WIKIDATA

Wikidata:Data model

> **This is an information page.**
> It is not one of Wikidata's policies or guidelines, but rather intends to describe some aspect(s) of Wikidata's norms, customs, technicalities, or practices. It may reflect varying levels of consensus and vetting.

Wikidata represents entities (https://hub.toolforge.org/Q35120?lang=en) as **data items** (e.g. Tim Berners-Lee (Q80) and CERN (Q42944) are data items). Knowledge about data items is represented via **statements**, whose basic structure consists of a *subject*, a *predicate* and an *object*. For example, **Tim Berners-Lee (Q80)** employer (P108) CERN (Q42944) .

- The **subject** of a statement is usually a data item — in this case, Tim Berners-Lee (Q80).
- The **predicate** of a statement is always a property — in this case, employer (P108).
- The **object** of a statement is a value of the *data type* of the property — in this case, an item, CERN (Q42944).

The property used in a statement determines both the meaning of the statement (i.e. the nature of the relationship between the subject and the object), as well as which values may be used, as specified by its **data type**.

For example, in the example above we used the property employer (P108), whose values must have the data type Item, allowing a data item to be set as the object of the statement (in the case of our example, CERN (Q42944)).

An example of a property with a different data type is start time (P580), whose values must be of data type Point in time, so it can only be used to state a point in time.

Wikidata also allows statements to be qualified with further properties, which are called **qualifiers**. For example, we might state **Tim Berners-Lee (Q80)** employer (P108) CERN (Q42944) start time (P580) June 1980 end time (P582) December 1980 .

The information on this page is not required to contribute to Wikidata or to consume Wikidata. To learn about contributing/consuming Wikidata, please refer to the pages Wikidata:Introduction and Wikidata:Data access respectively.

Three levels of data models

Wikidata is powered by the Wikibase software. While Wikibase defines 12 data types by default, it does not come with any property out of the box. Wikidata, however, has 13,158 properties, which have all been created specifically for Wikidata and are defined within Wikidata itself. (Don't worry about

that large number, 75% of these properties are just external identifiers, i.e. links to items in other databases.)

When we speak of a "data model (https://hub.toolforge.org/Q1172480?lang=en)" in the context of Wikidata, it can actually refer to one of three things:

- the data model of the Wikibase software (which is actually more elaborate than just semantic triples[1])
- the fundamental data model that Wikidata establishes on top of the Wikibase model, which includes the core properties such as instance of (P31), subclass of (P279) and subproperty of (P1647)
- any of the topic-specific data models (e.g. for instances of television series (Q5398426), there are the properties number of episodes (P1113) and number of seasons (P2437))

All of these different data models are described on different pages:

- the data model of the Wikibase software is described on *mediawiki.org* very technically in the specification and more accessibly in the primer to the Wikibase data model
- the fundamental data model of Wikidata is not strictly defined, nonetheless this page attempts to describe it
- the various topic-specific data models are loosely described via properties for this type (P1963) and more formally via entity schemas.

Note that Wikidata has no central authority that decides how data should be modeled, instead that question is decided collaboratively by the community through public discussion. The data model of Wikidata has evolved over time and is very much still evolving: new data types can be introduced, new properties are being proposed and created, problematic properties get deprecated and there is an ongoing effort to better describe how properties are meant to be used via property constraints and entity schemas.

Data model of Wikibase

The data model of Wikidata is based on the data model of Wikibase, which is described very technically in the specification and more accessibly in the primer to the Wikibase data model.

Wikidata extends the Wikibase data model via extensions. Most notably WikibaseLexeme adds three entity types for lexicographical data (*Lexeme*, *Form* and *Sense*), as described in the WikibaseLexeme data model. Wikidata uses several extensions to add more data types to Wikibase, as described in Wikidata:Data model#Data types.

