Live Linked Data

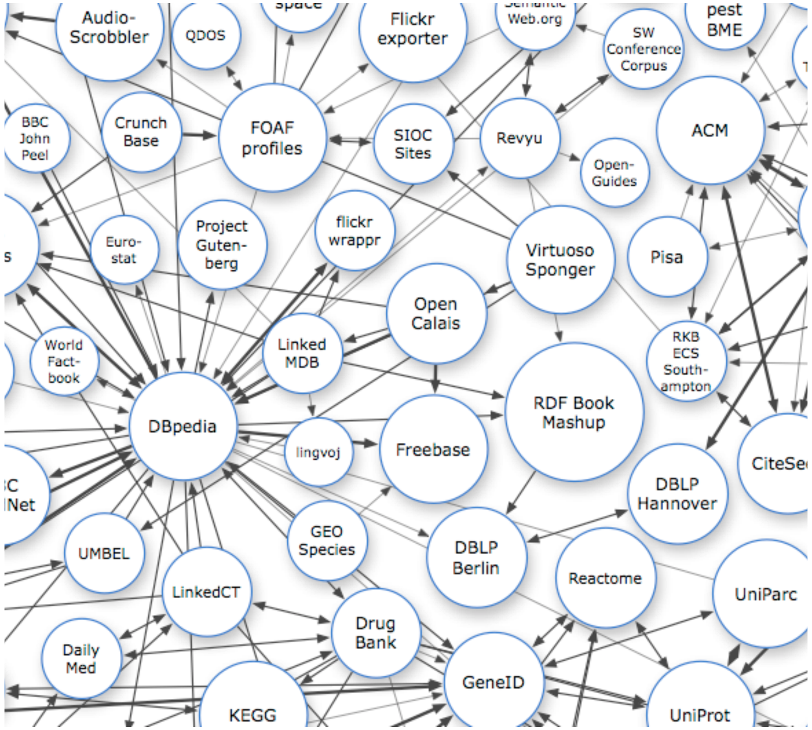
or how to make linked data writable with massive optimistic replication and Conflict-Free Replicated Data Types

Luis-Daniel Ibáñez, Nantes University, GDD team

Luis-Daniel Ibáñez, Hala Skaf-Molli, Pascal Molli, and Olivier Corby. 2012. Synchronizing semantic stores with commutative replicated data types. In Proceedings of the 21st international conference companion on World Wide Web (WWW '12 Companion). ACM, New York, NY, USA, 1091-1096. DOI=10.1145/2187980.2188246 http://doi.acm.org/10.1145/2187980.2188246

