Overview of other changes

An Introduction to RDF and SPARQL 1.2
ISWC 2025
2 November 2025

Pierre-Antoine Champin
Enrico Franconi
Ora Lassila
Ruben Taelman

https://www.w3.org/Talks/2025/iswc-tutorial-rdfsparql-12/

Outline

- Version announcement
- Base directions
- EXISTS
- Symmetric RDF (non normative)
- Other changes

Outline

- **Version announcement**
- Base directions
- EXISTS
- Symmetric RDF (non normative)
- Other changes

Manage syntax changes without new media types

Example: The media type of Turtle 1.1 is `text/turtle`

Problem: Syntax changes in Turtle 1.2.

New media types not a scalable solution:

text/turtle-1-2, application/rdf+xml-1-2, text/ntriples-1-2, ...

text/turtle-1-3, application/rdf+xml-1-3, text/ntriples-1-3, ...

text/turtle-1-4, application/rdf+xml-1-4, text/ntriples-1-4, ...

...

=> Also painful for clients to manage during content negotiation

(+CORS: 128 char header value limit)

Keep media types and introduce versioning

Precedent: JSON-LD already does this (“`@version`” keyword), HTML

Two ways to announce versions:

In-band within the syntax

Out-of-band outside of the syntax (media type parameter)

In-band versioning

VERSION keyword for Turtle-like syntaxes (incl. SPARQL)

```
VERSION "1.2"
PREFIX : <http://example.com/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
:a :name "Alice" ~ :t {| :statedBy :bob |} .
```

Version keyword

Purely informative: keyword MAY be used when 1.2 features are present.

(could be impossible to determine for some producers)

Can appear anywhere, should be placed early to make unsupporting parsers fail early.

Out-of-band versioning

MAY be used instead of, or together with in-band versioning.

Request:

```
GET /document.ttl HTTP/1.1
Host: example.com
Accept: text/turtle; version=1.2
```

Media type parameter

Response:

```
HTTP/1.1 200 OK
Content-Type: text/turtle; version=1.2
```

Defined version labels

`"1.2"`	RDF 1.2 syntax
`"1.2-basic"`	RDF 1.2 syntax without triple terms
`"1.1"`	RDF 1.1 syntax except for use of a version directive

Server considerations wrt. versioning (non-normative)

What if requested version is incompatible with dataset/graph?

1.2 → 1.1/1.2-basic Apply basic encoding algorithm (remove triple terms)

(see https://w3c.github.io/rdf-interop/spec/)

Reject 406 “Not Acceptable”

Ignore Return native representation

“Be conservative in what you send, be liberal in what you accept.”

Client considerations wrt. versioning (non-normative)

No version mentioned? Clients can assume latest RDF version (1.2)

Conflicting versions? Do one of:

- Ignore version announcement
- Raise error and abort
- Issue warning and potentially ignore some triples
- Parse fully and only emit triples for declared version

Outline

- Version announcement
- **Base directions**
- EXISTS
- Symmetric RDF (non normative)
- Other changes

Languages can have different reading directions

Direction can not be derived from language:
https://www.w3.org/International/questions/qa-direction-from-language

Language tags are insufficient → tag literals with language AND base direction

Language-tagged strings can now be (optionally) augmented with a base direction

```
:a :name "Alice"@en--ltr
```

Base direction

New datatype: `rdf:dirLangString`

ltr Left-to-right

rtl Right-to-left

SPARQL functions to handle `rdf:dirLangString`

`LANG()`	Get language of a literal (EXTENDED)
`LANGDIR()`	Get base direction of a literal (NEW)
`hasLANG()`	If term is literal with language (EXTENDED)
`hasLANGDIR()`	If term is literal with base direction (NEW)
`STRLANGDIR()`	Make literal with value, language, and base direction (NEW)

Outline

- Version announcement
- Base directions
- **EXISTS**
- Symmetric RDF (non normative)
- Other changes

SPARQL 1.1 introduced the (NOT) EXISTS filter expressions

For testing if some data does (not) exist.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person WHERE {
    ?person rdf:type  foaf:Person .
    FILTER EXISTS { ?person foaf:name ?name }
}
```

Several issues were discovered with EXISTS

Outcome of the SPARQL EXISTS W3C community group:

1. Some uses of EXISTS are not defined during evaluation
2. Substitution happens where definitions are only for variables
3. Blank nodes substituted into BGPs act as variables
4. Substitution can flip MINUS to its disjoint-domain case
5. Substitution affects disconnected variables

Further reading:

- https://w3c.github.io/sparql-exists/docs/sparql-exists.html
- https://afs.github.io/substitute.html
- https://ceur-ws.org/Vol-1690/paper72.pdf
- https://docs.google.com/presentation/d/16U0Z0HR6MwNWdBe-ctm3wLFd6wiKCum7DFRaTpxkkdM/edit?slide=id.g27ccbd8ed27_0_201

Examples: 2. Substitution happens where definitions are only for variables

The target term of BIND must always be a variable.

```
SELECT ?person WHERE {
    ?person rdf:type  foaf:Person .
    FILTER EXISTS { BIND ( :p AS ?person ) }
}
```

Projection with SELECT must always be a variable.