Data types

The data types of Wikidata are described at Help:Data type and listed at Special:ListDatatypes. Wikidata extends the data types of Wikibase via the following three extensions:

- WikibaseLexeme adds the Lexeme, Sense and Form data types to refer to its introduced entity types
- Math adds the Mathematical expression data type

- Score adds the Musical Notation data type

This is possible because the data types of Wikibase are extensible. The introduction of more data types can be proposed on Phabricator (https://phabricator.wikimedia.org/tag/datatypes/).

The Wikibase data model has a canonical representation in JSON (https://doc.wikimedia.org/Wikibase/master/php/docs_topics_json.html), which is further described at Wikidata:JSON format.

Note that several data types have limitations, which are listed at Help:Data type.

Also note that there is no clear semantical difference between String and External identifier ... several string properties are external identifiers and formatter URL (P1630) works for both.

Built-in data types

Data type	Number of properties
External identifier	9,932
Item	1,722
Quantity	682
String	348
URL	118
Commons media file	89
Point in time	70
Monolingual text	64
Property	22
Geographic coordinates	10
Tabular data	6
Geographic shape	3

Extra data types

Data type	Number of properties
Mathematical expression	36
Sense	19
Lexeme	15
Form	10
Musical Notation	6

Ranks

Every statement in Wikibase has one of three ranks (*normal*, *deprecated* or *preferred*). For the semantics of these ranks please refer to Help:Ranking#Usage.

No value and unknown value

See also: Help:Statements#Unknown or no values

- **S** P *no value* means that no such value exists (≡ ¬∃ X (**S** P X))
- **S** P *unknown value* can mean any of the following:
  - the value was once known but has been lost to time (e.g. **Paolo Baronni (Q7132144)** date of birth (P569) *unknown value*)
  - the exact value has never been known and might not ever be known (e.g. **star (Q523)** quantity (P1114) *unknown value*)
  - the Wikidata contributor who made the statement knows the value exists but doesn't know it personally
  - the value is a known object, but there's no Wikidata item about the object (perhaps because it's not notable).

Order of values

While Wikibase always stores values in a specific order (insertion order by default), the order of values generally does not imply any semantics. Semantic order is instead expressed via qualifiers, for example:

- series ordinal (P1545) to qualify the order of has part(s) (P527) values, e.g. **United States Constitution (Q11698)** has part(s) (P527) Article One of the United States Constitution (Q48416)

series ordinal (P1545) 1 , or
- time-based qualifiers like publication date (P577) to qualify software version identifier (P348) values

Note that the order expressed via qualifiers does not necessarily match the order of values in the user interface or the API because these interfaces simply return values in the serialization order, which may or may not match the semantic order expressed by the qualifiers.[2]

Fundamental entities

The fundamental properties of Wikidata are described in

1. Fundamental properties.

For more information and people interested in the ontology (https://hub.toolforge.org/Q324254?lang=en) of Wikidata, please refer to the Ontology WikiProject.

Fundamental properties

Note: This section assumes that you are familiar with logical operators (https://hub.toolforge.org/Q211790?lang=en), for a less technical explanation please refer to Help:Basic membership properties.

The three arguably most important properties of Wikidata are based on RDF Schema (https://hub.toolforge.org/Q1751819?lang=en), which is described in the RDF Schema specification (https://www.w3.org/TR/rdf-schema/).

- instance of (P31) is equivalent to rdf:type (https://www.w3.org/TR/rdf-schema/#ch_type)
- subclass of (P279) is equivalent to rdfs:subClassOf (https://www.w3.org/TR/rdf-schema/#ch_subclassof)
- subproperty of (P1647) is equivalent to rdfs:subPropertyOf (https://www.w3.org/TR/rdf-schema/#ch_subpropertyof)

These properties have the following semantics:

- **A** instance of (P31) B ∧ **B** subclass of (P279) C ⇒ **A** instance of (P31) C
- $\mathbf{P_1}$ subproperty of (P1647) P_2 ∧ **A** P_1 B ⇒ **A** P_2 B

Please note that subclass of (P279) and subproperty of (P1647) are both transitive properties:

- **P** instance of (P31) transitive Wikidata property (Q18647515) ∧ **A** P B ∧ **B** P C ⇒ **A** P C