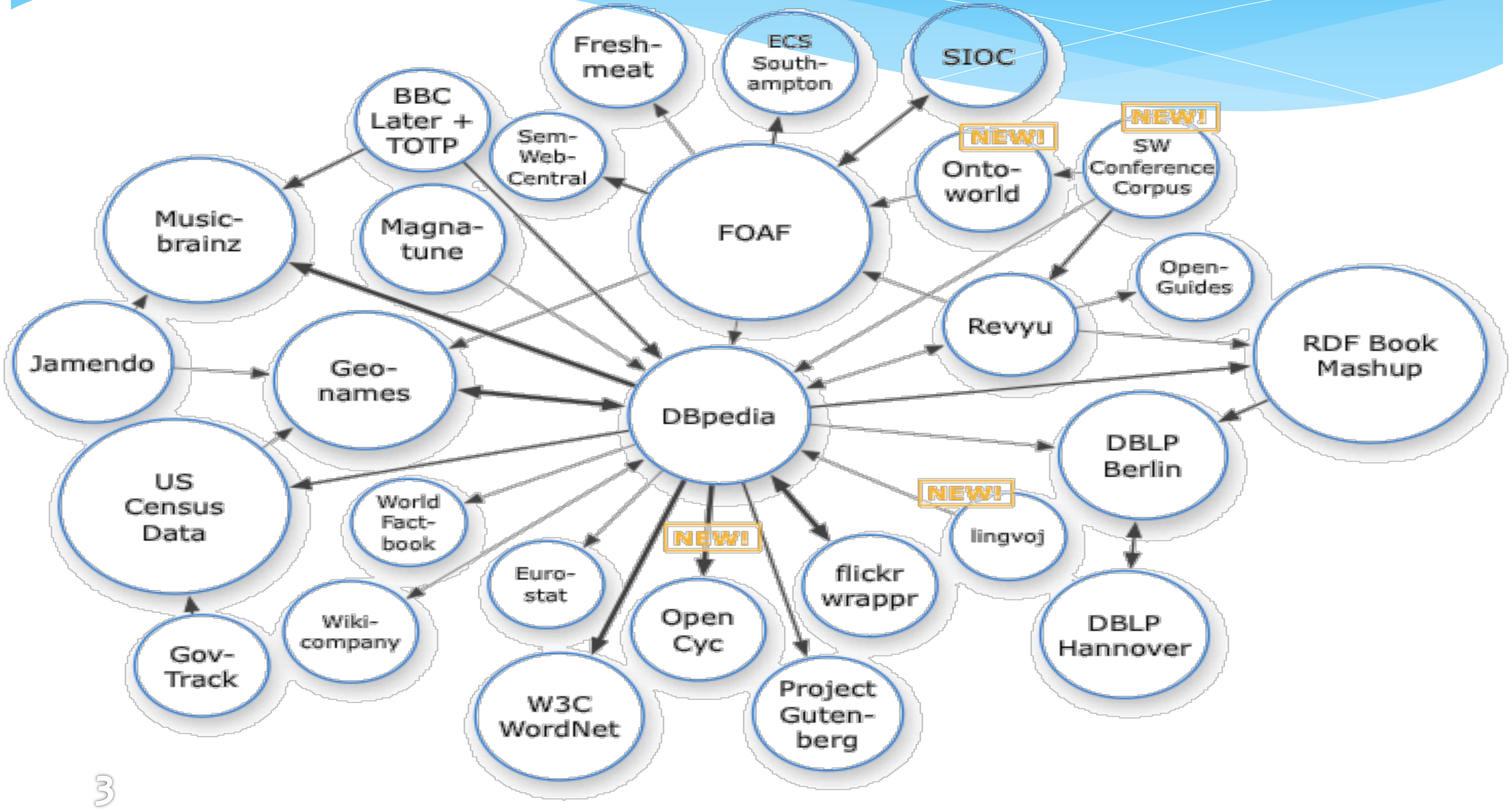
Linked Data

- **Autonomous participants** agree on a set of principles for publishing RDF data, allowing its querying and browsing across distributed servers.
- In 2011, More than 30 billion RDF triples distributed between ~700 **autonomous participants**[1].

