Open Geospatial Consortium

Saved 2018-02-28 11:02

External identifier of this OGC® document: [http://www.opengis.net/doc/AS//FG/P1/FM/1.0](http://www.opengis.net/doc/AS/dggs/1.0)

Internal identifier of this OGC® document:  
[https://opengeospatial.org/as/17-087r8.html](https://opengeospatial.org/as/17-087r7.html)

Logical identifier of standard  
https://opengeospatial.org/as/FG/P1/FM/1.0

Internal reference number of this OGC® document: 17-087r8

Version: 1.0

Category: OGC® Abstract Specification

Editor: John R. Herring

Geographic information —  
Features and geometry –  
Part 1: Feature models

Information géographique —   
Entités géographiques et géométrie —   
Partie 1 : Modèles de entités géographiques

Copyright notice

Copyright © 2018- Open Geospatial Consortium   
to obtain additional rights of use, visit [https://www.opengeospatial.org/legal/](http://www.opengeospatial.org/legal/).

Warning

This document is not a Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC nor ISO Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

**Error! Reference source not found.**

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR 5655 FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third-party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of contents

Contents page

[1 Scope 7](#_Toc507581184)

[1.1 The first part: feature models 7](#_Toc507581185)

[1.2 Other parts of this standard 7](#_Toc507581186)

[1.3 Geometry as feature spatial extent 8](#_Toc507581187)

[1.4 Normative language 8](#_Toc507581188)

[2 Conformance 9](#_Toc507581189)

[2.1 Conformance classes: https://opengeospatial.org/as/FG/P1/FM/1.0/general/ 9](#_Toc507581190)

[2.2 Conformance targets 11](#_Toc507581191)

[2.3 Existing Implementations 12](#_Toc507581192)

[3 Normative references 12](#_Toc507581193)

[4 Definitions 13](#_Toc507581194)

[5 Symbols (and abbreviated terms) 20](#_Toc507581195)

[6 Feature models 20](#_Toc507581196)

[6.1 Introduction to feature taxonomy 20](#_Toc507581197)

[6.2 Requirements Class: Feature Taxonomy and Data: https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/ 24](#_Toc507581198)

[6.3 Requirements Class: Ontology: https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/ 30](#_Toc507581199)

[6.4 Requirements Class: Feature schema: https://opengeospatial.org/as/FG/P1/FM/1.0/schema/ 30](#_Toc507581200)

[Annex A Conformance Classes 31](#_Toc507581201)

[A.1 Test Suite for Taxonomy https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/ 31](#_Toc507581202)

[A.2 Test Suite for Ontology https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/ 35](#_Toc507581203)

[A.3 Test Suite for Schema https://opengeospatial.org/as/FG/P1/FM/1.0/schema/ 35](#_Toc507581204)

[Annex B Example implementation 37](#_Toc507581205)

[B.1 Object implementation of taxonomy-controlled data 37](#_Toc507581206)

[B.2 Relational implementation of taxonomy-controlled data 37](#_Toc507581207)

[Annex C Bibliography 39](#_Toc507581208)

List of Figures

Figures Page

[Figure 1 — Basic Feature Taxonomy Structure 27](#_Toc507581209)

[Figure B.1 – A Taxonomy-based Instantiation in Classical OOPL 37](#_Toc507581210)

List of Tables

Tables Page

[Table 1 – Example feature entities in a key: value format 23](#_Toc507581211)

[Table 2 – Example feature-type taxonomy entries in a key-value format 24](#_Toc507581212)

1. Abstract

The Implementation Standard "Geographic information — Features and geometry – Part 1: Feature models" describes how geographic information in datasets and databases using a "feature model" are structured, created, stored, queried and manipulated.

URLs in this document are identifiers of structural elements of a standard, such as requirements, tests, requirement classes, and conformances test suites.

1. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc

OGC document

cartography

controlled vocabulary

geospatial information

geography

taxonomy

big data

ontology

semantic web

schemata

SQL

geographic

geospatial

databases

information management

unstructured data

structured data

semi-structured data

key-value pairs

1. Preface

This document "Geographic information — Features and geometry – Part 1: Feature models" covers:

1. the feature model which models the items for which information is carried
2. Schema and ontological definitions of features and their properties.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

1. Submitting organizations

The following organizations, members of the OGC Simple Features SWG (standards working group) submitted this Document to the Open Geospatial Consortium (OGC):

1. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

| Name | Affiliation |
| --- | --- |
| John Herring (ed.) | Oracle USA |
| Paul Birkel | Geosemantic Resources, LLC |
| Hideki Hayashi | Hitachi, Ltd., Central Research Laboratory |
| David Valentine | University of California, San Diego Supercomputer Center |
| Clemens Portele | interactive instruments GmbH |
| Satish Sankaran | Esri |
| Keith Ryden | Esri |
| Carl Stephen Smyth | Open Site Plan |

Features and geometries – Part 1: Feature models

[https://opengeospatial.org/as/FG/P1/FM/1.0/](http://www.opengis.net/spec/FG/FM/1.0/) [[1]](#footnote-1)

# Scope

## The first part: feature models

This standard enumerates the requirements for defining and representing geographic features in geographic information systems, applications and transmittable data formats.

This "Part 1: Feature models" defines a new General Feature Model with various representations, and how it determines the basic form of geographic information. This is derived from and extends **ISO 19109,** which, in its current form, only addresses schema-based data storage defined in UML, but leaves possibilities open for other non-schematic design systems as extensions, such as dynamic object systems, as JavaScript and other formats using key-value pairs.

The strict schematic formulations such as might be used in a complied object language or in query languages requiring a predictable format such as SQL in relational databases or any Object-Oriented Databases which depend on a compiled set of object types, tend towards uniformity in the structure of any feature class. This schema approach matches formal abstractions and is useful in controlled applications that depend on the consistency of a strict relational or object-class structures. In this sense, strict schemata are an interoperability barrier between different applications which may share sematic structure but not implementation structures.

The more flexible feature data structures described in this document, will use semantics, taxonomies and ontologies, to interpret a variable structure that match a common taxonomy and constraints but may vary in their structures because two instances in the real world might not be fully describable by a single "attribute template". The approach follows the ideas of a controlled vocabulary that strictly defines terms such as, in this document, name feature types and properties but includes restriction in a class only if demanded by the semantics of reality. Such approaches have proven workable in medical, legal and cultural record keeping, see [14] and [16], and its use is not new to geographic information. Feature and attribute catalogs (which are controlled vocabularies) have been in use in digital mapping since the 1980’s, see [6].

Semantically structured data sets aim to represent reality, not the specific needs of a single application. This leads to application independent data stores that can support all applications because they support a data structure that is both flexible and extendable. This shifts the support of interoperability from creating transfer formats, to creating inclusive, flexible, and thereby interoperable data stores usable by a wide range of application.

## Other parts of this standard

"Part 2: Metrics for geometry" will describe the distinctions between "Euclidean geometry" in a plane such as might be appropriate for a large-scale map and "geodetic, ellipsoidal and non-Euclidean geometry" in general on a curved surface such that of the Earth. The most common implementation is ellipsoidal geometry using the datum surface. Spherical geometry is sufficient for a small area using a local best fit spherical approximation consistent with local curvature. Global use of a single sphere is problematic; for example, geodesics on a sphere close in a single arc (great circles) but on more general ellipsoids, only the meridians and the equator do so. The other issue is that the ellipsoidal coordinates used are based not on the central angle of the radial line to the position, but the tangential angles of the surface, which introduces other subtle and non-uniform errors in "spherical approximations".

"Part 3: Simple geometry" will describe the simplest mathematical geometry definitions, based on linear interpolation in the **CRS**. The first 3 parts are a replacement for **ISO 19125-1:2004, Geographic information – Simple feature access – Part 1: Common architecture**. For completeness, this part will also include definitions for geodesic and rhumb "line" interpolations which complete the concept of "line" from Euclidean geometry. Linear equations in the CRS match the Cartesian coordinate equations for lines. The concept of rhumb lines match the constant bearing concept in navigation, which are lines in maps using a Mercator projection. While linear curves depend on the coordinate applied to the geography, geodesic and rhumb lines are controlled by the geometry of the reference surface (usually an ellipsoid of the CRS). Geodetics match the concept of the shortest distance curve. What is a single curve type in flat Euclidian spaces requires 3 different curve types in non-Euclidian, non-flat, ellipsoidal geometries.

Further parts will describe other geometry and topology structures and a set of parallel standards will define WKT (well-known text) and JSON (JavaScript Object Notation) for both features and geometry. It should be noted that WKT and JSON are dependent on key-value encodings, which can use taxonomies to organize the keys, and in some cases, the values as in code lists.