```
SELECT ?person WHERE {
    ?person rdf:type  foaf:Person .
    FILTER EXISTS { SELECT ?person { :a :b ?person } }
}
```

→ Different interpretations lead to inconsistent behaviour across SPARQL engines.

Solution: working towards “deep injection”

Goal is to remain close to the original intent of EXIST in SPARQL 1.1

Deep injection Inject values inside the pattern (↔ at beginning of pattern)

Without projection Values projected out in sub-SELECTs are affected

Per solution mapping Evaluate EXISTS pattern once per solution mapping (↔ once for all solution mappings)

Work in progress!

Outline

- Version announcement
- Base directions
- EXISTS
- **Symmetric RDF (non normative)**
- Other changes

Reminder: allowed term types in RDF 1.2 triples

Subject	IRI, blank node
Predicate	IRI
Object	IRI, blank node, literal, triple term

In RDF 1.1, generalized RDF was introduced

Subject	IRI, blank node, literal
Predicate	IRI, blank node, literal
Object	IRI, blank node, literal

Can be useful for specific tools and proofs.

Non-normative: tools are not required to accept this, and may cause interop issues!

RDF 1.2 extends generalized RDF, and adds symmetric RDF

Generalized RDF triple

Subject	IRI, blank node, literal, triple term
Predicate	IRI, blank node, literal, triple term
Object	IRI, blank node, literal, triple term

Symmetric RDF triple

Subject	IRI, blank node, literal, triple term
Predicate	IRI
Object	IRI, blank node, literal, triple term

→ Useful for proofs around RDFS entailment.

Outline

- Version announcement
- Base directions
- EXISTS
- Symmetric RDF (non normative)
- **Other changes**

Removal of plain and simple literals concepts

SPARQL 1.1

Distinction between literals with datatype xsd:string and plain literals (no datatype).

→ Inherited from RDF 1.0 (1.1 was not out yet)

SPARQL 1.1 had its own different meaning of simple literal:

syntactic sugar for xsd:string literals

SPARQL 1.2

Removal of distinction in favor of RDF 1.1/1.2's simple literal.

Clearer algebraic syntax and algebra operators

Example in SPARQL 1.1: Filter evaluation semantics

```
eval(D(G), Filter(F, P)) = Filter(F, eval(D(G),P), D(G))
```

FILTER in syntax != Algebraic filter operator

Algebraic syntax is now explicitly defined

Algebraic syntax and algebra operators clearly distinguished now

Many new specification tests

https://github.com/w3c/rdf-tests/

Declarative set of test to measure conformance of a system

New tests for

- N-Triples 1.2 (70)
- N-Quads 1.2 (68)
- Semantics 1.2 (155)
- TriG 1.2 (60)
- Turtle 1.2 (92)
- RDF-XML 1.2 (30)
- SPARQL 1.2 (230)

Definition of *quad* (non-normative)

Quad: often used informally to refer to triple with graph, but never defined in RDF specs.

E.g. RDF/JS represents triples as quads in JavaScript.

> *A quad can be represented as a tuple composed of subject, predicate, object, and an optional graph name.*

Added `rdf:JSON` datatype

Was already used by JSON-LD 1.1.

Lexical space

set of all RDF strings that conform to the JSON Grammar

Value space

smallest set containing strings, numbers (xsd:double), finite unordered maps mapping strings to values in the value space, lists of values in the value space, and literal values (true, false, and null)

→ intuitively: all "JSON values"

Full list of changes within each spec

§ A. Changes between SPARQL 1.1 Query Language and SPARQL 1.2 Query Language

This section is non-normative.

- Normative changes:
  - Update grammar for triple terms, reifiers, reified triples, annotation syntax, and triple term functions in 19.8 Grammar
  - Add functions related to triple terms to 17.4.6 Functions on Triple Terms: `TRIPLE`, `isTRIPLE`, `SUBJECT`, `PREDICATE`, `OBJECT`
  - Update grammar for literal base direction syntax
  - Update grammar for VERSION declaration and a new section to describe its usage
  - Add functions related to language tag and base direction: `LANGDIR`, `hasLANG`, hasLANGDIR, and `STRLANGDIR`
  - Define parser input as being an RDF string. Exclude Unicode surrogates from Unicode escape sequences.
  - Remove concepts of plain and simple literals, in favor of explicit mentions of `xsd:string`
  - Migrate XML Schema references to 1.1
  - Update references to XPath from 2.0 to 3.1
  - Define `EBV` as a functional form.
- Editorial changes:
  - Give an actual function signature to 17.4.2.2 sameValue
  - Improve wording of blank nodes in 16.2.1 Templates with Blank Nodes
  - Improve display on mobile
  - Move `sameValue` (was `RDFterm-equal`) and `sameTerm` to 17.4.2 Functions on RDF Terms
  - Add note on deduplication of triples produced by CONSTRUCT to 16.2 CONSTRUCT
  - Remove historical notes on rdf:langString datatype from 17.4.2.12 DATATYPE

Next steps

After RDF and SPARQL 1.2 recommendations, WG in maintenance mode

→ Faster evolution of RDF and SPARQL specs

Will consider input from SPARQL DEV community group:
https://github.com/w3c/sparql-dev/