  | Schedules, courses, instructors, training materials.  | Learning  |
| M  | Analysis & Reporting  | Train miles, hours flown, vessel km, resource tracking.  | (Usage, Activity, Tracking, Difficulty reports)  |
| P  | Publications, Manuals, Schedules  | TC publishing and library facilities.  |    |
| U  | Qualification & Certification  | Conditions or standards that need to be complied with.  | (Licensing, Medical)  |
| R  | Regulations & Standards  | Applicable criteria, specification, rules, or directives e.g. air directives, Air policy.  |    |
| S  | Surveys & Studies  | Examination, statistical study (usually in a sample) or assessment towards some established criteria or requirements.  |    |
| T  | Table Reference  | Look up tables.  |    |
| W  | Work Tables  | Usually used for data conversion tracking system to system, logging activities and regular temporary working tables.  |    |
| X  | External System Interface  | External representation of data or data structures being interfaced with or used within a system e.g. source table extraction for purposes of transfer and loading from a system outside of Transport Canada.  |    |
| Y  | Registry  | List of items important to business.  | (Ship, Aircraft)  |
| Z  | Extended Metadata  | Extended Control Metamodel used for Extraction, Transform and Load Processes.  | Multimodal  |

Note: For system development the following letters are not to be used I, O, Q in table prefix identification, as they are easily misinterpreted.

**4.6. Data Subject Area Information Holdings Sub-Categories**

A Subject Area Information Holdings Sub-Category is a high-level data classification in Transport Canada that represents specific modal characteristics. The documentation of the Data Model concepts, and in particular Entities and Attributes, may indicate to which Sub-Category area(s) they apply to.

Table 2 lists the set of information holding sub-categories, which must be, used in conjunction with the Subject Area Categories.

| **Table 2 – Holding Sub-Categories**  |
| --- |
| **Sub-Category**  | **Information\_Mode \_txt**  | **Definitions**  | **Remarks**  |
| A  | Air  | All activities related to Air or Aviation  |    |
| B  | Communication  |    |    |
| C  | Corporate  | All departmental activities not related to another specific mode  |    |
| D  | Dangerous Goods  |    |    |
| F  | Finance  |    |    |
| H  | Human Resources  |    |    |
| M  | Marine  |    |    |
| R  | Surface  | Rail  |    |
| S  | Security  |    |    |
| T  | Surface  | Roads  |    |
| Y  | Multimodal  | Cross Modal  |    |

Note: For system development the following letters are not to be used I, O, Q in table prefix identification, as they are easily misinterpreted.

**4.7. Relationship**

A Relationship involves several Entities/Tables in an association. Only binary Relationships will be used, meaning Relationships that involve exactly two Entities/Tables. Every binary Relationship has two roles. A role is defined by the set of objects, part of an Entity/Table, participating in the Relationship.

Data Model Relationships represent closely the business information structure. Their role names must be meaningful and must be selected so as to allow users to easily read them in a ‘natural language’ like manner.

All Relationships must have documentation and it is recommended to provide instance level examples for relationships.

Entity Example:

An ORGANIZATION may employ zero, one or more EMPLOYEE(s)

An EMPLOYEE must work for one ORGANIZATION

In the above example, the role ‘works for’ between Entities ‘EMPLOYEE’ and ‘ORGANIZATION’ is defined as the set of all employees working for a ‘ORGANIZATION’. The role ‘employs’ between Entities ‘ORGANIZATION’ and ‘EMPLOYEE’ is defined as the set of all ‘Organizations’ that employ ‘Employees’. The Relationship can be read in natural language as follows: ‘ An Organization employs employees’ and ‘An employee works for an Organization’. An example at the instance level, with use of appropriate Attributes, would be as follows: ‘Jones works for Transport Canada'.

**4.8. Constraints**

Constraints are rules that limit the population of the Data Model.

Generally, subtypes are used when strong typing is desired. When data can be strongly typed, i.e. categorized into near static roles of a subset of the supertype population, then sub-types shall be used. E.g. 'AIRCRAFT' and 'SHIP' are partitions of 'CARRIER', with non-overlapping roles that can be considered relatively static in current technology.

For more dynamic roles of an entity, these could be modelled as subtypes but could be considered as being restricted. For example, ‘CUSTOMER’ and ‘PROSPECT’ are partitions of ‘PARTY’ and could be modelled as sub-types. But a ‘PROSPECT could change to be a ‘CUSTOMER’ over time. Also ‘PERSON’ and ‘ORGANIZATION’ could be modelled as sub-types of ‘PARTY’. Where valid combinations exist, these are better to be modelled as Classifications or Roles. For example, a ‘PROSPECT’ could be a ‘PERSON’ or an ORGANIZATION’.

Figure 1 below describes modeling Classification constraints in a Data Model.



Figure 1 Modeling Classification constraints in a Data Model

For entity name, a number of ‘entity name CLASS TYPE’s are defined. For example, if entity name is ‘PARTY’, then if ‘PARTY CLASS TYPE’ is ‘PERSON’ with values of ‘PARTY CLASS’ being ‘MALE’, ‘FEMALE’, then ‘PARTY CLASS CONSTRAINT’ could specify ‘Valid / or Invalid combinations’ of ‘PARTY CLASS’

This allows the Data Model to record the Constraints, and the Data Model would remain robust to the addition or removal of Classifications.

It is suggested that the Modeling of constraints in this fashion would be restricted to the repository and not be included in ER diagrams, as to do so would make them overly complex.

**4.8.1. Subtype Constraints**

A **Subtype Exclusion Constraint** indicates that the set of instances of one Subtype is disjoint from the set of instances of another Subtype of the same Entity/Table.

Entity Example:

A ‘Customer’ of Transport Canada is defined as an Organization or a Person that has been billed in the previous 3 months. Thus, ‘Organizations’ which are ‘Customers’ cannot be ‘Prospects’ and vice versa.

A **Subtype Totality Constraint** indicates that the union of instances of all the Subtypes involved in the constraint equals the set of instances of their common Supertype.

Entity Example:

Every ‘Person’ must be either a ‘Male’ or a ‘Female’. Note that a Subtype Totality Constraint does not automatically imply a Subtype Exclusion Constraint. In order to indicate that a ‘Person’ can’t be both ‘Male’ and ‘Female’ we would need the Subtype Exclusion Constraint.

**4.8.2. Column Constraints**

The use of column level constraints is not recommended due to lack of integra-ability and consistency. It is more advantageous to accommodate changes when the constraints are declared as table constraints. A constraint when declared at the column level, is in line with the column definition, and can generally only refer to that column only. If there is a constraint that refers to another column in the same table then the column level constraint has to be moved to a table level constraint. This requires many changes to the schema, which can be avoided by initially declaring all constraints at the table level as a table constraint.

The use of Erwin’s null option is permissible (“Not null” or “Null”) at the column level, as well, the use of the column level “valid value” constraint is also permissible at the column level.

An **Optionality Constraint** indicates whether every instance of the Entity must have a value for this Attribute or not.

A **Range Constraint** indicates the maximum and the minimum values of an Attribute.

A **Value Constraint** indicates the set of values to which the value of the Attribute must belong.

**4.8.3. Relationship Constraints**

A **‘Relationship Totality’ Constraint** is a constraint on the set of instances (role) of one of the Entities that participate in the Relation. It means that all instances of that Entity must participate in the Relationship, that is, the entity is mandatory.

(Mandatory is denoted by ‘must’, optional is denoted by ‘may’.)

A **Relationship Cardinality Constraint**

The cardinality of a relationship is specified as one-to-one, or one-to-many, or many-to-many

All Relationships must have Totality and Cardinality Constraints.

**4.8.4. Inter-Relationship Constraints**

**Mutually Exclusive Relationships**

In the following example, the Address Usage Type determines whether the relationship is Manufacturer to Address Usage, or Facility to Address Usage. These Relationships are said to be in a Relationship set.

In an alternate configuration shown in Figure 2, Address Usage Type could be modelled as a parent entity to Address Usage.



Figure 2: Example of an Alternate configuration in Modeling the Constraint

**4.8.5. Procedural Constraints**

Procedural Constraints are rules that involve Relationships, Subtypes and Column values in a constraint that requires navigation through the model.

The sum of employee salaries of a given department must be smaller than the budget of that department. This is an example of a procedural constraint.

**4.9. Other Considerations**

**4.9.1. Abbreviations**

Transport Canada has standardized spellings and abbreviations used in applications and databases. In addition, rules and guidelines have been defined to provide a standard method to prefix table names.

**4.9.2. Data and Process Modeling Tools Limitations**

Data and Process Modeling Tools have restrictions on Entity/Table Names to a maximum of specific character lengths dependant on the RDBMS. For example the maximum Oracle table name length is 30 characters. Therefore all Entity/Table Names must not exceed 30 characters and the Entity/Table Names shall be in upper case.

Attribute/Column names are subject to the same name length restrictions as Entity/Table names.

Attribute/Column names are also in upper case in Data and Process Modeling Tools.

**4.9.3. Data Modeling Tools Features**

Relationship Constraints in the Data Modeling Tools Entity Relationship diagrams are represented by:

1. a solid line indicates that the entity is mandatory in the relationship (indicated by the word ‘must’)
2. a dotted line indicates that the entity is optional in the relationship (indicated by the word ‘may’)

**5. Transport Canada Function and Process Models**

This section describes the Transport Canada Metadata Process Model components and their applicable standards, guidelines, process or procedures to be followed during their definition.