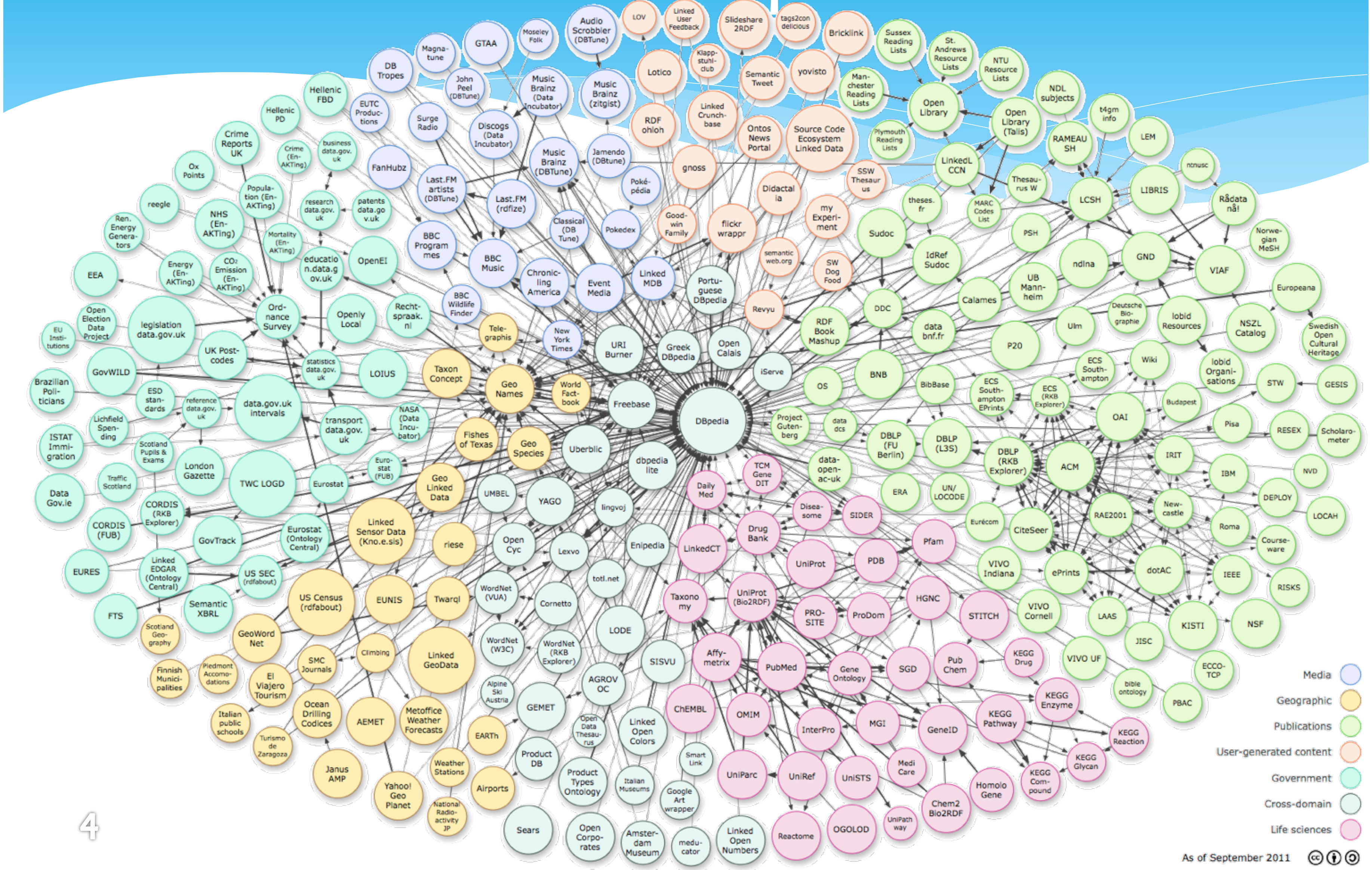


[1] Billion Triple Challenge + Linked Open Data Metadata
T. Käfer et. Al., Towards a Dynamic Linked Data Observatory, LDOW@WWW12

LOD - 2007 October



LOD - 2011 September



Here is a list of links created by LinkedMDB that relate the movie "The Shining" to other open datasets:

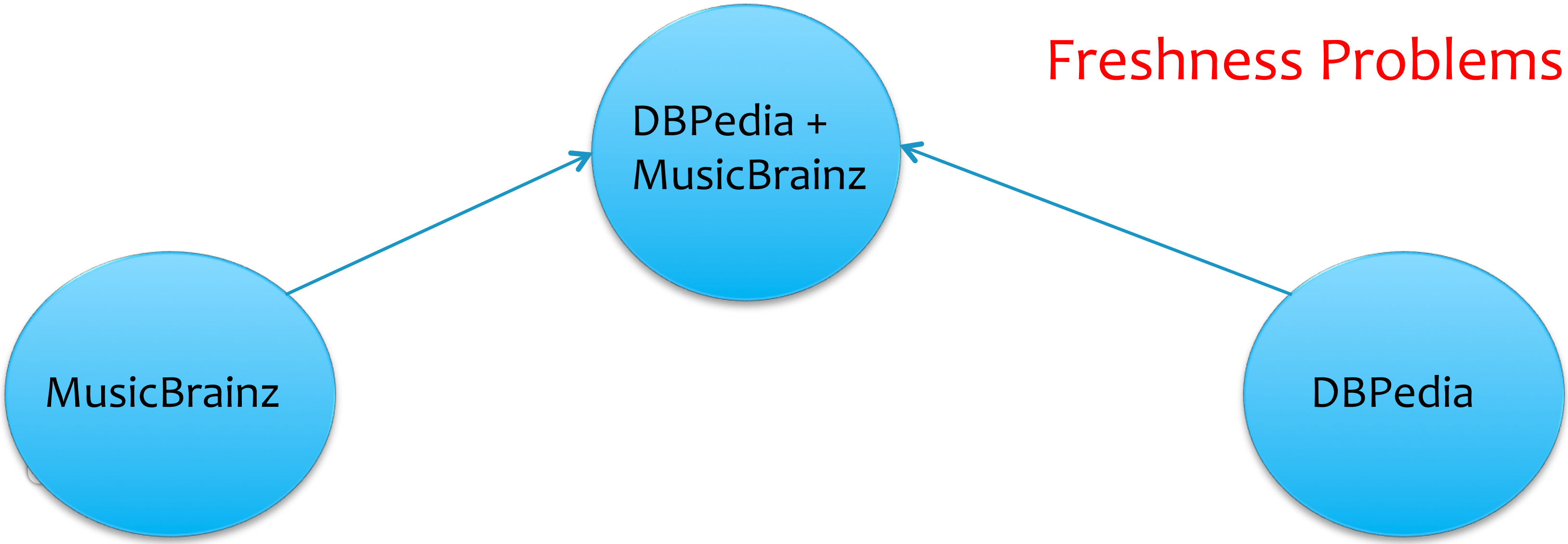
- owl:SameAs to "The_Shining_(film)" on DBpedia
- owl:SameAs to "The_Shining_(film)" YAGO
- dbpedia:hasPhotoCollection to The movie's photos on flickr™ wrappr
- movie:relatedBook to the movie's related book on RDF Book Mashup
- owl:SameAs from one of the music composers of the movie, Béla Bartók, toMusicBrainz: http://zitgist.com/music/artist/fd14da1b-3c2d-4cc8-9ca6-fc8c62ce6988
- rdfs:SeeAlso from one of the writers of the movie, Stanley Kubrick to RDF Book Mashup: http://www4.wiwiss.fu-berlin.de/bookmashup/persons/Stanley+Kubrick
- foaf:based_near to the locations of the movie in the US and GB: http://sws.geonames.org/2635167/ and http://sws.geonames.org/6252001/
- movie:language to the language of the movie on lingvoj.org: http://www.lingvoj.org/lingvo/en