> Wikidata has subproperties of instance of (P31) and subclass of (P279).
>
> So don't forget to take that into account when consuming data from Wikidata. See subproperties of *instance of* (https://query.wikidata.org/embed.html#SELECT%20%3Fproperty%20%3FpropertyLabel%20%3FpropertyDescription%20WHERE%20%7B%0A%20%20%20%20%3Fproperty%20wdt%3AP1647%20wd%3AP31%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5

D%2Cen%22.%20%7D%0A%7D) and subproperties of *subclass of* (https://query.wikidata.org/embed.html#SELECT%20%3Fproperty%20%3FpropertyLabel%20%3FpropertyDescription%20WHERE%20%7B%0A%20%20%20%20%3Fproperty%20wdt%3AP1647%20wd%3AP279%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D).

Another important property is inverse property (P1696), which is equivalent to owl:inverseOf (https://www.w3.org/TR/owl-ref/#inverseOf-def) and carries the following semantics:

- $\mathbf{P_1}$ inverse property (P1696) P_2 ∧ **A** P_1 B ⇒ **B** P_2 A

Restrictiveness of qualifiers

Qualifiers can be either restrictive or non-restrictive. Restrictive qualifiers change the meaning or scope of a statement, they have to be taken into account by data consumers that want to correctly interpret Wikidata statements. Non-restrictive qualifiers on the other hand just add additional information that can be safely disregarded without changing the meaning or scope of the statement.

Examples for restrictive qualifiers are:

- qualifiers that restrict where a statement applies (e.g. applies to jurisdiction (P1001) and valid in place (P3005))
- qualifiers that restrict when a statement applies (https://query.wikidata.org/embed.html#SELECT%20%3Fproperty%20%3FpropertyLabel%20%3FpropertyDescription%20WHERE%20%7B%0A%20%20%20%20%3Fproperty%20wdt%3AP31%20wd%3AQ115429021%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D) (e.g. start time (P580) and end time (P582))
- qualifiers that limit how universally a statement applies (e.g. nature of statement (P5102) sometimes (Q110143752))

The restrictiveness of properties when used as a qualifier is currently modeled via instance of (P31) restrictive qualifier (Q61719275) and instance of (P31) non-restrictive qualifier (Q61719274) (note that you as always have to take the transitivity of instance of (P31) into account).

Unfortunately some properties aren't clear-cut and can be both restrictive as well as non-restrictive when used as a qualifier, so we can group qualifier properties into four categories:

- properties that are clearly restrictive (https://query.wikidata.org/embed.html#SELECT%20%3Fprop%20%3FpropLabel%20%3FpropDescription%20WHERE%20%7B%0A%20%20%20%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719275%20FILTER%20NOT%20EXISTS%20%7B%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719274%20%7D%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D) when used as a qualifier
- properties that are clearly non-restrictive (https://query.wikidata.org/embed.html#SELECT%20%3Fprop%20%3FpropLabel%20%3FpropDescription%20WHERE%20%7B%0A%20%20%20%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719274%20FILTER%20NOT%20

EXISTS%20%7B%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719275%20%7D%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D) when used as a qualifier
- properties that can be both restrictive and non-restrictive (https://query.wikidata.org/embed.html#SELECT%20%3Fprop%20%3FpropLabel%20%3FpropDescription%20WHERE%20%7B%0A%20%20%20%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719274%20.%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719275%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D) when used as a qualifier
- properties that have not been classified at all regarding their restrictiveness (https://query.wikidata.org/embed.html#SELECT%20DISTINCT%20%3Fprop%20%3FpropLabel%20%3FpropDescription%20WHERE%20%7B%0A%20%20%20%20%7B%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ15720608%20%7D%0A%20%20%20%20UNION%20%7B%20%3Fprop%20p%3AP2302%20%5B%20ps%3AP2302%20wd%3AQ53869507%3B%20pq%3AP5314%20wd%3AQ54828449%20%5D%20%7D%0A%20%20%20%20FILTER%20NOT%20EXISTS%20%7B%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719274%20%7D%0A%20%20%20%20FILTER%20NOT%20EXISTS%20%7B%20%3Fprop%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ61719275%20%7D%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D) when used as a qualifier