**5.1. Function Model and Decomposition**

A function is a business activity that contributes to the achievement of an objective of the business. Functions are arranged in a hierarchy. A function hierarchy displays the functions in a multilevel view.

The additional information that can be collected with a function is the data associated with functions, in terms of entities/tables and attributes/columns, along with a description.

Each function in the hierarchy is decomposed until the lowest level functions (called elementary business functions or processes) that comprise it have been identified.

The Process Modeling Tool's Function Hierarchy Diagram is used to decompose functions, define elementary functions and show how functions use data.

The objective is to record all the business functions. Functions may be automated ("Record Customer Order") or non-automated (e.g. "Assemble Products in a Box). Automated in this sense means whether a computerized application is being used to record information about the function.

**5.1.1. Levels of Decomposition**

The function model hierarchy describes the decomposition of the function elements.

The decomposition can generally proceed to 4 or 5 levels of decomposition, the top function being level 0. This is a self checking mechanism, where the (lowest level + 1) of a decomposition becomes a process that involves some form of input or output in an automated data processing system.

**5.1.2. Where to Stop at the Decomposition**

A function at the higher levels generally constitutes a business activity that can be carried on without the need or the existence of a computerized application. In other words, the function is purely a business function.

At some point during the decomposition, the function conceptually splits into an automatable function with part of it still being in the realm of conducting a business.

An example would be "Receive Incoming Shipments", which involves the physical receipt of goods (conduct of business part), and also involves entering the shipping notice or waybill details into the computer system (automatable part).

The next decomposition step of the automatable part may be "Register Waybill". This decomposition step involves some form of computer input / output, and is termed as a Process. It is important to note that this step still represents a complete logical unit of work, but in the data processing sense. A further splitting or decomposition of this step would not be a complete logical unit of work.

This last (or lowest) level is denoted as level "n", i.e. a process. The functional decomposition stops at level "n-1", i.e. the level when the function involves an "automatable part" and still maintaining a "purely business part", and where a further decomposition results in a purely automatable process involving computer facilities input / output, or, the consumption or generation of a product.

Examples at n-1 and n levels are:

| **Table 3: Examples of Functional Decomposition Lowest Levels**  |
| --- |
| **n-1 (business part + automatable)**  | **n (automatable only)**  |
| Register Vessel  | Enter Vessel Registration (in computer application)  |
| Initiate Pollution Incident  | Enter Pollution Incident (in computer application)  |
| Update Inspector Qualifications  | Enter Inspector Information (in computer application)  |

**5.1.3. Hint: Decomposition Limit via Naming**

A self-checking mechanism aid is to be aware of the names of the functions during decomposition. The names at the level "n" of decomposition generally will be of the form "Create" something, "Retrieve" something, "Update" something or "Delete" something---or their equivalents like Enter, Produce, Generate, Dispose, Remove, Erase etc. These names are telltale signs that the "n" level decomposition has been reached.

**5.1.4. Hint: The 10-minute Rule**

A given function may be decomposed in several ways. Also, a given function may have several alternate sub-functions, along with several alternate names that can be allocated to a given function or sub-function or process, i.e. the resulting decomposition may have different flavours and names. Apply the 10- minute rule. Do not spend more than 10 minutes on any exercise in decomposition or naming. The section below discusses the reasons why.

**5.1.5. The "Right" Way vs. The "Wrong" Way**

The procedures for defining functions and the procedures for decomposition may be done differently by different people. In other words, two or more persons defining functions and decompositions of the same business may arrive at different answers. This is quite acceptable and it does not matter, as long as all the functions are being covered.

**5.1.6. Data Model makes it Formal**

Why does this not matter? A function model is not a very formal model i.e. there is not a formal grammar to support the function model. What matters is the data that is to be identified in the data usages of Entities and Attributes. A data model is derived from the data usages. A data model is a formal model with formal grammar associated with it.

What this means, is that it does not matter how the functions are organized, as long as the data usages have been recorded. The output styles of function modeling will ultimately result in one formal data grammar or schema.

**5.1.7. Function Naming**

Names given to function consist of a combination of (Action verb + Noun), preferably leading with the action verb. Avoid connectors like "and", "&", "or" and the like, unless the functions really go together, e.g. Manage Training & Education or Manage Enforcement & Compliance.

Avoid the phrase "Process" as the action verb.

Quite often, at level 1 and 2, the names of the functions could be similar to some parts of an organization structure.

**5.1.8. Process Modeling Tools Limitations and Naming**

To accommodate long names, an abbreviated name shall be used as the function name.

The abbreviated name shall be of the format as below:

F <functional area designator> \_ <idnumber> \_ <abbreviated function name>

Where:

<functional area designator>::= || *subject area identifier, e.g. A, B, C…naming a subject area ||*

<idnumber> ::= || *1 digits as below:*
*Level 0 : <subject area identifier><-><idnumber>;e.g. F-0*
*Level 1: <subject area identifier><idnumber>; e.g. F0*
*Level 2: <subject area identifier><idnumber>e.g. F1, F2;*
*Level 3: <subject area identifier><idnumber><.><idnumber>, e.g. F1.1, F****1.2****;*
*Level 4: <subject area identifier><idnumber><.><idnumber><.><idnumber>, e.g. F1.1.****1****, F1.1.****2****, F1.1.3;*
*Level 5: <subject area identifier><idnumber><.><idnumber><.><idnumber><.><idnumer> e.g. F1.1.1.****1****, F1.1.1.2, F1.1.1.3||*
<abbreviated function name> ::= ||*verb + noun construct.||*

For the <functional area designator>, 4 major functional subject areas have been defined (in the INR Strategy, Safety & Security, October 2001): **R**egulations, **I**nfrastructure, **M**ission Critical Functions, and **O**perations Support.

| **Table 4**  |
| --- |
| **Functional Area Designator**  | **Major Function\_txt**  | **Definitions**  | **Remarks**  |
| F  | Infrastructure  |    |    |
| M  | Mission  |    |    |
| S  | Operations Support  |    |    |
| R  | Regulations  |    |    |

**Note: For system development the following letters are not to be used I, O, Q in table prefix identification, as they are easily misinterpreted.**

For the decomposition and naming, the top-level function (Level 0) is generally the organization name.

The 1st level is essentially the **main business activities** being performed by that organization (like directorates).

Level 2 is where the main activities are decomposed and so on.
E.g. Level 1 function: FR0\_Manage Transport Regulation
E.g. Level 2 function: FR1\_Manage Policy & Standards

**5.1.9. Function Description Narrative**

Each function shall be described briefly in the textual part of the functional definition. The narrative shall not include references to the level, hierarchy, and upper level or lower level functions. Rules for definitions, see Section 2.1.1 Data Definition Rules and 2.1.2 Data Definition Guidelines.

**5.1.10. Associating Data with the Functions**

The lowest level function shall have data associated with it. This will be documented in the corresponding and synchronized dataflow diagram.

It is mandatory that at least the entity usages be documented for the lowest level function ("n-1" i.e. not the process level). If available, Attribute Usage can be documented.

The usage is to be documented in terms of Create, Retrieve, Update or Delete of the Entities or Attributes used. Examples of data are to be recorded where available, in the Comments column in the Entity or Attribute Usage Tabs.

**5.2. Business Process Models**

Process Modeling provides a way to diagram the processes and flows of data to and from other processes and data stores.

Processes are at the "n" level function in the decomposition. Processes address the automatable part of the "n-1" function.

**5.2.1. Some important rules for Processes**

Processes must by definition, be able to address an elementary logical unit of work that cannot be split without losing its meaning. A process is a repeatable user task that can be activated to perform only one complete logical unit of work. A collection of elementary logical units in one process (as one logical unit) is not allowed.

It is necessary to treat each process as an independent unit of work, and not "connect" the processes via flows. In other words, the model must only contain independent elementary processes that operate with data stores. The inter-dependencies are modelled independently, in a mechanism that is external to the process model diagram. Performing the processes independently will insulate the model from any changes in sequencing.

Processes can call other processes that are common processes. For example, a process that is dedicated to performing a derivation algorithm (e.g. calculate the net stock across regions of a particular item of supply).

By restricting the processes being modelled to performing a single unit of work, or elementary task that cannot be split without losing meaning, all the processes are re-usable, and can be called by any other process. The elementary processes can be packaged in a sequence, independent of other processes. Thus, the sequencing within a package itself is insulated from other package sequences, because every process is a stand-alone, and thus is unaffected by a particular sequence.

**5.2.2. Process Modeling**

With the Process Modeling Tool (BPWIN), business processes can be modeled using three different diagrams and they are Business Flow Diagrams, Data Flow Diagrams and Functional Hierarchy Diagrams. All of these diagrams types can contain Processes, Stores, Flows, Outcomes (output product), events and Organizational Units.

**5.2.3. Dataflow Diagrams**

Using the Dataflow Diagrams in the Process Modeling Tool, processes are also modeled to depict the dataflows, data stores and external agents that interact with the process.

**5.3. Process Modeling Tool's treatment of Functions and Processes**

Process Modeling Tools or other materials do not provide any guidance in this area, and treat it as a matter of refinement and "awakening".

That is why, the earlier section on "Where to stop at the Decomposition" identifies the "n" level function as a process, and "n-1" level as a function, in the hierarchy of decomposition.