Source : http://wiki.linkedmdb.org/Main/Interlinking

Queries in Linked Data: Local Copy and Query

What is the largest city of Vicente Fernández' origin country?

Select ?country ?city where
{http://musicbrainz.org/Vicente_Fernandez http://musicbrainz.org/fromCountry ?country.
?country http://dbpedia.org/ontology/largestCity ?city . }

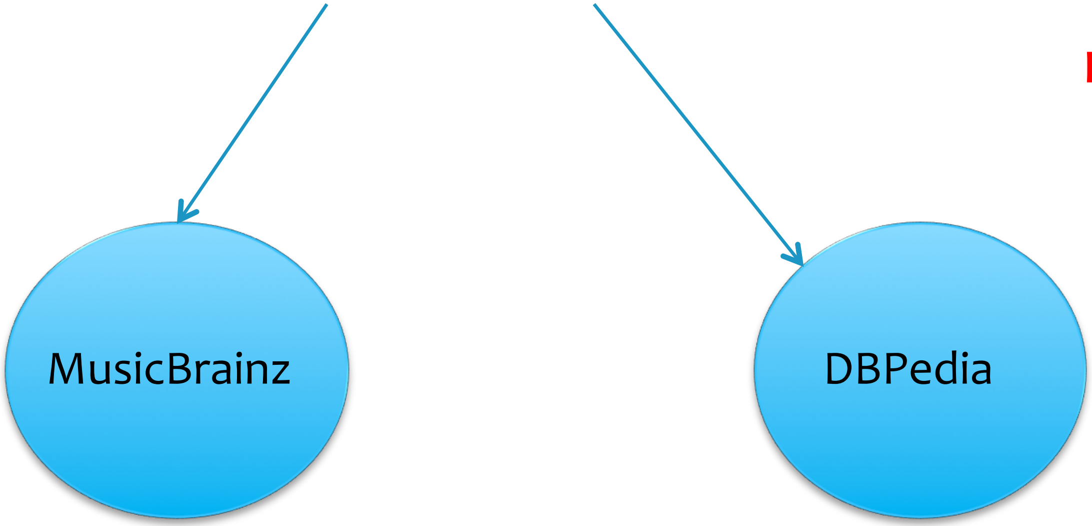


Freshness Problems…

Queries in Linked Data: Distributed Queries

What is the largest city of Vicente Fernández' origin country?

```
SELECT ?country ?city WHERE {
SERVICE <http://musicbrainz.org/> {Vicente_Fernandez fromCountry ?country. }
SERVICE http://dbpedia.org/ {?country largestCity ?city . } }
```

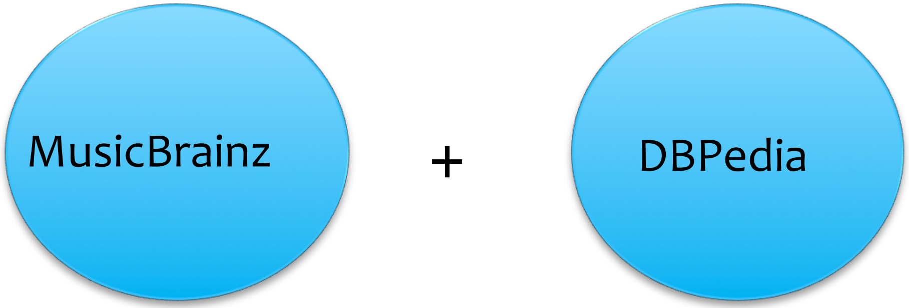


Endpoint reliability and performance Problems

Querying LOD: Sometimes OK

What is the largest city of Vicente Fernández' origin country?

```
Select ?country ?city where
{http://musicbrainz.org/Vicente_Fernandez http://musicbrainz.org/fromCountry ?country.
?country http://dbpedia.org/ontology/largestCity ?city . }
```