## Geometry as feature spatial extent

A feature is a representation of a real-world object. For spatial applications, the most important property of a feature is location which will be dealt with in later parts of this standard on geometry, which will be derived from and extend **ISO 19107 Geographic information — Spatial schema** and will contain implementation guidance and suggestions. The corresponding SQL syntax will be defined in **ISO/IEC 13249-3: Information technology — SQL Multimedia and Application Packages - Part 3: Spatial**. Where possible, the SQL/MM approaches to geometry will be paralleled in this series of standards, except where an equivalent approach is more consistent with the common mathematical approached used in CAD/CAM applications, modified to fit non-Euclidean reference surfaces. The only example of this divergence currently-known is the use of projective space in the common mathematical derivations for Bspline curves. The current SQL/MM approach uses a slightly different approach to including weights for points, which would have to be translated to projective space for the common Bspline implementations.

## Normative language

All normative language (statements that directly affect the implementation of items compliant to the standard, called "standardization targets") is restricted to **requirements**, **recommendations** and **permissions**. The word "**shall**" implies that conformance requires the statement to apply in the conditions cited in that statement; these are referred to as "**requirements**" and are numbered with "**Req#**".

The word "**should**" implies that the use of best practice recommends the statement to apply in the conditions cited, and these are referred to as "**recommendations**" and are numbered with "**Rec#**".

The word "**may**" implies that conformance and practice allow the statement to apply in the conditions cited, and these are referred to as "**permissions**" and are numbered with "**Per#**". Permissions are not necessarily complete, in the sense that everything not specifically blocked by a requirement is permissible. Permissions are often used to prevent the over-interpretation of requirements which might block a valid implementation of the requirement as stated. This is akin to the open-world assumption, e.g. all approaches are permitted unless explicitly and unambiguously forbidden by a requirement.

Any similar words used the text body ("must", "might", "will", etc.) are not official normative statements and are not marked as such; they usually refer to a logical result of compliance to one or more normative statements, and often appear in the discussion of normative implications. No unmarked statement is normative although it may be truly implied by the marked normative statements. The statements (requirements, recommendations and permissions) marked and numbered are normative even if the "normative language" is inappropriately used.

Requirements ("**Req#**") are collected in requirements classes. Corresponding conformance classes, in Annex A, collate all requirements associated to each conformance class which outline or suggest tests that may be use in to prove conformance with the associated requirements class. Each requirement class and conformance class have a URL label in the namespace of this document. Conformance classes often require conformances to a previously defined class.

Recommendations ("**Rec#**") and Permissions ("**Per#**") can be anywhere in the document because they are logically not testable and are not addressed in a Conformance Class "testable items" in any sense. Any test of a requirement that blocks the usage of either any recommendation or permission is invalid. All normative statements, requirements, recommendations, permissions and tests are labelled statements a having a URL in the namespace of the clause, which in turn is in the namespace of this document defining the standard. Any test suite that references this document should use the URL’s to validate its procedures. All such names are listed immediately after each named item.

# Conformance

## Conformance classes: https://opengeospatial.org/as/FG/P1/FM/1.0/general/

A conformance class tests the ability of a system to conform to the mandatory requirements expressed in this document. Recommendations and Permissions are not tested. The Conformance Clauses are in Annex A where requirements are associated to required tests. There are three conformance classes in Annex A which will gather the Requirements of the corresponding requirement classes, clauses 0, 6.3, 6.4 for representing feature models types and suppling suggestion to how the tests may be formulated:

* A.1 Feature Taxonomy Conformance
* A.2 Feature Ontology Conformance
* A.3 Feature Schema Conformance

In a Feature Taxonomy or Controlled Vocabulary, two lists of "well-known" terms are created to act as named keys and searchable indexes for both:

* Feature Types, and
* Property Types (which include association roles)

The abstract root of the Feature Types is always "Feature" defined in 4.6, Properties are attributes that describe feature instances usually represented by data types or relationship roles that represent associations to other features. The abstract root of all properties is always feature property defined in 4.14. Subclassed under properties is the feature relation defined in 4.15 which is a collection of feature relation roles defined in 4.16.

1. Each feature or property type should be accompanied by a semantically valid definition stating which real-world object might be represented or described by this type.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/general/Rec1
2. Each feature or property type may be defined by a reference to a standard language-specific dictionary.  
   https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per1

Note to Per1: In English, the Oxford and Webster’s dictionary may be the default for "British" and/or "American" English. More recent Oxford dictionaries carry extensive variations in spelling that can cover several dialects.

1. Each feature instance may be indexed by any number of semantically consistent feature types and be associated to and indexed by any number of properties   
   https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per2
2. In a Feature Ontology, further restrictions may be included to enforce semantic constraints on the data sets. These constraints may include information about valid property and association for individual feature types.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per3
3. A single feature instance may be associated to multiple, semantically consistent feature types.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/general/Per4

Example: A bridge may carry a road way, a railway, a canal or a walking path, so that the bridge’s geometry is likely to be part of several feature types even though it is one physical object (real-world phenomena).

In a Feature Ontology as defined in this document, definitions for features, properties and relationships specify the possibilities for the data representations; i.e. the form of the digital objects each representing one possible view of a feature – see [4], [7], [8], [9], [11], [15] and [19]. An ontology must first contain a taxonomy but can go further to include constraints consistent with those real-world features that can enforce semantic consistency. A feature data set conformant to an ontology is one in which the feature, properties and relationship are semantically consistent with their definitions in the specified taxonomy and other ontological constraints. Since these constraints should reflect "real-world" truths, so in a sense, a valid taxonomy-structured data set which matches the real-world should pass any valid ontology constraints specified by the ontology. In other words, those constraints are test for the quality of data collection in its representation of reality. If a collected data set is known to be validly consistent with reality, then a "violation" of an ontological constraint may be a flaw in the constraint, not necessarily in the collection.

Note: In this document, the terms “feature”, “class”, “category” and “taxon” may be used interchangeably. They are essentially different metaphorical views of the same concepts. In a web view, instances of a feature class are feature “resources” and will often be associated with URI identifiers.

“Whereas current Web content has implemented the separation of content from presentation to a large extent, the Semantic Web aims to externalize the inherent semantics from syntax, structure and other considerations. This has led to a layered characterization of metadata that have been used to capture these various aspects of information”, see [15], p 24. This separation of concern leads to two meta-requirements (see[15], p 26):

* “To enable the abstraction of representation details such as the format and organization of data and capture the information content of the underlying data independent of representational details”[[2]](#footnote-2).
* “To enable the representation of domain knowledge describing the information domain to which the underlying data belongs”[[3]](#footnote-3).

The solution for this separation of semantics and structure is found in the use of both schema and ontologies for the definition of data, see [15]. Chapters 5, 6 and 7. Classical structured data formats such as relational or XML data require schemata, but the merging of that data requires the understanding of the underlying semantics. Fluid data structures such as JSON (JavaScript Object Notation) or WKT (well-known text) are based on key-value structures which are dependent on data definition and mostly independent of data structure. The merging of these flexible data formats is dependent on a common set of keys, and a common set of definition for those keys (a common set or mergeable sets of taxonomy metadata).

If the data set is controlled by a taxonomy, each feature instance can incorporate any classes, attributes or associations that are consistent with the semantics of the definitions involved. In this case, each feature instance is essentially its own class. “Dynamic” programming languages such as JavaScript (objects represented by JSON) and various forms of LISP, see [17], can support this approach. Even a strict object language can be used to build a dynamic class system using associations in lieu of compile-time linkages, see Figure B.1. Ontology languages begin with a taxonomy but can also add other semantic restrictions like those in an object model. Object models can be mapped to ontologies, but not all ontologies are as restrictive as a non-dynamic UML model. Even in this case, it would be possible to include a feature relation that interlinks object representing the same physical object.

1. In a non-dynamic object model, if two separate class instances represent the same physical object, then there should be an "alternate representation relation" that links those instances based on that equality.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/general/Rec2

In a feature schema implementation, definitions for each feature type including the properties that might or must be contained and the relationship in which it might or must participate. Since the items in a schema are the same as the items in an ontology, a feature schema should contain a feature ontology and taxonomy (or a reference to a default dictionary). This recommendation for an ontology to be associated to each schema is new but has been a best practice for decades. As mentioned, it is the integration of information management in the semantic web that is driving this movement to a more formal semantic approach to data handling.