Assignment of data usages can be done for a function, as well as for dataflows. It is dangerous to assign usages to both, since the usage on the function is derivable from the sum data usages of the flows. Where flows exist, the data usages of the dataflows take precedence over the data usages of the functions. This can happen during various stages of the analysis where decisions are taken to only document functions and data usages, which later could be expanded to cover dataflows also.

**6. Applying Transport Canada Data Models**

**6.1. System Information**

The metadata for Transport Canada's operational systems, data sources and data targets (in terms of information on database sources and targets for the data warehouse), needs to be captured. Table 5 below shows some examples of metadata attributes to be captured for system information.

| **Table 5: MetaData Attributes – System Information**  |
| --- |
| **Meta Data Attributes**  | **Examples**  |
| File Attributes  | * Filenames
* File types
* File characteristics
* Aliases
* Volume metrics
* File path
* File location
 |
| File Versioning and updates  | * Versions
* Date last updated
* Frequency of update Frequency of refresh
* Volatility
 |
| Privileges  | * Data owner
* Authorized users
 |
| Indexes  | * Indexed columns
* End user access patterns
 |

**6.2. Structured Data – Data Models**

Many of the current information holdings are structured. Structured data is defined as data that is organized in some fashion that can be classed as being in an ordered collection and defined using some form of a definitional facility. In most cases, this data is accessible via some form of a manipulation language or facility. Structured data is usually available or produced in some form of ordered data processing formats like databases, or files that include file definitions of their data contents. Examples are: Databases, XML data, inverted files etc.

**6.3. Unstructured Data – Format and Definitions**

Unstructured data is defined as data whose format is a specially defined format like textual data or video data that is it is in essentially a "free format", and does not fall into an ordered collection. Examples are: Text, Paper Files, Written Reports like Manuals, Procedures, Official Airline Guide, Jane's etc., graphics, audio and video training materials, microfiche/microfilm materials, X-ray films, medical charts, etc.

Data Modeling Tools have no facilities at this time to capture information on Unstructured Data types. The metadata model and the extended Transport Canada Information Schema (TCIS) contain tables that define the format and content labels of unstructured data. This is done via declaring information holding as unstructured, and collecting associated format and unstructured type information. Examples of unstructured types are: Audio, Graphics - Animated/moving, Graphics - Still Image, Spatial, Text, Video + Audio. These types have been declared based on ISO SQL Multi-Media Standard (SQL/MM) classifications.

**6.4. Capture Data Models using Data Modeling Tools**

For those holdings where formal schemas exist, these need to be defined in their respective work areas. Work areas need to be established by each Transport Canada Functional Subject Areas or, preferably by Data Subject Areas, in Data Modeling Tools.

**6.4.1. Legacy Systems**

Currently, the majority of Transport Canada's information holdings are structured. It is necessary to capture the data models of as many information holdings as possible in the Model Repository. In this manner, the available documentation from the current holdings can be consolidated within Transport Canada.

It is not practical to spend time and resources to capture the schema information from "all" legacy systems. It is also not practical to re-design a fully encompassing model and an operational integrated database based on the entire suite of legacy systems. It is sufficient to capture schema information on selected applications or information holdings in the Metadata Repository.

The Data Warehousing approach recognizes that it is impractical to manage the information of an enterprise as a unified database. Instead, this approach extracts information from logically and physically distinct databases, integrates the information, and stores it in a large-scale "warehouse" database that allows read-only access to possibly non-current data. The extraction and integration processes are dependent on the creation of mappings from the schemas of individual source legacy databases to the schema of the warehouse.

The long-term objective of Transport Canada is to define a consolidated integrated warehouse model, and map the source legacy systems to the model. This is the ideal, which may not be feasible or realized in one iteration, and can only be built in pieces.

It is necessary that at least a minimal set of data models for corresponding Data Warehouse source system schemas be captured.

**6.4.2. New Applications**

Metadata with respect to all new applications, need to be captured in the Model Metadata Repository. It is required that all new applications shall conform to the standards defined in this document.

**6.4.3. Data Models – Nominal and Minimal Set**

The optimal set of data model constructs, of course, would be to capture a complete schema in the form of a data model in any representation of a formal grammar---ER, NIAM, or UML or their variants.

Thus, the **Nominal** set is defined as a data model that is captured using Data Modeling Tools, either via import mechanisms or, manually inputted. This includes documentation supporting all the constructs of entities, attributes, relationships, business rules, etc.

This luxury of being able to capture all the constructs, along with the associated time, is generally not available. However, for the identified source systems, there is a requirement to capture a minimal set for mapping and other purposes.

The **Minimal** set of (information in lieu of) data models shall consist of the following constructs:

1. Table names (Entity)
2. Column names (Attributes)
3. Column properties
	1. data type
	2. data size
	3. nullable (desirable)
	4. default (desirable)
4. Constraints
	1. Primary Key (if used)
	2. Unique Constraint (Alternate Key) (if used)
	3. Foreign key
	4. Column Constraints (if used)
	5. Table Constraints (if used)
	6. Triggers (if used)
	7. Any other editing rules, inter-table constraints etc. (if used)
5. Derivation rules (if used)

The capture of this minimal set of information must be input as "data” in the data model and defined in the metamodel within the repository.

**6.4.4. Source System Data Models – Configuration Management**

Once the data models from the source systems have been captured, it is necessary to establish ongoing maintenance procedures to ensure that changes being done in the source systems are reflected and applied appropriately in the captured data models in the Metadata Repository. It is also important to maintain an audit trail reflecting information on what was changed, when, by whom, and identifying each change request with a change identifier. These need to be also reflected in the repository versions.

**6.5. Attribute and Data Element Definitions**

Attributes in a data model are mapped to data elements or columns in physical realizations of databases or files.

Attributes and data elements have a defined format along with additional properties. These contain items like data type, data size, null ability, default etc.

In addition data elements or columns can be of a user defined data type, with defined semantics and associated methods for manipulation.

Data Modeling Tools are used to define the former, i.e. with the defined format and properties.

**6.6. Attribute Naming Standard**

In the absence of an attribute-naming standard, the observer may be led to multiple interpretations of the possible meaning of the term. The looseness of the terms used in the absence of a norm does not contribute to the determination of proper usage and meaning of the attribute.

The use of a class term in the form of a suffix associated with each attribute facilitates in human understanding as to what the attribute means.

For example:
Without class terms:
a) VENDOR
b) DELIVER\_TO

With the use of Class Terms:
c) VENDOR\_ID
d) DELIVER\_TO\_NM

In the first set of examples it is not clear whether the <vendor> is a vendor name or an identifier like vendor number. Same with <deliver to>, not clear whether this is a name, text, code or location address.

In the second set of examples, it is clear that a vendor is denoted by an identifier, and delivery is being made to someone or something with a name.

Attributes/Columns shall be suffixed with a class term to assist in identifying the kind of attribute.

For a list of class terms, see Annex B.

**6.7. Attribute Definition Standard**

Attributes shall be described as per the standard for data definitions, see Section 2.1.1 Data Definition Rules and 2.1.2 Data Definition Guidelines.

**6.8. Source System Data Elements**

Transport Canada has over 350 application systems and significant other information holdings. These 350 applications include both structured and unstructured information holdings, with the large majority being structured.

It is neither feasible nor practical to capture all the data elements from all the current information holdings. Instead, these are to be captured when the Transport Canada Integrated Data Warehouse Collector Data Model is being defined. In addition, available documentation or facilities may dictate the capture of the source system data elements to be at the full capture or minimal set capture level, where only the table/column names are captured in an extended information schema.

Mappings shall be established from the source system data elements to Data Warehouse attributes for those source systems of interest.

Data elements whose source is from current or legacy systems probably will not be conforming to any published general standard. These will have to be documented in their existing form and appropriately mapped when necessary to standard name conformant attributes/columns.

**7. Transport Canada Data Warehouse Models and Methodology**

The Transport Canada Data Warehouse Models shall facilitate the incorporation of maximal data cleansing and data quality checks.

**7.1. Integrated Data Warehouse Collector Data Model**

The Data Warehouse describes the tools, techniques and technologies necessary to describe a Data Warehouse Data Model and associated database(s), where information critical to the delivery of Transport Canada programs and services is stored. This information is gathered from various sources, both internal and external.

The Integrated Data Warehouse Collector Data Model shall be defined using the Data Modeling Tools and represent a subset of the Transport Canada data model. The Data Warehouse Data Model can be incrementally constructed via the proposed combination of Project-Function based approach, where selected projects drive the building of the data warehouse that address the targeted business functions.

The Integrated Data Warehouse Collector Data Model shall reflect the integration of the selected applications or functions into an optimally normalized (Third Normal Form, Boyce-Codd Normal Form, up to the Fifth Normal Form) model.

The following modeling constructs have been defined in Data Modeling Tools:

1. Entities
2. Attributes
3. Attribute properties
	1. data type
	2. data size
	3. nullable
	4. defaults
4. Relationships
5. Relationship properties
	1. Roles
	2. Cardinality
	3. Optionality
6. Constraints
	1. Primary Key
	2. Unique Constraint (Alternate Key)
	3. Foreign key
	4. Check Constraints
	5. Table (including column) Constraints
	6. Triggers
	7. Any other editing rules, constraints, assertions etc.
7. Derivation rules

**7.2. Data Warehouse Implementation Model**

The data warehouse implementation model is a physical realization of the Integrated Data Warehouse Collector Data Model. Entities are mapped to corresponding tables, attributes are mapped to corresponding columns, and relationships, business rules are mapped to appropriate foreign key, check, and default table constraints.