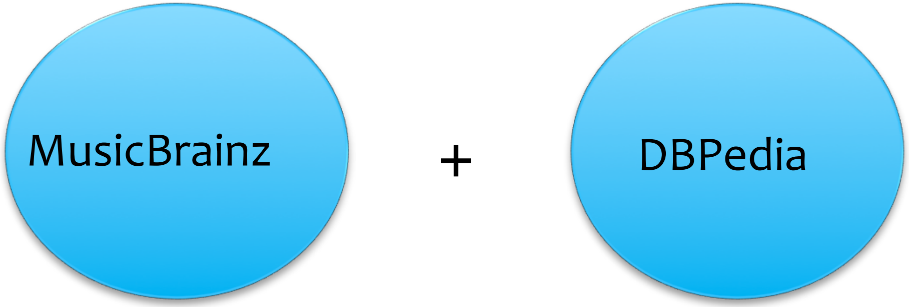
MusicBrainz + DBPedia

?country = Mexico
?city = Mexico City

Querying LOD: Sometimes KO

What is the largest city of Serge Gainsbourg’s origin country?

```
Select ?country ?city where
{http://musicbrainz.org/Serge_Gainsbourg http://musicbrainz.org/fromCountry ?country.
?country http://dbpedia.org/ontology/largestCity ?city . }
```



MusicBrainz + DBPedia

?country = France
?city = Prefectures In France

?????

Issues

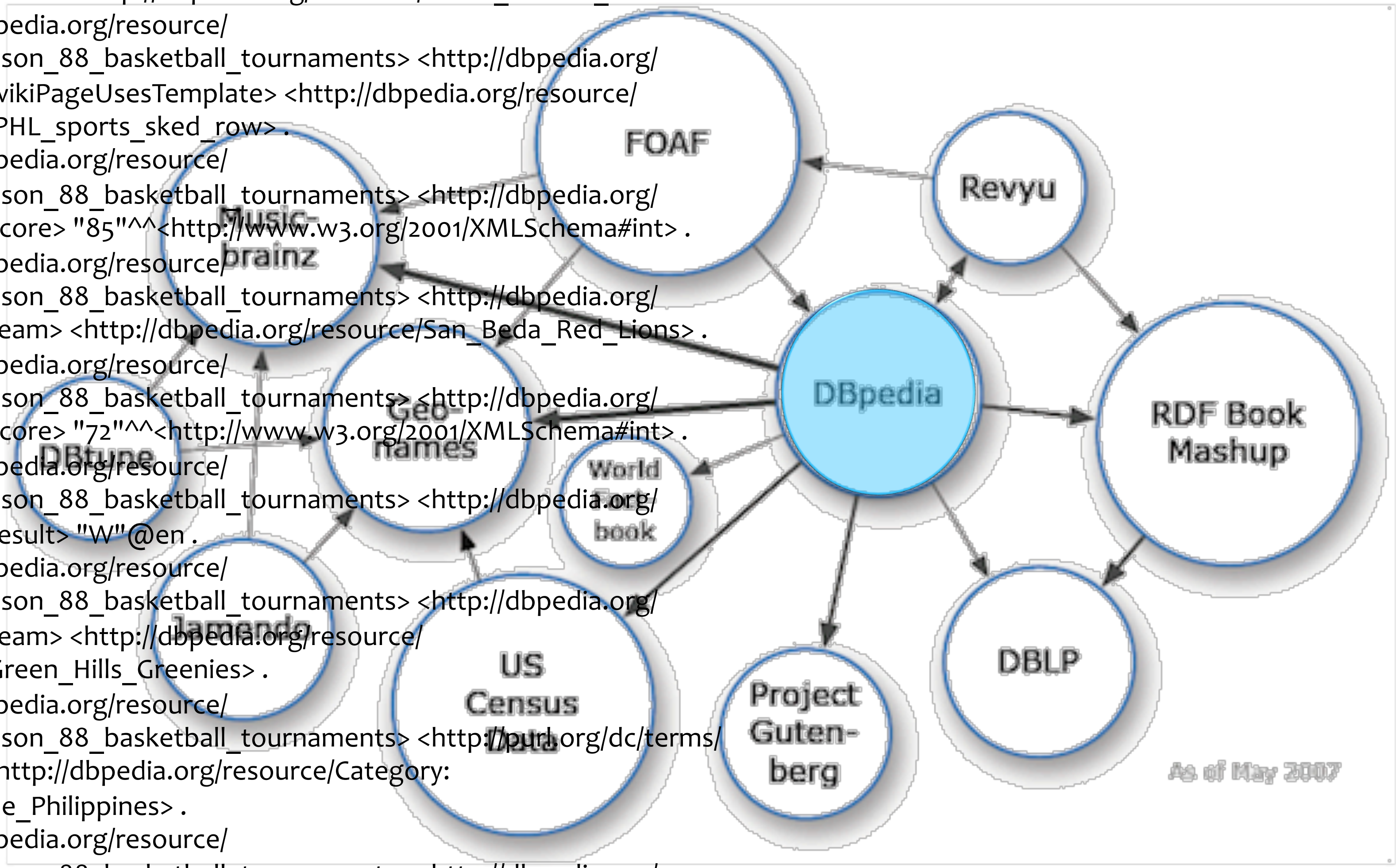
- Quality of data is poor, if I find an error, where to change and how can I change?
  - If accessing remote datasets: they are read-only (autonomous participants)
  - If modifying local copy: how to synchronize with original? How to publish changes?
- **How to make Linked Data Writable ? How to move from Linked Data 1.0 to Linked Data 2.0?**