In cases where the "dictionary definitions" for the names of the features, properties or relations is sufficient to meet the ontology’s requirements, then these definitions will be sufficient. In cases where multiple languages are used in the application community for the ontology, a well-structured multi-lingual cross walk will be necessary to alleviate issues of interoperability within multi-lingual environments.

## Conformance targets

This standard defines conformance for combinations of associated instances as follows:

1. A taxonomy and the features and properties it allows consistent with the taxonomy’s definitions,
2. An ontology is a taxonomy where the entities are consistent with the ontology’s restrictions and
3. A schema is an ontology which fully defines the structures of each feature and property type allowed.

A taxonomy lists all feature and property types. A single feature instance can have any number of feature type "tags" and any number of property-value pairs if the semantics of the combination are valid, which will be true if the feature instance reflects the existence and state of the real phenomena that the feature represents, see 4.6.

An ontology consistent with this standard will contain a taxonomy, see **Req25**, and include constraints, based on the semantics of the interrelationships of the features and properties. In the best circumstances, the features possible in an ontology will be the same as would be collected by the taxonomy alone which, implying that the listed constraints reflect reality.

A schema consistent with this standard will contain a taxonomy, see **Req27**, but many schemata consistent with previous standards will not. In most of those cases, it defines feature-type and property-type names consistent with geographic tradition and are consistent with common dictionary entries (e.g. Oxford or Webster’s). A schema is more likely to be associated to a specialized application which requires consistent object structures. Schemata tend to segregate feature types, possibly by theme, however the case where a real-world feature/phenomenon is represented by two separate feature types, depending on the system, the digital representation of this feature may be separated into multiple entity/object classes based on applications implying that the real-world feature is instantiated as a digital entity multiple times. If the system recognizes this case, it has several mechanisms to represent the reality:

1. A single feature-identity is used for multiple digital-entities which have separate software instantiations, having different schema representations.
2. In a dynamic object model where a single object/entity can instantiate multiple schema classes, the data instances can be structurally isomorphic to the standard taxonomic model.

## Existing Implementations

Some cloud computer implementations use taxonomies, or their near equivalents to structure, index and search for data, see [20]. Big Data application literature similarly discusses the use of taxonomies to control the semantic of tags and the integration of structured, semi-structured and unstructured data, see [1] and [9].

In the literature, the "taxonomy to ontology to schema" spectrum discussed in this standard is usually referred to as "unstructured to semi-structured to structured" data. The elements of the feature taxonomy are feature class names, the elements of the property taxonomy are property or association role names. Ontologies may add constraints on data element instances and schemata may add constraints which will include a full object model. These constraints are often most useful in testing data for consistency. Each instance of feature shall have a feature identity to facilitate association specification (see **Req18**).

# Normative references

The following publications contain normative information on the topics in this document.

ISO 19101-1: Geographic information — Reference model

ISO 19103: Geographic information — Conceptual schema language

ISO 19109: Geographic information — Rules for application schema

ISO 19111: Geographic information — Spatial referencing by coordinates

ISO 19126: Geographic information — Feature concept dictionaries and registers

ISO/IEC 13249-3: Information technology — SQL Multimedia and Application Packages -   
Part 3: Spatial

ISO 19162:2015 Geographic information — Well‑known text for coordinate reference systems   
<https://www.iso.org/standard/63094.html>   
OGC 12-063r3\_Well-known\_text\_for\_CRS   
<https://portal.opengeospatial.org/files/?artifact_id=57255&version=1>

SKOS: Simple Knowledge Organization System Reference.   
<https://www.w3.org/TR/skos-reference>.

ModSpec OGC 08-131r3 The Specification Model - A Standard for Modular specifications.   
<https://portal.opengeospatial.org/files/?artifact_id=34762>

# Definitions

In addition to the list below, any definition in any normative reference in Clause 2.2 will be acceptable. All Standard English words are in either in the Oxford or Webster's dictionary or both (sometimes with spelling differences) apply.

1. coercion,   
   type coercion,   
   implicit type conversion

〈programming〉 automatic conversion of a value in one type to another based on the equivalence of values

Note 1 to entry: Coercion will usually be implemented as a conversion of the value of the expressed type to an expression of a related type (that shares its semantic value but not its class structure). It is sufficient that there exists a conversion function taking the original value that would create an equivalent instance of the target class. If a function protocol requires a real number, then the integer (a subclass of real) value of "1" will be coerced into a real number of equivalent value, i.e. the real number format "1.0". Coercion is an automated ability of most programming language.

1. composite feature

〈modeling〉 feature with an extent consisting of a set of contained features

Note 1 to entry: Feature components (the contained features) are often labeled by the type of geometric primitive use to express its extent; giving point feature, line feature, area feature, solid feature (depending on the dimension of the geometric primitive being used. The spatial extent of a composite feature is the spatial union of the spatial extents of all its components.

1. container,   
   container object

〈programming〉 object which contains or is associated by some form of aggregation to other objects

Note 1 to entry: If a contained object does not specify optional attributes, then it may inherit the value of those attributes from its container. If its containers are not consistent (possible where the containment is by reference), then the contained object should contain values for all ambiguous attribute values.

1. controlled vocabulary   
   controlled taxonomy

〈taxonomy〉 established list of standardized terminology (names, words or phrases) with associated definitions for use to identify, describe, index or retrieve information

Note 1 to entry: The vocabulary is "controlled" in its use in the sense that only terms from the list may be used for the intended purpose. In this document, the principal vocabulary terms are the "feature", "property", "relations" and "relationship roles" in the taxonomy that controls items in a conformant feature data set. In its earliest uses, legal and medical communities used a well-defined list of terms to be used; in the legal community in the descriptions of legal opinions, reports, records and laws, and in the medical community to similarly control medical records and reports of both diseases and their treatments.

Note 2 to entry: Another view of the idea of controlled may also apply in the sense that in an information community using a controlled vocabulary would also have a control procedure or mechanism for its use and its evolution in the addition or modification of the terms. Hence both the vocabulary and the items built from that vocabulary are semantically controlled. For the scheme to work, all terms must be defined. The term "vocabulary" is often replaced by "thesaurus" or "taxonomy" indicating the importance of consistency of the formal, lexical and conceptual semantics in the vocabulary’s use. In its simplest use, vocabulary may connote a set of words, but the use to which a "controlled vocabulary" is put, the importance of the implied meaning is self-evident. It may be that "lexicon" (vocabulary of a branch of knowledge) might be more precise which is implied in using "terminology"(terms used with a technical application in a subject of study, theory, profession, etc.) in the definition in [16] .

Note 3 to entry: Vocabularies can and often include a referential index based on the use of terminology for subsequent query, locate and retrieve processes.

[Modified from [12], [13], [14] and [16]]

1. dynamic object

〈programming〉 object instance which is, as necessary, assumed to support any operation, property or relation

Note 1 to entry: Each dynamic object can be modified at run-time by the program adding or modifying an existing object. Once an object is dynamic, it becomes its own class, which can be "cloned" to create identical class instances. Metaphorically, the object can "learn" anything the program can "teach" it.

Note 2 to entry: Dynamic objects are or can be supported in several programming languages such as C# ("C-sharp"), JavaScript, Visual Basic, Python and even C. Layering a block style of primitive, such as in Figure B.1, can produce similar objects in a complied language environment. The programming language cannot constrict the inventiveness of the programmers.

1. feature

〈geographic information〉 abstraction of real-world phenomena or entity

[ISO 19101 and expanded in ISO 19109]

Note 1 to entry: Feature instances are data objects, and as such are distinguished by their object identity. The entity or phenomena that the feature represents are real-world objects. All feature instances that reference the same real-world phenomena should have the same feature identity but do not need to be stored together. Storage mechanisms are the purview of the applications. If the same feature has alternate representation, and are thus identified as distinct, this may lead to misinterpretation of the data. In this concept, the feature identity is distinct from the digital object identity which are always distinct identities if they are not stored as a single digital object.

1. feature association

〈geographic information〉 relationship that links one feature with another feature

[modified from ISO 19110 and ISO 19109]

Note 1 to entry: The original definition of feature association is "relationship that links instances of one feature type with instances of the same or a different feature type" and assumed a schema constrained feature model which is validated by feature types and object instances. An ontology-constrained feature model depends on comparisons of feature definitions with association definitions for validation. If conformance is by schema-constraints, the original definition is still valid.

1. feature catalog

〈geographic information〉 catalogue containing definitions and descriptions of the feature types, feature attributes and feature associations occurring in one or more sets of geographic data, together with any feature operations that may be applied

[ISO 19110]

1. feature class  
   feature type

〈geographic information〉 class model of a feature with common aspects

Note 1 to entry: A class for this interpretation is "intentional" in the sense that it is defined by general characteristics that must be shared by all its members. "Extensional" classes are defined by which objects are in them, which may have no common characteristics.