The data warehouse implementation may consist of additional tables, pre-formed queries or procedures and, additional constructs to support the data warehouse architecture, especially to support data cleansing and data scrubbing activities in a tight data quality controlled environment.

The data warehouse implementation models will partially be the products of automatic generation of schemas using extract, load and transform facilities. Additional constructs may need to be added to the generated models. These may include additional access path indexes, temporarily disabling constraints.

See [Annex B](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#annexb) for mappings and rules concerning formalisms.

**7.3. Data Warehouse End User Business Output Product Data Models**

The output products of the data warehouse implementation can be categorized into several forms of end user business data models.

Depending on the nature and targeted audience of the end user requirements, the output products could consist of the following combinations:

1. Multiple Data marts
2. Multi-dimensional Cubes
3. Reports
4. Queries
5. Views
6. Data Warehouse Subset implementations using data mirroring
7. Star Schema
8. Snowflake schema
9. Fact Tables, etc.

One or more data models in the Model Repository shall support each type of output product, with suitable mappings to the Data Warehouse Implementation model.

End user business definitions on columns of interest shall be available as extracted from the Repository. Be it a cube, a star schema configuration, a snowflake configuration, a report or view, each column in the output product needs to have a definition that is the same as the source input definition.

**7.4. Data Warehouse – Transformation Mappings and Rules**

To enable the data warehouse and make it operational, it is necessary to keep track of mappings between each of the model components.

The Data and Process Modeling repository already has built in mechanisms for providing several of the cross mappings between repository components.

However, specific mappings need to be designed for cross model mapping via mapping schemas, which include the associated rules. These mapping schema tables need to be defined in a database and populated accordingly from the row instances of the inputted data models.

Mapping necessarily takes the following forms:

1. 1:1 (one to one)
2. 1:n (one to many)
3. n:1 (many to one)
4. m:n (many to many)
5. 0:1 (zero to one)
6. 1:0 (one to zero)

Where the left-hand side represents the source, the right hand side represents the target. In other words, data from the source system may realize itself in the target system via a transformation applying the above rules, plus, any accompanying filtering criteria on each.

The mapping schema must be insulated from a specific source-target combination. In other words, the same mapping schema shall be applicable to one or more source-target combinations.

**7.5. Data Warehousing Mapping Framework**

Considering there are several mappings involved amongst several components like source systems, staging areas Data Warehouse, and Data Marts, it is necessary to establish a mapping framework that needs to be adopted and implemented via the model repository, supplemented by any additional information schema tables.

Table Z951\_MAPPINGS in the extended information schema tables provides a facility to define the mappings for the mapping framework between a given source table to a given target table, providing related information on the data type, filtering criteria, transformation criteria etc.

**Figure 3** below defines a mapping framework that is supported by table Z951\_MAPPINGS.

Figure 3: Mapping Framework
Legend:
S - Source Systems
G - Staging Area
D - Data Warehouse
M - Data Marts



Mapping Schemas are shown as:

SG - Mn: Source to Staging Area Mapping for Source system <n>
GD - Mn: Staging Area to Data Warehouse Mapping for Staged system <n>
DM - Mn: Data Warehouse to Data Mart Mapping for Data Mart <n>
SD - M1-n: Source to Data Warehouse Mapping for Source Systems 1to <n>

The framework positions the various components involved in mappings in a given source-to-target environment.

For example:

Mapping schema **SG-M1** represents the mapping between **source system** to **staging area** for source system 1.

Mapping schema **GD-M1** represents the mapping between the **staging area** for source system 1 to **Data Warehouse** for source system 1.

Mapping schema **SD-M1** represents the mapping between **source system** to **Data Warehouse** for source system 1.

The Metadata Repository is logically portrayed as involving Data and Process Modeling Tools and extended information schema tables supplementing the Data and Process Modeling Tools metadata.

The Modeling Tools metadata in effect, represents the information schema of the Transport Canada Model and its associated components. The mapping schemas contain the mappings between the various components. Since Data and Process Modeling Tools do not have the capabilities to define the mapping schemas, these have been defined in additional supporting tables that will constitute Transport Canada's extended information schema. The physical realization of the metadata in the extended information schema tables will be discussed later.

**7.6. Data Warehouse Mapping Source System to Staging areas**

As mentioned earlier, it may not be feasible to simply capture the source systems data models in isolation. The opportunity to capture the source system data model components will arise when the candidate source system has been selected to provide source data for the data warehouse

Data models need to be defined for the source system (at least a minimal set), the staging area, along with associated mapping criteria between the two. In general, the staging area data model will be a full source system model or a subset of the source system mode, more typically the latter. The table structures of the staging area would normally be representative of the source table structures. The data sizes and data formats of the attributes/columns of the source system will be maintained accordingly.

This mapping between source to staging will be the driver for the Extract Transformation Loading facility to extract data from source for populating into the staging area.

Staging area tables must be defined in SQL format, with source system naming conventions, and be treated as Foreign Tables as per SQL99.

**7.7. Data Warehouse Mapping Staging areas to Data Warehouse**

The mapping between the staging area and the integrated data warehouse area will consist of CAST definitions (a conversion function in ISO Standard SQL to convert data types e.g. integer to character) for data type transformations, cleansing algorithms where applicable, or any filtering criteria (if the filtering was not done from source to staging area).

An integrated target data warehouse data model will have been defined. The mapping schema will contain appropriate entries for the mapping between the source staging area and the target integrated data warehouse area.

The mapping forms listed above will normally be applicable. For 1:1 mappings, only one set of filtering criteria need to be defined between the source area and staging area, or between staging area and data warehouse, but not in both.

**7.8. Data Warehouse Data Filtering, Data Transformation rules**

The mapping row entry in the mapping schema shall contain all CASTing rules i.e. transformation or conversion of data types in SQL99 format, any filtering or merging/splitting criteria in SQL99 syntax, to take into account the different mapping forms listed above.

Data filtering, data transformation rules are applicable at all stages of transfer.

Specific rules may be applicable to data from the source system to staging area tables.

**7.9. Data Mappings from Source to Transport Canada Data Warehouse (DW)**

The data mappings from source system models to the Data Warehouse Model also need to be maintained.

Note that this is in addition to the mappings between source system and staging area models, and staging area to Data Warehouse Models. There are different functions involved for each mapping. Essentially, these two mappings are used for Extract, Transformation and Loading (ETL) services. That is, to be able to manipulate and transform the data.

The mapping between source and Transport Canada models is for traceability and semantics. This is used for deriving the Transport Canada and Data Warehouse models, and for making decisions on inter-system feeds and operations.

It is important to maintain consistency and integrity amongst the 3 mappings listed above (ETL). The Data and Process Modeling metamodel does not at this time provide capabilities to handle these constructs.

Thus, there is a need to create a mapping schema, and to either add this as an extension to the Data and Process Modeling repository, or, to maintain it externally in a separate database environment.

It is recommended that the development of the external mapping schema be a 2-step process.

Step 1: would be to develop this as an external database in the extended Information Schema tables, ensuring the feeds are from Data and Process Model metadata table extracts i.e. the table and column names of the source data model, staging data model, and the TC (and DW) data model.

Step 2: (optional) would be to integrate the mapping schema into metadata tables as Modeling extensions. This should happen at a later date, after some form of stability has been achieved in the external process.

Considering that there are numerous changes being undertaken by Oracle, essentially with respect to adaptation of SQL99 features in Oracle DB, and, resulting in changes in Data and Process Models and associated facilities, it is not recommended that extensions be undertaken to the Data and Process Model metamodel.

**7.10. End User Requirements – Data Mappings from Data Warehouse to Data Marts**

Mappings between data warehouse and the data mart models also need to be maintained.

The following is the information that the end-users require:

1. Business descriptions of target (end-users) data elements
2. Description of transformations (from source to data marts)
3. Description of data load (errors, dates)
4. Business descriptions of preformatted reports

This set of metadata may be difficult to deliver in the current environments where intelligent interfaces amongst Data and Process Modeling Tools and Business Intelligence tools do not yet exist.

However, information on Items a) thru c) could be made available via the combination of Data and Process Modeling and extended information schema tables.

**7.11. Data Warehouse Cleansing and Scrubbing Rules**

Data cleansing essentially is concerned with ensuring the integrity and quality of data. These include rules for restructuring records or data elements, column value decoding and translation, rules for supplying missing data element values, and rules for data integrity and consistency checking. Data that violates constraints are denied entry into the database declared collector tables automatically by the DBMS. These rows are resident in the RPR tables. The Z903 table issued to produce reports to identify the issues for those rows that were denied entry. Caution should be used when disabling constraints. The incomplete data can pass through to the collector tables but the row is identified in the Z903 table. This row is also resident in the RPR for future tracking and audit control. The violated data can be displayed in a report for business analysts to review.

The automated approach to data cleansing requires each business rule (also cleansing rule) to be named and identified. It is also necessary to be able to identify each incoming row of data in a given table. The loading process shall be designed to undergo a rule conformance check of the incoming data row with each applicable cleansing rule.

Additional tables have been defined in the extended information schema to track the violators of the rules by row identifiers and rule identifiers.

Details are collected on a given row that violates one or more rules, the status assigned to the row for cleansing for a given run, e.g. ERROR, UNDER REVIEW, PENDING, OK IGNORE, FIXED etc. See table Z903\_CONSTRAINT\_VIOLATIONS.

Statistics are collected by table, rule violated, giving the number of rows violated in a given run.