Negation

Wikibase does not have built-in support for negation (https://hub.toolforge.org/Q190558?lang=en), negation therefore has to be modeled with separate properties. For example has part(s) (P527) can be negated with does not have part (P3113). Such negating properties only exist for a few properties (https://query.wikidata.org/embed.html#SELECT%20%3Fprop%20%3FpropLabel%20%3Fother%20%3FotherLabel%20WHERE%20%7B%0A%20%20%20%20%3Fprop%20wdt%3AP11317%20%3Fother%20FILTER%20NOT%20EXISTS%20%7B%20%3Fprop%20wdt%3AP31%20wd%3AQ115449020%20%7D%0A%20%20%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%7D). When the need for a new negating property arises, it may be proposed.

The semantics of negating properties are modeled via negates property (P11317), as follows:

- **P1** negates property (P11317) P2 ∧ **A** P1 B ⇒ ¬∃ **A** P2 B if both statements have none or the same restrictive qualifiers.
- **P1** negates property (P11317) P2 ⇒ **P2** negates property (P11317) P1 (negates property (P11317) is a symmetric property)

Whether or not a property expresses the absence of something is currently modeled via instance of (P31) Wikidata property to express the absence of something (Q115449020) .

Differences from OOP

Contrary to object-oriented programming (https://hub.toolforge.org/Q79872?lang=en) there is nothing preventing an entity from being both an instance as well as a class.

Furthermore an entity can be an instance of multiple classes, as well as a subclass of multiple classes.

Lastly you might expect that an instance automatically inherits all statements from its parent classes, however that is explicitly not the case, as explained in Wikidata:Data model#Inheritance.

Inferring classes

Properties may specify class of non-item property value (P10726) which has the semantics:

- **P** class of non-item property value (P10726) C ∧ **A** P B ⇒ **B** instance of (P31) C

Classes can be defined to be a union (https://hub.toolforge.org/Q185359?lang=en) or a disjoint union (https://hub.toolforge.org/Q842620?lang=en) of other classes with union of (P2737) and disjoint union of (P2738) respectively. Their concrete semantics are as follows:

Let's define $\textbf{classesOf}(X) := \{C \mid X \text{ is an instance of } C\}$.

Classes may specify union of (P2737) which has the semantics:

- **C** union of (P2737) list of values as qualifiers (Q23766486) list item (P11260) S_1 list item (P11260) S_2 list item (P11260) $S_{...}$ list item (P11260) S_N ∧ **X** instance of (P31) C ⇒ classesOf(X) ∩ {S_1, S_2, ..., S_N}| ≠ {}

Classes may specify disjoint union of (P2738) which has the semantics:

- **C** disjoint union of (P2738) list of values as qualifiers (Q23766486) list item (P11260) S_1 list item (P11260) S_2 list item (P11260) $S_{...}$ list item (P11260) S_N ∧ **X** instance of (P31) C ⇒ |classesOf(X) ∩ {S_1, S_2, ..., S_N}| = 1

Inheritance

If you are familiar with object-oriented programming (https://hub.toolforge.org/Q79872?lang=en), you might expect that instances of a class inherit the statements of a class. This is generally not the case. For example just because **horse (Q726)** studied by (P2579) hippology (Q1157006) and **Apology (Q4780432)** instance of (P31) horse (Q726) does not mean that **Apology (Q4780432)** studied by (P2579) hippology (Q1157006) . However there are some properties that are likely to be inherited:

Property	Inverse property
has part(s) (P527)	part of (P361)
has characteristic (P1552)	*none*
has cause (P828)	has effect (P1542)
uses (P2283)	used by (P1535)

For example **public website (Q115449506)** part of (P361) World Wide Web (Q466) and **YouTube (Q866)** instance of (P31) public website (Q115449506) can be used to correctly infer **YouTube (Q866)** part of (P361) World Wide Web (Q466) .