1. feature component

〈geographic information〉 feature with only one geometric attribute which is a geometric primitive appropriate to the dimension of the coordinate reference system

Note 1 to entry: Feature components are often labeled by the type of geometric primitive used to express its extent; giving point feature, line feature, area feature, solid feature (depending on the dimension of the geometric primitive being used.

1. feature concept

〈geographic information〉 concept that may be specified in detail as one or more feature types

EXAMPLE The feature concept “road” may be used to specify several different feature types, each with a different set of properties appropriate for an application. For a travel planning application, it might have a limited set of attributes such as name, route number, location and number of lanes, while for a maintenance application it might have an extensive set of attributes detailing the structure and composition of each of the layers of material for which it is composed.

Note 1 to entry: A feature concept can be thought of as an "abstract root" or "superclass" of a related set of feature types. In an ontology specification, this should be instantiated as a "is a" relationship at the definition level. Similarly, in a schema, this should be instantiated in the subclassing hierarchy.

[ISO 19126]

1. feature concept dictionary

〈geographic information〉 dictionary that contains definitions of and related descriptive information about concepts that may be specified in detail in a feature catalogue

Note 1 to entry: This is essentially the root rationale for using an "ontology" which merges the concept of "definition" and "class".

[ISO 19126]

1. feature identity

〈programming〉 unique identifier associated to a real-world phenomenon

Note 1 to entry: This identifier identifies the reality of the thing being described by a feature object. This is distinct from the object identity of a feature instance. Compare "feature" and "feature identity" with "object" and "object identity". In general, supporting the distinction between object and feature is an application decision, but it can lead to semantics issues with query results. Most query languages search for object or data instances of which a feature may associated to many, but more semantically correct statements are about features.

1. feature property   
   feature attribute

〈geographic information〉 potential attribute, quality, or characteristic of a feature

[modified from ISO 19101]

Note 1 to entry: The original definition of feature attribute used "characteristic" instead of "property". In general semantics, the two definitions are identical in interpretation, but since this document uses the more generic term "property", the definition has been adjusted to coincide with terminology both in ontology-constraints and schema-constraints.

Note 2 to entry: This definition should not be construed to indicate how a feature property is implemented. Any containment, or logical reference can be used in the various implementations of property.

1. feature relation  
   feature association

〈geographic information〉 relationship that links instances of one feature type with instances of the same or different feature types

Note 1 to entry: In the original definition, the last "type" in the definition was singular, which incorrectly implies that all relations are binary. In database terminology, a relation can reference any number of entities. A relation on sets S1, S2, S3, S4, … Sn can be defined as a subset of the cross-product of the n sets (a set of n-tuples).

[19110], [5]

1. feature relation role

〈geographic information〉 name or semantic purpose of an offset in a feature relationship

Note 1 to entry: Roles in a relation can be implemented as references or pointers in any number of the objects in the relation instance.

1. feature schema

〈geographic information〉 collection of definitions for feature classes, properties and associations

1. feature subclass  
   feature subtype

〈geographic information〉 feature class or type, whose members are also members of a superclass

Note 1 to entry: This implies that "broader" or "narrower" feature classes are defined in a transitive manner. This means subclass is "transitive narrower" than its superclasses and a superclass is a "transitive broader" than its subclass.

Example: The abstract root class of "feature" contains all "features"

1. hierarchical taxonomy

〈taxonomy〉 taxonomy such that categories are arranged in a tree structure built to represent the context of a category, using parent-child "is-a" relationships where the parent is broader conceptually than the child

Note 1 to term: Some sources require that the hierarchy be single inheritance (only one superclass per category), but this is not required in general and is inconsistent with dynamic class models.

Note 2 to term: Some taxonomies also use a containment view of the terms broader and narrower to include containment hierarchy so that a "door" is an element of a building means the "door" is a narrow concept than building. In a hierarchical content, door and building are not in the "is-a" hierarchy and cannot be used to mimic object inheritance as can be done in a hierarchical "is-a" taxonomy.

1. identifier

〈programming, web〉 linguistically independent sequence of characters capable of uniquely and permanently identifying that with which it is associated

[ISO 19135]

1. key value pairs

〈programming〉 ordered pair of two data items; first, the key (identifier) identifying the semantics of the associated value, and second, the value, either the identified data or a reference (pointer or identity) to that data

Note 1 to entry: In this document the common expression will be formatted as: "〈key〉: 〈value〉".

1. name

compact and human-readable designator that is used to denote and identify a data item

Note 1 to term: Aliases may be defined for use within the scope of an information system, but data sets used in information exchange use standard specified name for all data items.

1. object

〈programming〉 identifiable component of a software system or design, now more commonly applied to a component that is in some sense self-contained, having an identifiable boundary.

Note 1 to entry: An object is an instance of a class in object-oriented programming, but the term may be used generically in computer programming to indicate an instance of any software component. This standard is concerned with the "objects" that contain information about real world phenomena or features. A feature instance is an object, and as such will be supplied with an object identity by the programming system (often related to physical addresses in a program’s image.

[modified from A Dictionary of Computer Science,[3]]

1. object identity

〈programming〉 identifier associated to an instance of an object to distinguish it from all other objects

Note 1 to entry: The object identity of a feature instance is not the same as the feature identity. The object identity names a block of data, the feature identity names the real-world phenomena that the object is representing. The object identity is unique to a block of data. The feature identity should be unique to a block of reality. The two concepts are often confused and the moniker "feature identity" is often used to describe the object identity of a feature class instance.

1. ontology

〈semantics〉 formal naming and definition of types, properties and relationships of entities that exist in a domain of discourse.

Note 1 to entry: The definitions in a feature ontology include features and feature properties (attributes, associations and roles). An ontology defines a set of representational primitives with which to model a domain of knowledge or discourse and can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models. See [13].

1. ontological schema

〈programming〉 taxonomic schema with additional ontological constraints

Note 1 to entry: Any object schema is an ontological schema that included all appropriate object-oriented constraints for the appropriate OOPL targeted.

1. property

〈modeling〉 named descriptive value associated to or contained in an entity (object or feature type)

Note 1 to entry: A property’s name normally reflects the semantics of the property’s value, which may be a value or the identity of an associated entity. Properties linking entity to one another are relations. Properties associating descriptive data values to an entity are attributes.

1. semantic class

〈ontology, taxonomy〉 set of entities consistent with a taxonomy definition consisting of a necessary and sufficient set of criteria for an entity being an element of the class.

Note 1 to entry: The set of criteria of a semantic class will be necessary but not sufficient for any of its subclasses. A subclass is necessarily more restrictive than any of its superclasses as they inherit their superclass’s criteria but must distinguish themselves as more specific and non-redundant.

1. semantic class hierarchy

〈ontology, taxonomy〉 hierarchy of sematic classes based on subseting

Note 1 to item Any logical subset (B) of a semantic class(A), based on the a more specific definition is a logical subclass of that class (A).

1. sequence

〈mathematics〉 ordered set of items in which the order has semantic importance

1. schema

〈modeling〉 representation of a plan or theory in the form of an outline or model.

Note 1 to item A schema can be expressed as a plan or theory in the form of an outline or model. The realization can be based on definition of entities in the model (**ontology**) or as structures representing those entities (**schema**), or as both.

1. set

〈mathematics〉 unordered collection of related items with no repetition in which the order is of no importance nor significance

1. taxon  
   class  
   category  
   type

〈taxonomy〉 classifying entity in a taxonomy consisting of a name, a definition and relations to other such entities in the taxonomy

Note 1 to term: The terms are interchangeable, in the sense that a "taxon" in a taxonomy will become a "class" in an object schema, or a type in a "query language", as appropriate. The most common type of relationship is a hierarchical one listing classes as broader (superclass) or narrower (subclass). See **hierarchical taxonomy**.

1. taxonomy

〈taxonomy〉 system for classifying entities by logical definitions (criteria) and a hierarchy subtype relationship base on logical subsets of instances in a class

Note 1 to term: Taxonomies are often used as indexes to support and optimize searches based either on entity types, property types or values. In most of the instances in this document, a taxonomy will tend towards being a controlled vocabulary, basically since the more controlled across users and user communities the better the affect toward interoperability of datasets.