These rules must be captured in the Metadata Repository.

**7.12. Data Warehouse Load Control Statistics**

This section primarily addresses data loading in a data warehouse environment. In the Extract-Transform-Load (ETL) process, it is necessary to maintain an audit trail of the number of occurrences loaded, number of occurrences rejected and, such statistics as start date-time of load run, end date-time of load run etc. Statistics are collected for every load run for any given table, qualified by a given database. The number of rows violated denotes the rows that were rejected due to violations of declared integrity constraints of Primary Key, Uniqueness constraints, Check constraints, Foreign Key constraints, and Assertion constraints (inter table). In addition, there are many supporting constraints that are recorded in the model table Z905. Details are collected on a given row that violates one or more rules, the status assigned to the row for cleansing for a given run, e.g. ERROR, UNDER REVIEW, PENDING, OK IGNORE, FIXED etc. details found in table Z903\_CONSTRAINT\_VIIOLATIONS. Additional tables have been added to provide user-friendly bilingual text support for error messages based on rule violations. (Z902\_RUN\_VIIOLATIONS). It could be conceivable that certain rules may not be applicable for certain runs, and thus these rules could be temporarily disabled by run. This can also be automated via table Z906\_DEFERRED\_CONSTRAINTS. This information is needed whether it is an initial load, refresh load, or periodic load. Information is also needed on the data cleansing runs and associated results.

**7.13. Data Warehouse Audit and Control Metamodel**

A load control metamodel has been defined to support the required statistics and audit control information. See Annex D.

The load control metamodel is defined via the extended information schema of entities/tables. This model is defined using Data and Process Modeling Tools. The schema generated from the models is used to build and maintain an external database. The extended information schema series of tables are to be populated for every load operation in the data warehouse.

**7.14. Identifying Data Warehouse Loads**

A run identifier identifies each load run. A run identifier is assigned for a given run, and may involve data from one or more databases, and one or more tables.

It is important that the run be associated with a run start date and run end date, along with the date interval of the data being loaded.

A run is defined as being open when there is no run end date. Many runs can be open at the same time. A run is closed by a run end date. A run once closed (by a run end date) cannot be reopened. Instead, a repair run (run type = special load, see below) can be initiated as a separate run. A given table cannot be involved in more than one open run at the same time; i.e. the run shall be closed before another run is initiated for the same table.

A run type is also to be identified, stating the kind of run i.e. daily load, weekly load, monthly load, quarterly load, half year load, annual load, fiscal year load, initial load, re-load, test load, special load etc.

In addition, there are many supporting constraints that are recorded in the model, see table Z901.

**7.15. Data Warehousing Collecting Load Statistics**

Statistics are collected for every load run for any given table, qualified by a given database.

A row count is to be recorded before the run for the table being loaded, and a row count after the run is to be recorded for the table being loaded.

Also recorded are number of rows input, number of rows loaded, number of rows violated on the load. The number of rows violated denotes the rows that were rejected due to violations of declared integrity constraints of Primary Key, Uniqueness constraints, Check constraints, Foreign Key constraints, and Assertion constraints (inter table).

The start datetime and end datetime of the load for the given table are recorded in the given run.

In case of an abnormal termination of the load, the datetime is recorded, along with number of rows at abend.

In addition, there are many supporting constraints that are recorded in the model, see table Z905.

**7.16. Data Warehousing Automating and Tracking Data Cleansing/Data Quality Issues**

Some proponents in data warehousing state that the data cleansing activity could very easily consume up to 80% of the loading schedule times.

It will be more productive to track the data going through the data cleansing process, particularly, when many precise steps have been defined.

Data cleansing issues are often relegated to being a labour intensive after-load process, or, in some cases treated as part of the loading process in an arbitrary fashion. This approach does not leave room for addressing the data in error and the follow-on actions. Data cleansing algorithms are usually applied manually through discrete ad-hoc processes and procedures.

Data quality and data consistency has to be dealt with on multiple levels. It is simply not sufficient to deal with this issue during a load stream, on a stand-alone basis. Data is normally being integrated in a data warehouse from multiple sources. Data consistency is to be sought immediately after the integration has occurred. It is futile to spend resources on an individual loading stream and then to hope that the data is going to be consistent when integrated. For example, data could be consistent in each source system within its individual boundaries. But, when integrated, this very same data could become inconsistent when brought together.

This issue could be handled manually, after integration to review consistency, and to follow up via hand-written application procedures to further cleanse the data. Issues that still need to be addressed are "what to do with culprit data?", "how does one follow up with feedback to source system and awaiting clarifications or update to cleanse already loaded data?"

An intelligent data cleansing architecture uses the facilities and features of the modern day Database Management System based on established International SQL standards. All data have to adhere to some form of rules based on formal propositions regardless of their residency, be it in source systems, in the data warehouse or in data marts.

When there are 2 or more source applications, the data warehouse architecture shall define a fully integrated normalized data structure, strongly supported by structurally enforced business rules and constraints using formal syntax and semantics of ISO SQL, for the collector database where source system data are being integrated.

Facilities must also be made in the architecture to house the culprit occurrences that need to be further cleansed, and to define minimal "integrity sets" to pass through the data that is consistent as a set but not otherwise.

**7.17. Data Warehousing Additional User Friendly Support Tables**

Additional tables have been added to provide user-friendly bilingual text support for error messages based on rule violations. See table Z902\_RUN\_VIIOLATIONS.

**7.18. Data Warehousing Individual Rules Disabled for Selected Runs**

It could be conceivable that certain rules may not be applicable for certain runs, and thus these rules could be temporarily disabled by run. This can also be automated via table Z906\_DEFERRED\_CONSTRAINTS**.**

**7.19. Data Marts**

Data Marts are essentially de-normalized extractions of data from the data warehouse database, that represent subsets of data over some period of time, or range of object occurrences, and restricted to accommodate selected dimensions.

Data Models supporting the de-normalized (sometimes pre-joined for 1:1) structures, in the form of Star Schemas, Snowflake Schemas etc., are to be defined in the Metadata Repository.

The Integrated Data Warehouse Collector Data Model is an optimally normalized integrated data model, representing a selected implementable subset of the combination of entities/tables, attributes/columns, relationships that model the implementable subset of the Transport Canada data model.

It is important that the data warehouse data model is optimally normalized to enable the implementation of automated facilities for data cleansing and scrubbing. Normalization to the highest degree or at least the 3rd normal form, is mandatory to ensure the integration of the various source system data semantics.

The modeling formalisms and standards are depicted in [Annex B](http://www.tc.gc.ca/eng/corporate-services/imit-metadata-standards-634.htm#annexb).

**Annex A – TC Metadata Model**

This Annex defines the TC Metadata Model used by Data Modeling Tools. The metamodel incorporates entities, attributes and relationships that constitute the TC metamodel. The entities represent the extended functionality that is required for Transport Canada.

The table below lists the tables used in the Transport Canada metadata model.