Live Linked Data Approach

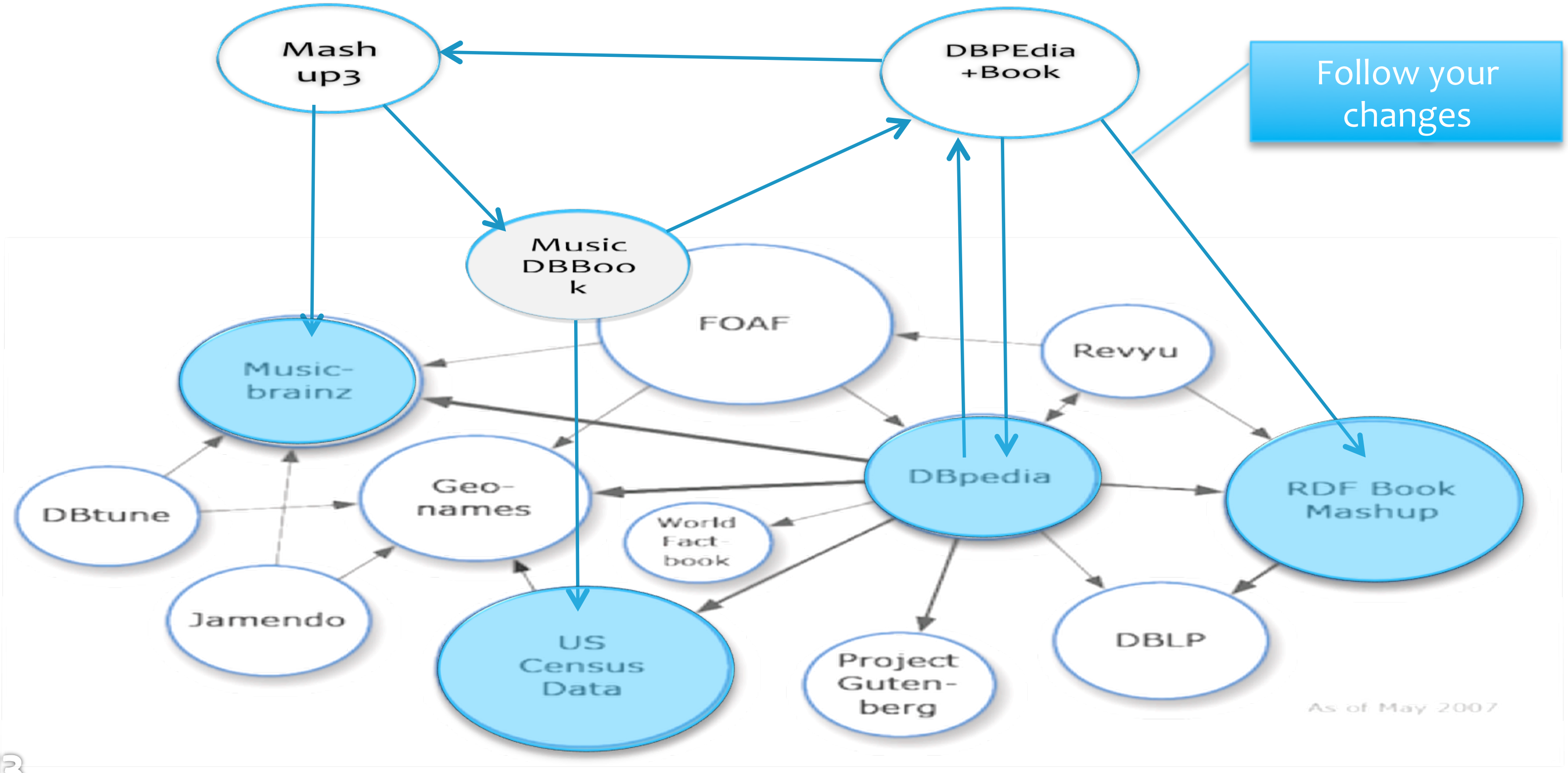
- Enable Massive Optimistic Replication in Linked Data:
  - SPARQL Update 1.1 allows to update RDF data locally.
  - Update feeds to provide streams of updates (e.g. DBPedia Live) -> The writing is "indirect", only if the other participant decides to consume I can change his dataset.
  - Conflict-Free Replicated Datatypes[1] (CRDT) to ensure data consistency when concurrent updates occur.
  - Consistency = System Correctness = Convergence + Intention preservation