When attempting to make such inferences don't forget to take ranks, restrictive qualifiers and negation into account, as explained in Wikidata:Data model#Does a statement apply?.

Does a statement apply?

The following is an attempt at outlining a strategy to decide whether a particular statement applies to a given entity:

1. Statements ranked as deprecated have been superseded and therefore no longer apply.
2. Statements with a restrictive qualifier only apply with regards to the respective qualifier.
3. Statements of certain properties are likely to be inherited (see inheritance). Note however that instances or intermediary classes may negate statements inherited from a parent class, as described in negation.

Reflexive statements

A P A has unclear semantics if A is a class, it could mean:

1. an instance of A has a relation P to another instance of A (which may or may not be the same instance)
2. an instance of A has a relation P to a different instance of A (which cannot be the same instance)
3. an instance of A has a relation P to itself

See object is for a proposal to introduce a qualifier property to differentiate these cases.

Format string properties

Wikidata has several format string properties, such as formatter URL (P1630), DOI formatter (P8404) and URN formatter (P7470).

The formatting mechanism of these properties and what kind of values they produce is currently not stated in a machine-readable manner, however that might change with the introduction of the proposed format string properties.

Property constraints

Wikidata employs property constraints to combat property misuse. Property constraints are implemented by Extension:WikibaseQualityConstraints and are stated on properties via property constraint (P2302) since 2017. [3] The violation of such property constraints is directly displayed in the Wikidata user interface.

More complex property constraints can be implemented as SPARQL queries and placed with `{{Complex constraint}}` on property talk pages. The violation of such complex constraints is periodically reported by a bot on pages within the Category:Complex constraint violation reports category.

For more information about property constraints, please refer to the help portal and the property constraints WikiProject.

Topic-specific data models

Wikidata covers many topics, such as art, biology, countries, cities, monuments, movies, people, software, websites, writings, etc. All entities of these topics that are notable somehow need to be represented as data items with statements. So which statements should be made for a specific entity type and which properties should be used for these statements? The answers to these questions are subject to the topic-specific data model that should be used for the specific topic. So, which data model should be used for a given topic? That is decided collaboratively by the Wikidata community through public discussion. The discussions and efforts about a specific topic in Wikidata are organized via WikiProjects.

Where can you find topic-specific data models?

- properties for this type (P1963) statements on data items
  For example **television series (Q5398426)** properties for this type (P1963) number of episodes (P1113) expresses that instances of television series (Q5398426) usually have a statement with the number of episodes (P1113) property.
- all pages about *data model(s)* (sometimes called *data structure* or *item structure* or simply *model(s)*) can be found in this category and in this template. [4]
- subpages of WikiProject pages (https://www.wikidata.org/w/index.php?title=Special:Search&search=intitle%3A%2F(data%20models%3F%7C\%2Fproperties)%2Fi&ns4=1&prefix=Wikidata%3AWikiProject), e.g. Wikidata:WikiProject Movies/Properties
- entity schemas can be found in the EntitySchema directory, e.g. E17

Entity schemas

An alternative approach to property constraints is using the Shape Expressions (https://hub.toolforge.org/Q29377880?lang=en) data modelling language. For Wikidata such schemas can be stored within the `EntitySchema:*` namespace on the wikidata.org wiki (which is enabled by the EntitySchema MediaWiki extension). Note that the effort to establish such schemas for Wikidata is very much ongoing: the Shape Expression for class property proposal is currently on hold because the EntitySchema data type is not yet implemented.[5]

For more information about Wikidata Schemas, please refer to the Schemas WikiProject.

See also

- Wikidata in Wikidata
- Wikidata:Data access
- Wikidata:Coverage

References

1. Items can have labels, descriptions, aliases and sitelinks, statements have a rank and can have qualifiers and references, and values can also be specified as *no value* or *unknown value*.
2. Phabricator task T173432: Sort claims of a property in meaningful way
3. Phabricator task T102759: Migrate constraints from property talk pages to statements on properties
4. it is possible that such variety will be standardized in the future
5. Phabricator task T214884: linking Schemas in statements