| **Table Name**  | **Description**  |
| --- | --- |
| APPLICATION PLATFORM  | The platform on which the application or information holding is designed on i.e. 3rd generation languages, web based, SQL Server, ORACLE etc.  |
| ASSERTION\_TABLE\_USAGE  | Has one row for each table identified in a qualified <column reference> contained in the <search condition> of an assertion.  |
| ATTRIBUTES  | Defines a list of data items of an entity  |
| ATTRIBUTE\_COLUMN\_ MAPPING  | Mapping of entity attributes to table columns  |
| BRANCH  | List of TC branches  |
| BUSINESS\_RULES  | Has one row for each business rule or norm  |
| BUSINESS\_RULE\_ CONSTRAINT\_MAPPING  | Mapping of a business rule to an SQL constraint  |
| CHECK\_COLUMN\_USAGE  | Has one row for each column identified by a <column reference> contained in the <search condition> of a check constraint, domain constraint or assertion.  |
| CHECK\_ISA\_ASSERTION  | A subtype of Constraint\_is a\_check, has one row for each assertion constraint  |
| CHECK\_ISA\_TABLE\_CHECK  | A subtype of Constraint\_is a\_check, has one row for each table check constraint  |
| COLUMNS  | Has one row for each column in a table  |
| CONSTRAINTS  | Has one row for each constraint as defined in SQL  |
| CONSTRAINT\_ISA\_CHECK  | A subtype of Constraints, has one row for each check constraint  |
| CONSTRAINT\_ISA\_KEY  | A subtype of Constraints, has one row for each key constraint, primary key, unique constraint, and foreign key constraint  |
| CONSTRAINT\_ISA\_REFERENTIAL  | A subtype of Constraints, has one row for each referential constraint  |
| CONSTRAINT\_ISA\_TABLE\_ CONSTRAINT  | A subtype of Constraints, has one row for each table constraint associated a table  |
| DATA\_STORE  | A place where persistent data can exist  |
| DATA\_FLOW  | Flow of information that carries meaningful data  |
| DATA\_FLOW\_ATTRIBUTES  | The main data items contained in the dataflow  |
| DATA\_STORE\_ATTRIBUTES  | The main data items contained in the data store  |
| DBMS\_PLATFORM  | The database platform used to develop and maintain the application or information holding  |
| DEPARTMENT  | An organizational unit, of the government, commercial or other  |
| DIRECTORATE  | List of TC directorates for the purposes of this study  |
| DIVISION  | List of TC divisions for the purposes of this study  |
| ENTITY  | Has one row for each entity  |
| ENTITY\_TABLE\_MAPPING  | Mapping of entities to tables  |
| EVENT  | A happening or an instance that causes action, or is a result of an activity  |
| EVENT\_ISA\_TRIGGER  | A subtype of an event, as a instance that causes an action  |
| EVENT\_ISA\_OUTCOME  | A subtype of an event that is a result of an activity  |
| EXTERNAL\_AGENT  | A user or system interfacing with a function  |
| FUNCTION  | A business activity  |
| FUNCTION\_ATTRIBUTES  | The main data items created, updated, retrieved or deleted in the function  |
| FUNCTION\_ISA\_PROCESS  | A subtype of function that is a lowest level function  |
| FUNCTION\_PRODUCT\_ USAGE  | The products used in the conduct of a function or business activity  |
| FUNCTIONAL\_SUBJECT\_AREA  | The functions of Transport Canada as defined in a hierarchical list  |
| INDEXES  | Has one row for each database index  |
| INDEX\_COLUMNS  | Has one or more rows for each column contained in an index  |
| INFO\_HOLDING\_ ATTRIBUTES  | The main data items involved in or relevant to an information holding  |
| INFO\_HOLDING\_CATEGORY  | A grouping or a class of information holdings based on main data items used  |
| INFO\_HOLDING\_IN\_ CATEGORY  | A grouping of information holdings by information holding categories  |
| INFO\_HOLDING\_ INTERFACES  | The interfaces to other information holdings, applications, information systems from an information holding  |
| INFO\_HOLDING\_INTERFACE\_ ATTRIBUTES  | The attributes involved in the interface to other information holdings, applications, information systems from an information holding  |
| INFO\_HOLDING\_TYPE  | Denoting the structure of information holding, whether organized or structured in the form of a database or file, or unstructured in the form of text, digital, analog, image or other.  |
| INFO\_HOLDING\_ISA\_ STRUCTURED  | Additional information on structured holdings as to format and storage  |
| INFO\_HOLDING\_ISA\_ UNSTRUCTURED  | Additional information on unstructured holdings as to format and storage  |
| INFO\_HOLDINGS  | The main collection of information holdings, including all relevant details  |
| KEY\_COLUMN\_USAGE  | Has one or more rows for each row in the Table\_constraints table that has a constraint\_type of "unique", "primary key", or "foreign key". The rows list the columns that constitute each unique constraint, and the referencing columns in each foreign key constraint.  |
| KEY\_ISA\_UNIQUE  | A subtype of Constraint\_isa\_key, has one row for each unique constraint  |
| MAIN\_DATA\_ITEM  | A broad brush collective level of data denoting some form of grouping of attributes  |
| OS\_PLATFORM  | Details of the operating system platform involved with the information holding  |
| PROCESS\_SEQUENCE  | Defines an ordered collection of processes to be performed in a given progression  |
| PRODUCT  | An information item intended for use or consumption  |
| PRODUCT\_INFO\_HOLDINGS\_ATTRIBUTES  | The main data items involved with the product  |
| RELATIONSHIPS  | An association between entities involving two way roles  |
| UNSTRUCTURED\_TYPES  | Classifications of unstructured information holdings  |

**Annex B – Data Model Formalisms**

This appendix of standards applies to the Physical Data Model (PDM). The Logical Data Model (LDM) is not as tightly coupled to these standards definition. Please refer to the Quick Reference Guide (RDIMS – 1079406) for more detailed examples. The models will consist of both a logical and physical model. The examples in this Annex are displayed using the physical model unless otherwise indicated.

Annex B includes the template for the input of Data Models Entity Relationship Diagram entries. These entries are the descriptive properties for the metadata objects comprising of tables and columns, including primary key constraints, unique constraints, check constraints, assertions, procedural constraints, default values, and derivation rules.

The purpose of this section is to enable standardization of descriptive properties in Data Modeling, so that they are consistently represented.

NOTE:

The logical model does not require the table name prefix or underscores between words. Only attribute and verb phrase relationship displays are required on the logical model. This view can be used for model reviews with business clients.

Physical models will convert the spaces contained in the logical to an underscore. The physical model requires each table to start with a prefix from the category and subcategory tables followed by a three-digit number. Synonyms can be entered in the model for each table.

This document defines the formal grammar or selected options for use on data model constructs, to accompany Transport Canada Data Modeling Standards (Annex B). Formal grammar is defined using BNF (Backus Naur Form), as used in the ISO IS9075: 1992 Database Language SQL standard, or TBITS 2 SQL.

Throughout the document and this Annex it is strongly recommended to have constraints coded at the table level. However the use of

Erwin’s data model null option is permissible (“Not null” or “Null”) at the column level, as well, “valid value” constraints are also permissible at the column level.

The naming convention for associative tables (XREF) should use the table prefixes of the tables involved. However, it permissible to use the new table name prefix followed by XREF and the actual table names as long as the tables names do not have to be severely truncated to fit within the 30 character name length convention.

Foreign key attribute names may carry the table prefix in large models where it will facilitate and clarify navigation within the physical model.

Management of Government Information dates need only appear at the physical model if the modeler decides to implement them. It is up to the individual application/modelers/developers/application owners to implement the MGI principles as required within their application by use of these design dates and/or appropriate application design principles. The definition of the date attributes will be contained in the template model."