[1] M. Shapiro, N. Preguiça, C. Baquero, M. Zawirski. Conflict-free Replicated Data Types. SSS 2011.

```
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/years> <http://dbpedia.org/resource/NCAA_Season_88> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/wikiPageUsesTemplate> <http://dbpedia.org/resource/
Template:PHL_sports_sked_row> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/score> "85"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/team> <http://dbpedia.org/resource/San_Beda_Red_Lions> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/score> "72"^^<http://www.w3.org/2001/XMLSchema#int> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/result> "W"@en .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/team> <http://dbpedia.org/resource/
La_Salle_Green_Hills_Greenies> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://purl.org/dc/terms/
subject> <http://dbpedia.org/resource/Category:
2012_in_the_Philippines> .
<http://dbpedia.org/resource/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
resource/Template:PHL_sports_sked_row> "result16"@en .
<http://dbpedia.org/resource/
```



Live Linked Data Overview



Follow your changes

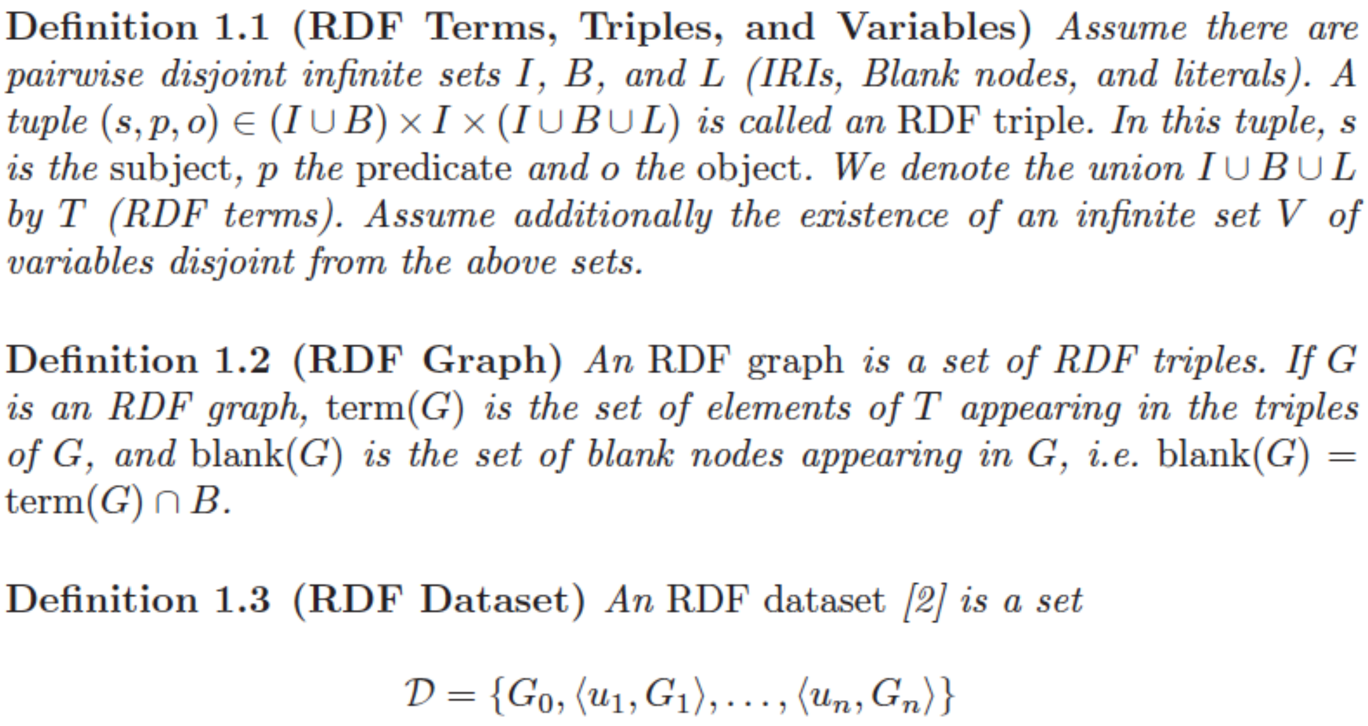
Live Linked Data Overview

- A social network of linked data participants based on a « follow your change » relation.
- Makes Linked Data a « read/write » space : **from linked data 1.0 to linked data 2.0**
- Creates assemblies of datasets and enable « **synchronize and search** » paradigm: between warehousing approach and distributed queries approach.