1. theme  
   layer  
   thematic layer

〈geographic information〉 abstract root classification for a feature hierarchy included feature which contribute to a single geographic subject

Note 1 to term: Themes might include both geological and sociological theme such as geology, hydrological, political, cadaster, political and sociological themes. Theme will often parallel a type of geographical related area of study. See [18]

# Symbols (and abbreviated terms)

The following abbreviations, acronyms, symbols and names are important key to understanding the normative or informational content to follow. They are in general known publicly, and readers of this document may need to understand their meaning.

|  |  |
| --- | --- |
| « » | limits of a UML stereotype or keyword, either in text or a diagram; the symbols are "guillemets" |
| API | Application Programming Interface |
| ATS | Abstract Test Suite |
| C++, C# | Programming languages based on C with object-oriented extensions, "C plus plus", "C sharp" |
| CAD/CAM | Computer Aided Design, Drafting or Manufacture, Mapping (usually geometry-based design systems) |
| CRS | Coordinate Reference System, usually a reference to an instance defined in ISO 19111 |
| JavaScript | prototype-based scripting language with objects, dynamic typing and first-class functions |
| JSON | JavaScript Object Notation, alpha-numeric format for JavaScript objects |
| LISP | Programming language based on LISt Processing. The standard is Common LISP. |
| MBR | Minimum Bounding Region (sometimes Rectangle) |
| RSID | Reference System Identifier |
| SQL 3 | Common name for SQL 99 during its development |
| SQL 99 | SQL language adopted in 1999, which includes object-oriented data types |
| SQL/MM | SQL Multi Media extensions for SQL, see [ISO/IEC 13249-3](file:///C:\Users\Public\Public%20Dropbox\Dropbox\WG9\Drafts\Features%20and%20their%20geometry%20well%20known%20text.docx#ISOIEC13249_3) |
| URI | Uniform Resource Identifier (IETF RFC 3986) |
| WKB | Well-Known Binary (data represented as standardized structured binary) |
| WKT | Well-Known Text (data represented as standardized structured text) |
|  | the empty set, i.e. containing no elements |

# Feature models

## Introduction to feature taxonomy

A feature is a representation of a real-world entity/object. This idea can trace its definition back to the 1960’s in the definition of Simula (a simulation language), the first object-oriented programming language. In Simula, "digital objects" represented "real-world objects"[[4]](#footnote-4). Since the 1980’s, spatial or geographic information systems have paralleled this same idea, at about the same time as Simula and OOPL were first being used. Much of the early standardization of GIS followed the same object-oriented paradigm and defined feature classes as object classes (often in UML or it predecessors, such as OMT object modeling technique).

The problem with digital objects representing real world objects is that, in most schematic modeling, all objects of the same class have essentially the same representation. The "real-world" is not as neat and clean. Any two real-world objects are distinct, often at a fundamental level and, for that reason, each may be a unique instance of their own "class", e.g. must be described in different manners with different properties, not just different values for the same property. While feature schemata are the core of many systems and their associated standards, there is another approach that supports schema but allows more flexibility as needed. This approach steps back from structural formalism and uses semantic **taxonomy** and **ontology**, to define feature types and prescribes descriptive properties of features as related parts of a **taxonomy**. In contrast with a schema, where a class is defined by its properties, a taxonomy-based system allows any feature to be associated with any property where the feature and property definitions are consistent with one another according to a given taxonomy.

A geographic feature taxonomy, ontology or schema will define a namespace that contains definitions of the keys that will be used to identify the semantics of key-value pairs associated to a feature:

* a set of feature types (a type name and a definition)
  + including a "is a type of" relation where a definition may references a more general type,
    - e.g. "weir is a type of dam such that"…
* a set of feature properties:
  + a set of feature attributes (an attribute name, a definition and a data structure or object to represent its value), and
  + a set of feature relations that associate features to one another (a name, a definition, a list of rolenames and pointer or reference type to identify the feature in each role).

A feature instance contains:

1. a unique local identity, (possibly part of a universal identity scheme such as URN)
2. one or more feature type names,
3. a list of properties with attribute names and associated values,
4. a list if feature relationship names each with a list of roles and the target feature identities for those roles.

A geographic feature schema is an ontology that further defines which properties and relations that can be (optional) or that must be (mandatory) associated to instances of which feature types. All properties not specifically mentioned in this list cannot be part of the structure of a feature of that type. Table 1 below shows how a set of features could be represented and gives some potential examples of a taxonomy for feature type definition that are used in the example.

Classical schema formalism arises from application requirements. For example, a radar-based application would concentrate on the attributes of a feature that affect its radar reflectivity, but a visual based application would concentrate on the visual attributes of a feature. Radar is not good at detecting color, but human eyes are. A radar system would not care what color the Golden Gate Bridge is painted, but a system for visual identification would know "international orange" is a good visual recognition clue; in fact, the "color is used in the aerospace industry to set things apart from their surroundings". See [10]. The use of "international orange" is of interest to a "visual-based" application, but not so much to a "radar reflectivity" application. The "interoperability" advantages of a single database for multiple application are obvious, especially when a single operation uses detection approaches in the "DC to light"[[5]](#footnote-5) spectrum range.

As different applications require integrated formats and information, it is easier to integrate taxonomies than schemata. In fact, integration of schemata often requires that a taxonomy be constructed to map the semantics of attributes and entity names used in the labels of the various schema. These taxonomies can then be compared to determine how the various application schemata can be integrated into a generic "application-independent" format.

Different applications will often have different schemata, and different data collection rules. If such applications are required for a single operation or project, the interoperability advantages inherent in a single datastore, and a single set of "facts" is obvious and usually essential.

Once the decision to create this "common database" the use of a "taxonomy merge" in unavoidable. Some features will be only one source dataset, but the majority of the "real-world entities" will have multiple instance features to merge. The first step is the merging of the taxonomies of the disparate datasets, then followed by identifying which of those entities are associated to multiple feature instances, which requires the merging of different object classes from different applications. Easy to do in a taxonomy, difficult to do in schema. Over time, the holes may be filled, and the merged dataset will eventually be an application-independent stores that supports all participants in the collections of the data; and the synergistic interoperability of the applications.

The driving purpose of OGC is and has always been making applications interoperable. A taxonomy paradigm for datastores seems to be a rational route to creating synergistic applications that have a single view of the world because they can depend on a merged, complete and flexible taxonomic datastore.

The values of attribute properties will have to be represented in a format consistent with the applications supported. In the textual examples in this standard those types may be in text, such as a JSON, YAML or WKT.

We can represent Wheeler Dam as in Table 1; the "URL #" expressions would be system generated feature ids.

Table 1 – Example feature entities in a key: value format

|  |
| --- |
| { Feature: "URL 1" ‼ [some URL]  FeatureType: "Dam"  FeatureType: "Bridge"  Name: "Wheeler Dam"  On: "URL 3" ‼ Tennessee River  CenterPoint 34°48'14.99" N, -87°22'32.99" W  Geometry: Location ‼ [LineString expression] }  { Feature: "URL 2" ‼ [some URL]  FeatureType: "Lock"  On: "URL 3" ‼ Tennessee River  In: "URL 1" ‼ Wheeler Dam }  {  Feature: "URL 1" ‼ [some URL]  FeatureType: River  Name: Tennessee River } |

In a taxonomy-controlled feature data set, the features and their properties are defined. The association of feature instances to properties only require that the definitions be consistent with data instances. Feature instances would beinstances associated to attributes and associations defined in the taxonomy and associated to instances as needed. Some programming languages (e.g. JavaScript and LISP) can handle this directly, but object-oriented languages that can be expressed in UML can be programmed to do so also. Taxonomy instances for feature types would specify: a feature type name (tag) and definition, feature type inheritance, and mandatory properties.

In a schema-controlled feature data set, the properties are defined by the controlling schema. For example, the "logical default" for a dam does not have the properties of a bridge, but in the real world several dams have been designed or modified to include a bridge in its superstructure (such as "Hoover Dam" on the Colorado, and "Wheeler Dam" on the Tennessee). In a parallel case, some but not all dams have hydroelectric properties or flood control mechanism, again optional descriptions are needed to create a complete description dependent on the applications using the data. In a schema-controlled data set, the class for "dam" would have to be able to coexist with "bridge" when it is needed. Considering the complexity of reality and difficulty of an *a priori* schema that could adapt to all possibilities, the ontological approach is more likely to be usable without continuous redefinition. A schema would require potentially dynamic multiple inheritance or instantiation in a dynamic object-oriented language.