| **Serial #**  | **Data Model Construct**  | **OptionsBold *indicates selected option***  | **Example**  | **Explanation**  |
| --- | --- | --- | --- | --- |
| 1.  | **Identifier Names (Entity/ Table, Attribute/Column,Constraint Names etc…**  | AbbreviationsData size of identifier  | 30 character restrictionRemove vowels and repeated alpha characters except at the start partial from the right or partial main concept  | Abbreviations are only to be used when necessary. When used, the original unabbreviated name is also to be recorded as documentation.The 30-character restriction can have an impact on identifier names being used in whatever procedures/programs necessary or triggers, for enforcing constraints, in particular, CHECK clauses.  |
| 2.  | **Definitions of tables, or columns**  | Definitions cannot be free form without any structure. **Follow the ISO based definition formalism.**  | Example of free form without structure:Vendor status: Status of vendor Example of standard and structure: Vendor Status: A rating that indicates the reliability of the vendor in term of meeting quality and delivery dates. e.g. Gold = excellent, Silver = good Follow ISO Standard IS:11179-4 based definition formalism: * not be circular
* state what it is, not what it is not
* attempt to identify a group/family it belongs in e.g. light bulb is an electrical appliance…
* don’t include constraints in the definition
* include examples as an add on sentence if available or as a separate construct (need to standardize this)
 |    |
| 3.  | **Table names**  | **Fill with Underscore**Do not blank fill  | **XXX\_YYY**  | The use of underscores at the physical, improves readability and usability. In particular, it avoids misinterpretations when the name is used as part of a sentence when the repository objects are referred to in written communications**Note: Use underscore in naming of tables when creating them in the Data and Process Models as part of the design. In this way, one is able differentiate designed tables vs. reverse engineered tables.**  |
| 4.  | **Table naming**  | **Associate abbreviated prefix along with identifier text**  | YC022\_AIRCRAFTIA025\_INSPECTIONSMA101\_AIRCRAFT\_ ACTIVITY **Format:****< holding category>** *see Subject Area Holdings Category in section 2.5* **< holding sub category>** *see Subject Area Holdings Sub- Category in section 2.5* **<sequence number>** *001,and increment by 1 for each new table* **<underscore>***\_* **<meaningful table name>** *semantic table identifier*  | A table prefix is made up of a subject category and sub-category and a table sequence number and are assigned to each table name. This allows for easy identification and ownership of tables and, in particular, enables naming table aliases in SQL triggers, procedures or orASSERTION/CHECKe.g. with number:CHECK (BA022\_AIRCRAFT.aircraft\_id NOT IN (SELECT IA025. Aircraft\_id FROM INSPECTIONS AS IA025 ….) The Logical Data Model is not required to have prefixes or underscore characters attached to the logical entities. For the Physical Data Model, underscore character shall be used and the table numbers will be used based on subject category and sub-category for ease of reference.  |
| 5.  | **Table Reference tables**  | **Specially identified as a** **Table Reference**  | AIRCRAFT\_TYPE**TA044\_AIRCRAFT\_TYPE****Format:****<Table prefix> <Holding Category><Holding sub category><sequence number> <underscore> <meaningful table name>**  | Easy to identify this is a reference table - (T) this as an AIR (A) sub-category, 044 sequential number followed by meaningful table name. The sub-category (C ) is reserved for Departmental Core tables.  |
| 6.  | **Entity Role names in a relationship**  | Role Name Syntax  | **Role name must fit into construct using the format below** Format: **<entity name> <optionality> <rolename> <cardinality> <entity name>** **<optionality> must | may [have|be]** **<cardinality> zero|one|many**  | In the Data Modeling Tools this is the same as a relationship name. What is important is that the actual relationship verb-phrase makes sense with the standard constructs. For example, Parent to/Child rules, begin the relationship sentence in reports with ‘An/A <entity name>’, prefixes the relationship name with MAY, and suffixes with ‘many’zero, one or more’ <entity name>. So, a relationship name of “the\_resident\_of” would be more readable than “resides\_at” in the following example. It is important to use the Role Name when there are multiple foreign keys between the same two tables. A C\_CUSTOMER MAY the\_resident\_of many zero, one or many C\_CUSTOMER\_ADDRESS A C\_CUSTOMER MAY reside at zero, one or many C\_CUSTOMER\_ADDRESS  |
| 7.  | **Subtype naming**Note: Subtypes directly corresponds to either its own table (subtable) or can be collapsed into a super table --- ensuring that all the relevant integrity constraints are carried over.  | a) Subtype naming cannot be arbitrary b)**Definitive, based on supertype name irrespective of level, with <ISA> syntax**  | Arbitrary example:CARRIER supertype,AIRCRAFT subtype Definitive example:CARRIER supertype, CARRIER\_CARRIER\_ISA\_ AIRCRAFT subtypeFor the 1st level of subtypes, use the “IS\_A” construct is used by the subtype symbol. For subsequent levels use double underscore (\_\_) as connectors. This symbol causes the creation of the “IS\_A” construct in the Child Relationship Parent to Child Rule. The Child Relationship Child to Parent Rule becomes “is a type of” construct.Eg.CARRIER\_AIRCRAFT\_JET CARRIER\_ISA\_AIRCRAFT JET, where CARRIER is the supertype, AIRCRAFT is a subtype of CARRIER, and JET is a further subtype of AIRCRAFT Format:**<subtype name>****<subtype entity name>\_ISA\_ <direct supertype><direct supertype entity name>|<subtype name><underscore><underscore><meaningful\_entitysubtype\_name>**  | By making it arbitrary, it is not known who the supertype is---lost in the hundreds of tables that will be produced. Know exactly who the supertype is. Note: (1) subtyping is strongly recommended, as long as a subtype matrix is drawn up meeting all required mandatory criteria i.e. no dangling subtypes (must have a relationship or meaningful attribute) (2) Sorting criteria determines the placement of the table or entity names in a list. The recommended option provides for easy identification of a set of entities/tables since they will be grouped together.  |
| 8.  | **Subtype indicator**  | **Required at each subtype level, exclusive** or total  | For navigation purposes, need to know what subtype to go to, including all the possible subtype populations, e.g. Party, vendor. A vendor can be an organization or a person, with each of the entities organization, vendor, person at the same level belonging to direct supertype Party. **a) Declare a subtype indicator as an attribute of the direct supertype entity if the subtypes are mutually exclusive****b) Declare a subtype indicator as a 2 column referenced table for subtypes with a non-exclusive role i.e. where the subtypes are inclusive, with Primary Key of (supertype column value + subtype role name)** |    |
| 9.  | **Column naming**  | **Prefix with a group class****Ending suffix with a class term**  | **1) User defined Columns:** Format:**[<context grouping identifier>] <underscore> <column name> <meaningful column name><underscore><class term>** *e.g.*TOTAL\_ …AMOUNT\_ …DATE\_ …PERCENT\_ …COUNT\_ …etc. Note: (1)Group like concepts togethere.g. begin withTOTAL\_FATAL\_NBR **<column name>****<meaningful column name><underscore><class term>** **<class term>ID|CD|NM|ENM|FNM|QTY|DTE|TXT|ETXT|FTXT|AMT|NBR| NUM|IND|RTE** **e.g.****VENDOR\_ID****DELIVER\_TO\_NM**,Where \_ID and \_NM are class terms. **Each column must be suffixed with a class term to assist in identifying the kind of column.** **Format for** **Class Terms**: **ID** - identifier, an arbitrarily assigned set of alpha and/or numeric characters used to identify objects based on some predetermined criteria.E.g. INSPECTION\_ID 103456 represents the unique identifier for a record of an inspection that took place. **CD** – code, a pair consisting of code and description, each possibly being unique. A code shall be used when the attribute column is expected to either contain a finite set of values or is based on a defined algorithm. This is generally implemented via a reference table (T) where the values are defined along with a referential integrity constraint to the table having the code values as attributes of the column. Alternately, for a small code population, the check clause can be substituted for constraining the values of the code. **NM** – name, used as an alternate identification which may or may not be unique (default, english) **ENM** - english name, as an alternate identification which may or may not be unique, used when \_fnm is present **FNM** - english name, as an alternate identification which may or may not be unique, used when \_enm is present **QTY** – referring to quantity, on which numeric operations can be performed **DTE** – date **TXT** – descriptive text (default, english) **ETXT** – descriptive text (English, used when \_ftxt is present) **FTXT** – descriptive text (French, used when \_ftxt is present) **AMT**- amount of money **NUM** – number,used to identify any series of numeric digits **NBR** - number, used for counts, totals, or any decimal digit not in any other class term **IND** - Indicator, generally a Yes/No indicator representing a boolean value **RTE** - Rate, referring to speed, fee or proportion e.g. consumption etc. **BLOB** - used for defining images, video, graphics **GIS** - refers to geospatial data such as vector or rastor data **TIME** - used when recording only the time  | The group class can be used for sorting columns within a table when used it can eliminate redundancy in a column name. The use of group class is only a suggestion and not a requirement. eg: START\_DATE\_DTE optionally using the group identifier DATE would be: DATE\_START\_DTE If the example was: VENDORDELIVER\_TO Instead of the case where class term is used, it is not clear whether the VENDOR is a vendor name or an identifier. Same with DELIVER\_TO, not clear whether this is a name, text, or code **Use of Class Terms are mandatory.**  |
| 10.  | **Constraint naming** - Primary Key  | Transport Canada Data Modeling Tool (Template) generates a default name that is acceptable. If it is not used it**must be user defined manually**  | **Format:****<table\_prefix>)<underscore> <primary key designator>**  | Truncation may occur due to limit of 30 characters Note: Data and Process Modeling append a prefix to the <PK>e.g. Entering PK in the Data and Process Modeling Tools for the primary key will result in the primary key identified as: TC014\_PK  |
| 11.  | **Constraint naming** - Unique Constraint  | Transport Canada Data Modeling Tool (Template) generates a default name that is acceptable. If it is not used it.**must be user defined manually.**  | **Format:****<table prefix><underscore>****<AK> <sequence number>**  | Note: Data and Process Modeling appends the table prefix with the <AK>e.g.TC014\_AK1 It is important to use the sequence number which starts at 1 and is incremental by 1 for identifying additional unique constraints (when there are more than 1)  |
| 12.  | **Constraint naming format** - Foreign Key (including Foreign Key resulting from subtype)  | Transport Canada Data **Modeling Tool (Template) generates a default name that is acceptable. If it is not used it** **must be user defined using table prefix and relationship identifier (and role name)**  | **Format:****<from table prefix name> <underscore> <to table prefix name> <underscore> \_FK**  | A foreign key is created by using the child table prefix followed by the parent table prefix followed by the foreign key identifier. e.g.YA022\_TA044\_FKWhere YA022 is the prefix for the referencing table YA022\_AIRCRAFT and TA044 is the referenced table TA044\_AIRCRAFT\_TYPE Note1: use of the table prefix names is mandatory to achieve this result.Note 2: the same format applies to FK relationships resulting from a supertype-subtype relationship, which are essentially a 1:1 relationship.  |
| 13.  | **Subtype PK Column naming –** In a Foreign Key resulting from subtype  | Modeling Tool (Template) Generates a default name that is acceptable. Subtypes when applied to columns inherits same name as the supertype column Subtype column has a different role and hence a new column name  | **Format:** **<subtype PK column name><to table prefix name> <underscore> <local column name>**  | Often it may be necessary to define a new role name for a local column for a subtype PK. e.g.Supertype PK has EMPLOYEE\_ID.Functional requirements may dictate the assignment of a MANAGER\_ID as a subtype attribute column PK, which has subsequent FK relationships to additional child tables. In this case the subtype PK column may be named after a role e.g. MANAGER but the <to table prefix> needs to be added as a prefix to the local column to denote the source of the FK.e.g.Supertable: AC002\_EMPLOYEE with PK EMPLOYEE\_IDSubtable : C022\_EMPLOYEE\_ISA\_MANAGER with inherited PK A02\_MANAGER\_ID  |
| 14.  | **Constraint naming** CHECK  | Modeling Tool (Template) Generates a default name that is acceptable. **Table name prefix, along with constraint number within table**  | e.g. CONSTRAINT YC022\_CK03\_ STOP\_GE\_START\_DTE CHECK (DATE\_STOP\_DTE > = DATE\_START\_DTE) tells us that this constraint belongs to YC022 and is the 3rd check constraint. Particularly useful when there are many check constraints in a table. Format:**<check constraint>****<table prefix name> <underscore> <CK> <constraint number within table> <underscore> <meaningful constraint identifier>**  | It is important to use the sequence number which starts at 01 and is incremental by 1 for identifying additional unique constraints (when there are more than 1). Note: meaningful constraint name may contain logical comparisons between other columns or value sets. This is denoted by the following acronyms. GT::= *Greater than*GE::= *Greater than or equal to*LT::= *Less than*LE::= *Less than or equal to*EQ::= *Equal to*NE::= *Not equal to*NN::=*Not Null* E.g.A001\_CK04\_END\_GE\_STARTCHECK (DATE\_END\_DTE >= DATE\_START\_DTE)  |
| 15.  | **Constraints Placement**  | Column level constraints vs. **table level constraints**  | Constraints are preferred to be declared at the table level. Each table constraint must be declared after the last column has been defined, beginning with the primary key constraint definition. e.g. in CREATE TABLE …DATE\_START\_DTE DATETIME NOT NULL **Constraints are preferred to be declared at table level** e.g. CONSTRAINT YA110\_CK04\_NN\_START\_DTE CHECK (DATE\_START\_DTE IS NOT NULL)  | In the case of column level constraints, the constraint name is system assigned which is difficult to trace ownership back in terms of violations. Same constraint is named by modeller at table level and is traceable via the dictionary tables or in INFORMATION\_SCHEMA.TABLE\_ CONSTRAINTS. Note:NOT NULL constraints can be defined using the tool’s “Not Null” check box in either the logical or physical model, however it is not recommended. These system generated table level constraints have unstable and inconsistent system names and can complicate error trapping and analysis.  |
| 16.  | **Use of permissible value sets or Value Constraints**  | Value sets  | **Preferred to be declared at the table level or via a reference table.**Note: Do not use the attribute/column level definition in a CHECK clause for large value sets (e.g. CHECK (AIRCRAFT\_TYPE IN ('B747', 'B747-400', 'B767'….25 more values follow.)  | Table level checking of permissible value sets must be provided so as to provide minimal impact on changes in values.**Notes:**Reference tables are preferred to be used even though examples of column value sets can be defined during the model definition. The Data Modeling tool creates by default a column level “valid values” CHECK constraint. This column level constraint is acceptable however it can increase the difficulty in trapping , identifying and resolving errors identified by these types of constraints.  |
| 17.  | **Assertions and Trigger usage**  | Conversion of Assertion Syntax to Trigger syntax  | a) Assertion naming: Format:**<assertion constraint> < table prefix name> [<underscore> <table prefix name> …] <underscore> <assertion constraint acronym> <underscore> <sequence number> <meaningful constraint name>)** **<assertion constraint acronym> AS** **<sequence number>** *1, increment 1* b) Trigger naming:**<trigger name>****<table prefix name> <underscore> <TG> <sequence number of trigger within table> <underscore> <trigger firing condition> <meaningful trigger name based on operation>** **<trigger firing condition> <BI> | <AI> | <BU> | <AU> | <BD> | <AD> | <BIU> | <AIU> | <IO>** **<BI> =** *Before Insert* **<AI> =** *After Insert* **<BU> =** *Before Update* **<AU> =** *After Update* **<BD> =** *Before Delete* **<AD> =** *After Delete* **<BIU> =** *Before Insert or Update* **<AIU> =** *After Insert or Update* **<IO> =** *Instead of*  | Assertions as defined in SQL92, are essentially CHECK clauses that can be enforceable between tables. However, Oracle’s current implementation does not have any capability to enforce Assertions. Hence these have to be converted to Triggers. Assertions can be defined at the semantic level of the data model. These will have to be flagged e.g. by naming as e.g. (<database category>\_<table number>\_< constraint type><constraint number within table>\_<constraint name text>), where constraint type could be CK for check, AS for assertion etc. This will make it easier for trace and conversion to triggers possibly, automatically. e.g. YA001\_TG1\_BI\_AIR\_REGISTRY\_ID  |
| 18.  | **Unique Identifiers and hidden columns in relationships**  | Do not leave everything to default **User defined UID naming**  | When you enter the UID general tab on an entity attribute/column in Data Modeling Tool, select the primary key check box.s: **User assigned based on entity/table naming** **e.g. <table prefix name> PK** | Applied to the INDEX option of the table in the physical view of the model.  |
| 19.  | **Complex procedural constraints**  | **SQL92 syntax**  | **Procedural constraints are to be declared as Assertion Constraints, which can later be converted to triggers, until this feature is available in Oracle.**  |    |
| 20.  | **Common tables (i.e. overlapping into subject areas) into a common Core**  | There are many tables that would be “involved” with many subject areas. These need to be re-defined into a “core table”, with modify privileges that need to be agreed upon regarding the data in them. e.g.. any column of a table that is involved with more than one subject area, is a candidate to be in the core group list of tables.  |
| 21.  | **Subject Matter based Integration vs. Technical model integration**  | There are two kinds of integration. Technical integration and subject matter integration. Technical integration is based on standards that can be established. There is the subject matter integration, that is based on semantics of the subject areas involved, which requires that two or more modeling teams be harmonized in terms of table and column constructs, constraints etc. Both Technical and Semantic Integration needs to be recognized and procedures established.  |
| 22.  | **Data Modeling Approach**  | The modeling approach taken should not be arbitrary. There shall be **supporting natural language sentences with examples, to justify any of the relationships or, the attributes of the entities---which essentially form the basis of any walkthrough with the users**. Projects based on an ad hoc approach generally meet with disaster. Supporting documentation on successful formalized approaches needs to be provided to assist in formulating a TC tailored approach (Note that no two projects are unique, and the plug-and-play of project methodology leaves a lot to be desired)  |
| 23.  | **Standard Data Types**  | Need to be defined for specific purposes e.g. amounts with decimal (19,4) varchar usage etc.  |    | **TC conforms to SQL92 standard data types and does not take into account any extensions (e.g. Money)**  |
| 24.  | **User Defined Data Types (UDT)**  | To be administered via Data Administration  | To avoid a proliferation of User data types, use the UDTs defined in ISO SQL/MM standard and in the emerging SQL2002 standard  | To be administered via Data Administration.  |
| 25.  | **Indexes**  | Custom Tables  | Used to apply naming standards to the key constraints.  | ORACLE generates the PK and AK indexes by default so there is no need to request this action. The FK index not automatically generated by the ORACLE RDBMS.  |
| 26.  | **Views**  | Identify views  | **<view> <V> <4 alpha><underscore> <meaningful view name>) or <V> <underscore><3 alph> <underscore><meaningful view name>** e.g. Vaaaa\_DIRECTORY\_CLIENT or V\_aaa\_DIRECTORY\_CLIENT  |    |
| 27.  | **Sequences**  | Identify sequences  | **<sequence> < table short name> <underscore> < SQ><sequence number> <meaningful sequence name>** **Usually the same as the trigger which uses this sequence)** e.g. CC001\_SQ1\_FO\_REPORT\_ID  |    |
| 28.  | **Synonyms**  | Using Synonyms  | These are defined as the same as the full table name.  |    |
| 29.  | **Procedures/Functions**  | Packaged Procedures & Standalone Procedures  | **<Procedure/function> <acronym><sequence number><underscore> <meaningful name>)**where**acronym SP or FN** e.g.SP\_ CHANGE\_PASSWORDFN\_ADD\_USER\_OBJECTNote: SQL databases will use PR for Procedures instead of SP.  |    |
| 30.  | **Packages**  | Package Specification  | **<Package> <PK><underscore> <meaningful name>** e.g.**PK RAIL\_REPORTS** |    |

**Annex C – Data Warehouse Audit and Control Tables**

Table below lists the tables used in the Transport Canada Data Warehouse Audit and Control Metamodel.

| **Table**  | **Description**  |
| --- | --- |
| Z901\_RUN  | Has one row for each run descriptor  |
| Z902\_TABLE\_CONSTRAINTS  | Has one row for each table constraint associated with a table  |
| Z903\_CONSTRAINT\_VIOLATIONS  | Has one row for each violated occurrence of a data row for a given table constraint  |
| Z904\_RUN\_VIOLATIONS  | Has one row that summarizes the data rows violated for a given table constraint  |
| Z905\_RUN\_STATISTICS  | Has one row for each table in a given run providing statistics for that table in the run  |
| Z906\_RUN\_CONSTRAINTS\_DISABLED  | Has one row for each constraint disabled in any given run  |
| Z907\_STAGE\_RUN\_CONTROLS  | Has one row for each set of run controls to be defined for a given stage (Staging to Repair, Repair to Collector etc.)  |
| Z908\_TABLE\_RUN\_CONTROLS  | Has one row for each set of run controls to be defined for a given table  |
| Z919\_TABLES  | Contains one row for each defined table including views.  |
| Z921\_GROUP  | Contains one row for each group descriptor. Tables are grouped for loading order based on their update dependencies.  |
| Z923\_DATABASES  | Has one row for each database  |
| Z951\_MAPPINGS  | Has one row that defines each mapping between a source column and a target column  |
| TR30\_LOADING\_STAGES  | Reference table that contains a row for each loading stage  |
| TR48\_RUN\_TYPE  | Reference table that contains a row for each run type  |

**Glossary and Acronyms List**

**API** - Application Programming Interface

**CAST** - a conversion function in ISO Standard SQL to convert data types e.g. integer to character

**CORBA** - Common Object Request Broker Architecture

**CWM** - Common Warehouse Metamodel

**CWMI** - Common Warehouse Metadata Interchange

**ER** - Entity Relationship modeling technique

**ETL** - Extract-Transform-Load

**GOL** - Government On-Line

**INR** - Safety & Security Information Needs Review

**IRDS** - Information Resource Dictionary System

**ISO** - International Standards Organization

**MDC** - Meta Data Coalition

**MDIS** - Meta Data Interchange Specification

**MOF** - Meta Object Facility

**NIAM** - Nijssen's Information Analysis Method

**OIM** - Open Information Model (from MDC)

**OLAP** - On Line Analytical Processing

**OMG** - Object Management Group

**SQL** - Database Language Standard Query Language

**SQL92** - ISO Standard Database Language SQL IS: 9075:1992

**SQL99** - ISO Standard Database Language SQL IS: 9075:1999

**SQL/MM** ISO SQL Multi-Media

**TC** - Transport Canada

**TCIS** - Transport Canada (extended) Information Schema

**UML** - Unified Modeling Language

**XMI** - XML Metadata Interchange

**XML** - Extensible Mark-up Language, subset of Standard Generalized Mark-up Language (SGML)