Each of the "dams" serves as part of a water control system (possibly the main purpose of a "dam"), a link in the road system across a waterway (the main purpose of a "bridge") and a source of hydroelectrical power (the main purpose of a "power plant"). This feature-type combination in a single feature is not universal. Some bridges are not associated to neither flood control nor hydroelectric dams. In New York state, the [Robert Moses Niagara Hydroelectric Power Station](https://en.wikipedia.org/wiki/Robert_Moses_Niagara_Power_Plant) diverts water from above the Niagara Falls and routes it below the falls while generating power. This hydroelectric power plant is not associated to a dam and occupies shoreline only on the New York side of the Niagara River. It also has a Canadian near-twin in the [Sir Adam Beck Hydroelectric Power Stations](https://en.wikipedia.org/wiki/Sir_Adam_Beck_Hydroelectric_Power_Stations) in Ontario, Canada. To minimize erosion at the fall’s face, both Niagara facilities are used to affect the amount of water going over the falls at Niagara. The Wheeler Dam on the Tennessee is also used for mosquito control; Wheeler Lake’s water level is periodically adjusted up or down to disrupt the mosquito life-cycle either drying out or washing out mosquito larvae. This extra-model information is important to only some applications, but to those applications they are often essential. **Error! Not a valid bookmark self-reference.** **Error! Not a valid bookmark self-reference.**, gives some potential examples of a feature type definition.

Table 2 – Example feature-type taxonomy entries in a key-value format

|  |
| --- |
| { FeatureType: Feature  Definition: abstraction of real-world phenomena or entity  Property: location: Geometry  Property: coordinateSystem: CRS }  { Relation: Subset  SorceRole superset|covers:x  TargetRole subset|covered|in|on:y  Type geometric relation (derived)  Definition: y.geometry is subset of x.geometry }  { FeatureType: Dam  Definition: barrier to stop or control the flow of water  Property: location: Geometry  ProperyRole: on: waterway  Property: name: Text  ProperyRole: Contains: Sluice[0..\*] }  { FeatureType: Weir  Defintion: barrier built across a waterway to make the water  approaching it deeper, usually by making the water  flow over a part of the barrier at a controlled height.  IsTypeOf: Dam }  { FeatureType: Lock  Definition: section of a waterway, enclosed by gates, in which the  amount of water can be modified to raise or lower a vessel  IsTypeOf: Sluice }  { FeatureType: Sluice  Definition: sliding gate or other device for controlling the flow of water  Property: on:Waterway  Property: in:Dam(0,1) }  { FeatureType: Watercourse  Defintion: natural or artificial water channel }  { FeatureType: Waterway  Defintion: Watercourse usable for travel  Property: navigability:NavigabilityParameters  IsTypeOf: Watercourse }  { FeatureType: Canal  Defintion: artificial Waterway  IsTypeOf: Waterway } |

The "use of taxonomy" versus the "use of schema" is a shift of emphasis from syntax (the form or structure of the information) to semantics (the meaning of the information). This transition should not be a surprise since from the beginning the great barrier to the flow of information has been format i.e. schema, and so the biggest boon to interoperability is to step over that barrier and shift our application in the same direction that the web is moving, towards sematic solutions. This makes the data easier to understand and thus easier to move, and finally easier to interoperate between functionality.

Requirements Class: Feature Taxonomy and Data:  
https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/

### Feature types and thematic layering

A feature taxonomy will always contain at least two collections of definitions (controlled taxonomies, or controlled vocabularies, see [2], [13], and [14]) that separately define and describe "real-world" features and properties. See Figure 1. A feature data set consistent with this feature taxonomy.

1. A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace.  
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req1
2. A taxonomy shall contain a complete semantically valid definition for each feature type and for each property type associated to the taxonomy by its namespace.  
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req2
3. The feature type component of a taxonomy shall include a "is a" type-hierarchy for features (e.g. a hierarchical taxonomy, engendering a multiple inheritance of feature type).  
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req3

Note: A hierarchical taxonomy allows the category/class structure to be inheritable like an object classes inheritance structure. If a taxonomy is used in a class model schema, the taxonomy hierarchy would be identical to the class hierarchy.

1. In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy; i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req4
2. The taxonomy entry for a feature type shall contain a name, a definition, a list of any direct superclasses and any required properties.   
   [https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/](https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req5)Req5

Note: Any other property type consistent with the feature definition would be optional in this case. In an ontology extension, constraints could be used to forbid specific properties.

1. The taxonomy entry for a feature type may contain common optional properties for entities of the feature type.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per5
2. Any feature instance may reference any number of semantically consistent feature types.  
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per6
3. Any taxonomy item may be associated to searching for instances of that type of item.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per7
4. Any property instance may be used to describe any feature instance if the semantic of the feature and property make logical sense.   
   [https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/](https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per4)Per8

Note: Ontology extensions would be able to prespecify consistent property, feature pairs.

1. The taxonomy entry for a property type shall contain a name, a definition, a default data type able to represent the value of the property.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req6
2. In an instance of a property, the value may be any datatype coercible into the default datatype.   
   [https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/](https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per5)Per9

Note: Type coercion is defined in 4.1, page 13.

Note: Linkages could be accomplished by URI associations, but other pointer types may be more appropriate for the representation format used for the taxonomy.

1. Each data entity consistent with a taxonomy shall be a dynamic object, contain a local identity associated to its name space and type names used to link it to semantically appropriate feature, property, role, or association names.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req7
2. If an implementation cannot support dynamic objects, then any number of static objects may be associated to the same feature identity.
3. A feature taxonomy shall contain a non-empty set of definitions of real-world phenomena "features" that may be represented as features in datasets consistent with this taxonomy.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req8
4. Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of prosperities or associations consistent with its definition.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req9
5. Theme or thematic layer definitions may be root classifiers for feature type definitions.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/ Per11

See definition 4.35

1. Root classes in a feature taxonomy may be themes or thematic layers.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per12
2. Any feature type definition may contain any applicable subtype of ("is-a") relationship (subclass) to other feature definitions.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per13
3. Every feature instance shall have a feature identifier that represents the real-world thing being described by the feature.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req10
4. Any feature instance shall be consistent with the corresponding feature type and property definitions in the associated taxonomy with which it is associated.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req11
5. Any feature instance may contain properties or be involved in associations consistent with its definition.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec3
6. Any feature instances that reference the same real-world phenomena should have the same feature identifier.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec4
7. A feature access process shall be able to collect all feature instances that refer to the same real-world entity.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req12
8. A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination of the same.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req13

### Feature properties

A feature model will always contain a non-empty collection of definitions (taxonomy) that either define or describe the *potential* properties of "real-world" features. The use of any property type to describe any feature type would be the purview of the collection mechanism and what situational peculiarities may require or preclude the use of the properties in the situation.

1. A feature taxonomy will contain a non-empty set of definitions of properties of real-world phenomena that may be represented as properties associated to features in data sets consistent with this ontology.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req14



Figure 1 — Basic Feature Taxonomy Structure

1. Properties shall have values that are datatypes, including references to other features.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req15
2. A feature property definition shall include a unique property name and a property type.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req16
3. Each instance of feature shall have a unique object identity.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req17
4. Each instance of feature shall have a feature identity.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req18
5. Each real-world phenomenon may be referenced by multiple feature instances.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per14
6. Any two features labeled with the same feature identity shall reference the same real-world phenomenon.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req19
7. Any two features instances that reference the same real-world phenomenon should where feasible have the same feature identity.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec5

This last recommendation may be difficult, especially across data sets. Object identities are easier to keep unique, if a hierarchical identity namespace system such as URN is used. Global grid systems may be useful in defining feature identity system based on location, but dynamic datums, which recognize a dynamic location issue, may be an encumbrance. Unfortunately, the planet is not as stable as we once thought, and we now know things move even without our permission nor intervention.

1. Each feature shall have at least one feature type.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req20

It will of course be possible to have a feature type called "Unknown" but knowing that we do not know is knowledge worth having (akin to Socratic Ignorance, "I do not think I know what I do not know,").

1. Each feature instance may have a property "Type" which contains all feature types which this feature instantiates.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per15
2. Each feature will be considered a member of each feature type in its type references and any of their supertypes.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req21
3. Each feature instance shall have complete ownership of all its property instances values.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req22
4. If a feature is in a containment hierarchy, each property should be present in the largest element consistent with the semantics.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec6

Example: Rivers are often broken into reaches (segments of river with no junctions with other waterway excepts at it endpoints). The name of the river, which applies to all its reaches, should normally be associated with the largest segment who shares that name.

1. A feature property’s definition may contain or be augmented by restrictions on which feature types it describes.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per16

### Feature associations

A feature taxonomy or ontology will contain a collection of definitions that either define or describe the potential associations between "real-world" features.

1. A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req23
2. Binary association shall be implemented as symmetric roles linking each feature in the relation to the other.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req24
3. A "n-ary" relation may be implemented as a "relation feature" with n binary relations named for the roles in the original relation.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Per17

### Feature Component

A feature component is feature whose geometry is a geometric primitive. In 2D or 2½ data, the primitives would be points, curves and areas (surfaces). In 3D, solids would be included.

1. A feature containing multiple types of feature primitive should be decomposed into feature components each containing a single geometric primitive. The original feature should then be represented as an aggregation of its components.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec7

Example: Roads are built in segments, usually intersection to intersection, but highways can then be described as a weak aggregation of road segments, some of which carry multiple “route marker” enumerating the shared highway numbers which only change at segment boundaries.

### Composite feature and their hierarchies

A composite feature is an aggregation of smaller, often simpler, features.

1. A feature hierarchy should consist of shared aggregations of composite features and feature components, terminating at the bottom as feature components.   
   [https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec8](https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Rec7)

Requirements Class: Ontology:   
https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/

An ontology includes a taxonomy for feature and property. It may add restrictions, for example, limiting which properties are associated to which features. Some describe taxonomy-controlled data as "unstructured" and describe schema-controlled data as "structured". Having labeled the extremes, they declare the choice as binary, missing the very broad middle ground, often called "semi-structured".

1. A feature ontology shall be fully compliant as a feature taxonomy.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req25
2. A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req26
3. An ontology may add constraints on feature types and properties.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Per18

Requirements Class: Feature schema:  
https://opengeospatial.org/as/FG/P1/FM/1.0/schema/

A feature schema is a feature ontology that has a more complete set of constraints consistent with a classical object model.

1. A feature schema shall be fully compliant with a feature ontology.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req27
2. A feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties and data types required by the definitions in the ontology.   
   https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req28

1. Conformance Classes
   1. Test Suite for Taxonomy   
      https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/

An implementation of the Conformance class Taxonomy shall contain valid definitions for feature and property types, meeting the requirements in Clause **6.2**. Each clause below shall associate the numbered requirements to a generic method of verification. Each test is associated to a single requirement defined in the requirements class.

* + 1. A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace.  
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req1/test

Test to identify the namespace of the taxonomy.

* + 1. A taxonomy shall contain a complete semantically valid definition for each feature type and for each property type associated to the taxonomy by its namespace.  
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req2/test

Test to assure definitions are complete and non-ambiguous (without contradictory descriptions). Property types will also contain a feature type, an object or a data type to represent the value of the defined property. Feature type valued properties are association roles.

* + 1. The feature type component of a taxonomy shall include a "is a" type-hierarchy for features (e.g. a hierarchical taxonomy, engendering a multiple inheritance of feature type).  
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req3/test

Test to verify that the feature type and property hierarchical taxonomies support a “is-a" or "is a subtype of " so that object inheritance is mimicked. All feature types should be transitively a subtype of the abstract root "feature".

* + 1. In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy; i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req4/test

Test to verify that results from a query statement include all viable subtypes of the feature types named.

* + 1. The taxonomy entry for a feature type shall contain a name, a definition, a list of any direct superclasses and any required properties.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req5/test

Test to verify that each feature type definition is complete, including superclass and required property list. There will always be supertypes or be directly inherited from the abstract root "feature".

* + 1. The taxonomy entry for a property type shall contain a name, a definition, a default data type able to represent the value of the property.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req6/test

Test to verify that each property type definition is complete.

* + 1. Each data entity consistent with a taxonomy shall be a dynamic object, contain a local identity associated to its name space and type names used to link it to semantically appropriate feature, property, role, or association names.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req7/test

Test to verify that each object instance can modify its properties and feature type list within the constraints of the taxonomy.

* + 1. A feature taxonomy shall contain a non-empty set of definitions of real-world phenomena "features" that may be represented as features in datasets consistent with this taxonomy.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req8/test

Test to verify that the manipulation of features does not cause any constraints of the ontology to be violated.

* + 1. Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of prosperities or associations consistent with its definition.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req9/test

Test to verify that all feature instances are associated to at least one feature definition in the feature taxonomy.

* + 1. Every feature instance shall have a feature identifier that represents the real-world thing being described by the feature.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req10/test

Test to verify that all feature instances have a feature identifier, which is consistent with all other feature instances associated to the same real-world feature.

* + 1. Any feature instance shall be consistent with the corresponding feature type and property definitions in the associated taxonomy with which it is associated.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req11/test

Test to verify that the definition of each feature type is semantically consistent with the property types associated to the feature instances associate to that feature type.

* + 1. A feature access process shall be able to collect all feature instances that refer to the same real-world entity.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req12/test

Test to see that all feature instances of the same real-world entity have the same feature identity.

* + 1. A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination of the same.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req13/test

Test to verify that all feature instances of the same feature type are locatable by type, included properties and the values of properties. The usual implementation is indexed feature types and property values.

* + 1. A feature taxonomy will contain a non-empty set of definitions of properties of real-world phenomena that may be represented as properties associated to features in data sets consistent with this ontology.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req14/test

Test to verify that properties and association roles are defined and can be associated to features. Test to verify association roles can link to features using either the feature or object identity value, or equivalent functionality.

* + 1. Properties shall have values that are datatypes, including references to other features.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req15/test

Test to make sure that all property definitions include a datatype to represent its value. This may include pointers to other features, e.g. the feature identity of a targeted feature.

* + 1. A feature property definition shall include a unique property name and a property type.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req16/test

Test by inspection that each property has a unique name, and an appropriate value consistent with its taxon definition.

* + 1. Each instance of feature shall have a unique object identity.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req17/test

Test that the feature identity of each feature is unique and can be used as a reference to implement property roles as pointers.

* + 1. Each instance of feature shall have a feature identity.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req18/test

Test that each feature has a feature identity.

* + 1. Any two features labeled with the same feature identity shall reference the same real-world phenomenon.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req19/test

Test that equal feature identities always reference the same real-world entity. This does not mean that there is only one object (identity) for a feature.

* + 1. Each feature shall have at least one feature type.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req20/test

Test to ensure that feature type is mandatory for each feature instance.

* + 1. Each feature will be considered a member of each feature type in its type references and any of their supertypes.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req21/test

Test that any retrieval of feature by feature type include all objects by that feature type and any of its subtypes as defined by the taxonomy hierarchy.

* + 1. Each feature instance shall have complete ownership of all its property instances values.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req22/test

Test to ensure that each property instance is associated to a single feature object instance.

* + 1. A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req23/test

Test to ensure that feature associations are defined by sets of association roles, e.g. by properties whose values are references to features, either by feature identities by object identities.

* + 1. Binary association shall be implemented as symmetric roles linking each feature in the relation to the other.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/taxonomy/Req24/test

Test to ensure that property entities are not shared to the extent that no property value of any feature can be modified except by navigation through the owning feature instance.

* 1. Test Suite for Ontology   
     https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/

An implementation the Conformance class Ontology shall contain ontology valid definitions and constraints meeting the requirements in Clause **6.3**.

* + 1. A feature ontology shall be fully compliant as a feature taxonomy.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req25/test

Test to ensure that the ontology is associated to a taxonomy consistent with the test suite for taxonomy

* + 1. A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/ontology/Req26/test

Test to ensure that constraints in the ontology can be tested and enforced in the feature instances. This included testing that the constraints in the ontology are not contradictory and thus preventing the creation of valid feature instances.

* 1. Test Suite for Schema   
     https://opengeospatial.org/as/FG/P1/FM/1.0/schema/

An implementation the Conformance class Schema shall contain schematic valid definitions and constraints meeting the requirements in Clause **6.4**

* + 1. A feature schema shall be fully compliant with a feature ontology.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req27/test

Test to ensure that the schema is associated to an ontology consistent with the test suite for ontology.

Note: Most schema languages (e.g. UML) are easily converted to ontology languages (e.g. OWL). Since most schema do not require definitions, a taxonomy may have to be created, unless it can be covered by **Per1.**

* + 1. A feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties and data types required by the definitions in the ontology.   
       https://opengeospatial.org/as/FG/P1/FM/1.0/schema/Req28/test

Test to ensure there is a complete and valid object model.

1. Example implementation
   1. Object implementation of taxonomy-controlled data

The discussion above implies that the most natural object language to use to implement taxonomic controlled feature data would have to have a dynamic schema/object model. Since most compiled languages (C++, C#) do not work quite that way, this design model assumes a non-dynamic schema by using inheritance and associations.

The classical object implementation of schema-controlled data uses a separate class for each feature type, with embedded attributes for simple properties and associations and roles. The alternative model in Figure B.1 breaks the feature classes into separate components allowing a statically structured language to be dynamic.



Figure B.1 – A Taxonomy-based Instantiation in Classical OOPL

* 1. Relational implementation of taxonomy-controlled data

In a strong feature schema, where all feature classes with their properties and associations are fully defined, as in the original "Simple Features for SQL" model, can used a single table for each feature class. Extending SQL for geographic information only requires the definition of a SQL-datatype for geometries, see ISO/IEC 13249-3.

If we loosen schema constraints, properties will have to be split off into separate tables to be shared by different feature types through foreign key links to these property tables.

To define the idea, the table design below assumes the other extreme where each property and association role can be associated to any feature type, i.e. where only the feature and property taxonomies are used, and no additional constraints are given. Just to be consistent, the taxonomy in the example is presented as a set of relation tables.

Table B.1 — Taxonomy Instantiation in a Classical RDBMS

|  |
| --- |
| Feature Taxonomy |
| featureType: ScopedName {id} ‼ primary key |
| featureDefinition: CharacterString ‼ text definition |
| Feature Inheritance |
| superType: ScopedName ‼ foreign key to Feature Taxonomy |
| subType: ScopedName ‼ foreign key to Feature Taxonomy |
| Association Taxonomy |
| associationType: ScopedName {id} ‼ primary key |
| associationDefinition: CharacterString ‼ text definition |
| Association Inheritance |
| superType: ScopedName ‼ foreign key to Association Taxonomy |
| subType: ScopedName ‼ foreign key to Association Taxonomy |
| Role Taxonomy |
| roleType: ScopedName ‼ primary key |
| roleDefinition: CharacterString ‼ text definition |
| association: ScopedName ‼ foreign key to Association Taxonomy |
| Property Taxonomy |
| propertyType: ScopedName ‼ primary key |
| propertyDefinition: CharacterString ‼ text definition |
| propertyValueType: ScopedName ‼ type of property value (usually datatype) |
| Property Inheritance |
| superType: ScopedName ‼ foreign key to Property Taxonomy |
| subType: ScopedName ‼ foreign key to Property Taxonomy |

Table B.2 — Taxonomy-base RDBMS feature implementation

|  |
| --- |
| Feature Table |
| featureType: ScopedName ‼ foreign key to feature taxonomy |
| featureID: ScopedName {id} ‼ primary key |
| required property: propertyType |
| Property Table for propertyID {one for each property in property taxonomy} |
| featureID: ScopedName ‼ foreign key to property taxonomy |
| value: propertyID.valueType ‼ type derived from property taxonomy |
| Association Table, columns ‼one for each role in the Role Taxonomy |
| ‼linked to the association} |
| ‼For each role, roleID where association is the same associationType. |
| featureID: ScopedName ‼ foreign key to feature taxonomy |
| … |
| featureID: ScopedName ‼ foreign key to feature taxonomy |

1. Bibliography

Additional information and definitions were taken from the following sources as indicated, with some edits for clarity or brevity in the special context of this International Standard. Normative References are in Clause 2.2.

|  |
| --- |
| 1. Andritsos, Periklis, and Keilty, Patrick. "Level-Wise Exploration of Linked and Big Data Guided by Controlled Vocabularies and Folksonomies". Advances in Classification Research 24 (2012). https://doi.org/10.7152/acro.v24i1.14670. |
| 1. ANSI/NISO (R2010. "Guidelines for the Construction, Format and Management of Monolingual Controlled Vocabularies". An American National Standard developed by the National Information Standards Organization, Baltimore, Maryland, USA. Approved July 5, 2006 by the American National Standards Institute, Reaffirmed May 13, 2010. https://www.niso.org/apps/group\_public/download.php/12591/z39-19-2005r2010.pdf. |
| 1. Butterfield, Andrew, Gerard Ekembe Ngondi, and Anne Kerr, eds. 2016. "A Dictionary of Computer Science". 7th Edition. Oxford, United Kingdom; New York, NY, United States of America: Oxford University Press. |
| 1. Chung, TongLee, Bin Xu, Peng Zhang, Yuanhua Tan, Ping Zhu, and Adeli Wubulihasimu. "Constructing City Ontology from Expert for Smart City Management". Semantic Technology, 187–94. Lecture Notes in Computer Science. Springer, Cham, 2013. doi:10.1007/978-3-319-06826-8\_15. |
| 1. Clapham, Christopher, "Concise Dictionary of Mathematics", Second Edition, Oxford University Press, 1996 |
| 1. Digital Geographic Information Exchange Standard (DIGEST) Part 4 FEATURE and ATTRIBUTE CODING CATALOGUE (FACC), 2000. https://www.dgiwg.org/digest/html/DIGEST\_2-1\_Part4.pdf. |
| 1. Fonseca, F., M. Egenhofer, et al. (2002). “Semantic Granularity in Ontology-Driven Geographic Information Systems.” AMAI Annals of Mathematics and Artificial Intelligence - Special Issue on Spatial and Temporal Granularity: 36.1-2 (2002): 121-51. Semantic |
| 1. Frank, Andrew U. "Tiers of Ontology and Consistency Constraints in Geographical Information Systems". International Journal of Geographical Information Science, IJGIS Special Issue (EURESCO Conference), 15, no. 7 (2001): 667–78. |
| 1. Ghawi, Raji, and Nadine Cullot. "Database-to-Ontology Mapping Generation for Semantic Interoperability". in Third International Workshop on Database Interoperability (InterDB 2007), in Conjunction with VLDB 2007, 2007. |
| 1. Golden Gate Bridge. Accessed December 18, 2017. http://goldengatebridge.org/research/factsGGBIntOrngColor.php. |
| 1. Guttman, A. (1984). "R-Trees: A Dynamic Index Structure for Spatial Searching", Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data – SIGMOD '84. p. 47. doi:10.1145/602259.602266 |
| 1. Harpring, Patricia. Introduction to Controlled Vocabularies: Terminology for Art, Architecture, and Other Cultural Works. Getty Publications, 2010. |
| 1. Hedden, Heather. "The Accidental Taxonomist". 2nd edition. Information Today, Inc., 2016. |
| 1. Introduction to Controlled Vocabularies. Getty Research Institute. Accessed October 19, 2017. https://www.getty.edu/research/publications/electronic\_publications/intro\_controlled\_vocab/. |
| 1. Kashyap, Vipul, Christoph Bussler, and Matthew Moran. The Semantic Web: Semantics for Data and Services on the Web. 2008 edition. Berlin: Springer, 2008. |
| 1. OECD Glossary of Statistical Terms, "Controlled Vocabulary Definition". Accessed September 19, 2017. <https://stats.oecd.org/glossary/detail.asp?ID=6260>. |
| 1. [The Common Lisp Foundation](http://cl-foundation.org/). "Welcome to Common-Lisp.net!" Accessed August 9, 2017. <https://common-lisp.net/>. |
| 1. Topological and Thematic Layering of Geological Map Information. Accessed October 27, 2017. https://pubs.usgs.gov/of/1997/of97-269/viljoen.html. |
| 1. Uschold, Michael. "Ontologies and Database Schema: What’s the Difference?" Accessed August 9, 2017. https://pdfs.semanticscholar.org/b44f/a4592b69183c1965d0075dea1a3bc58dfbfe.pdf. |
| 1. Using Oracle Data Cloud. Oracle Help Center. Accessed November 2, 2017. <https://docs.oracle.com/en/cloud/saas/data-cloud/dsmkt/preface.html> and <https://docs.oracle.com/en/cloud/saas/data-cloud/dsmkt/managing-your-taxonomy.html>. |

1. URLs inserted into the text are identifiers for sections (which will end in "/"), or paragraphs (which will not). [↑](#footnote-ref-1)
2. This implies that constraints in a formal ontology should be held submissive to reality, so that a single valid counter-example for a constraint, should imply that the ontology is in question, not the reality. [↑](#footnote-ref-2)
3. In the case of this specification, the domain knowledge is mainly in the taxonomy and ontology associate to the data. In geographic information the domain is the "feature representing a real-world phenomenon" See 4.6. [↑](#footnote-ref-3)
4. The GIS definition used "real-world phenomena" and Simula used "real world objects". This is an extension needed for GIS since many "features" displayed in data sets are "conceptual" like political boundaries, buffer zones, voting districts and others whose existing in the real world has real world implications, but is not always a real-world physical but a conceptual "object". [↑](#footnote-ref-4)
5. A military jargon for the "entire spectrum". [↑](#footnote-ref-5)