Enabling LLD with CRDTs

- Construct a CRDT for RDF Graph Stores with SPARQL UPDATE 1.1 Operations with minimal overhead under Live Linked Data conditions:
  - Unknown but steadily growing number of autonomous participants.
    - No central controls (coordination, primary replicas,small core of datacenters)
    - Followers pull from followed.
  - Dynamicity parameters:
    - Degree of change / Change frequency → varies between producers.
    - Growth rate → Positive, knowledge grows.

RDF Datasets



Definition 1.1 (RDF Terms, Triples, and Variables) *Assume there are pairwise disjoint infinite sets I, B, and L (IRIs, Blank nodes, and literals). A tuple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an* RDF triple. *In this tuple, s is the* subject, *p the* predicate *and o the* object. *We denote the union $I \cup B \cup L$ by T (RDF terms). Assume additionally the existence of an infinite set V of variables disjoint from the above sets.*

Definition 1.2 (RDF Graph) *An* RDF graph *is a set of RDF triples. If G is an RDF graph,* $\mathrm{term}(G)$ *is the set of elements of T appearing in the triples of G, and* $\mathrm{blank}(G)$ *is the set of blank nodes appearing in G, i.e.* $\mathrm{blank}(G) = \mathrm{term}(G) \cap B$.

Definition 1.3 (RDF Dataset) *An* RDF dataset *[2] is a set*

$$\mathcal{D} = \{G_0, \langle u_1, G_1 \rangle, \ldots, \langle u_n, G_n \rangle\}$$

http://users.dcc.uchile.cl/~cgutierr/ftp/sparql_semantics.pdf

RDF Datasets

- Assume there are pairwise disjoint infinite sets I , B, and L (IRIs, Blank nodes, and Literals, respectively). A triple (*s*,*p*,*o*) ∈ (I ∪ B)×I×(I ∪ B ∪ L) is called an **RDF triple**. In this tuple, *s* is the subject, *p* the predicate, and *o* the object[1].
- An **RDF Graph** is a set of RDF triples[1].
- An **RDF dataset** is a set: { G, (<u_1>, G_1), (<u_2>, G_2), . . . (<u_n>, G_n) } where G and each G_i are graphs, and each <u_i> is an IRI. Each <u_i> is distinct. G, is called the "default graph" and the pairs (<u_i>, G_i) are called "named graphs"[2].

[1] J. Pérez, M. Arenas and C. Gutiérrez, Semantics and Complexity of SPARQL, ACM Transactions on DB Systems, 2009

[2] http://www.w3.org/TR/sparql11-query/#sparqlDataset

SPARQL Update Queries

Syntax	Formal Operation
INSERT DATA *QuadData*	Dataset-UNION(GS, Dataset(*QuadData*,{},GS,GS))
DELETE DATA *QuadData*	Dataset-DIFF(GS, Dataset(*QuadData*,{},GS,GS))
DELETE *QuadPattern*$_{DEL}$ INSERT *QuadPattern*$_{INS}$ WHERE *GroupGraphPattern*	Dataset-UNION(Dataset-DIFF(GS, Dataset(*QuadPattern*$_{DEL}$,*GroupGraphPattern*,DS,GS)), Dataset(*QuadPattern*$_{INS}$, *GroupGraphPattern*,DS,GS)
DELETE *QuadPattern*$_{DEL}$ WHERE *GroupGraphPattern*	Dataset-UNION(Dataset-DIFF(GS, Dataset(*QuadPattern*$_{DEL}$,*GroupGraphPattern*,DS,GS)), Dataset({}, *GroupGraphPattern*,DS,GS)
INSERT *QuadPattern*$_{INS}$ *UsingClause** WHERE *GroupGraphPattern*	Dataset-UNION(Dataset-DIFF(GS, Dataset({},*GroupGraphPattern*,DS,GS)), Dataset(*QuadPattern*$_{INS}$, *GroupGraphPattern*,DS,GS)
LOAD (SILENT)? *IRIref*	Dataset-UNION(GS, { graph(*documentIRI*) })
CLEAR (SILENT)? DEFAULT	{{}} union $\{(iri_i, G_i) \mid 1 \le i \le n\}$
CREATE (SILENT)? *IRIref*	GS union {(iri, {})} if iri not in graphNames(GS); otherwise, OpCreate(GS, iri) = GS
DROP (SILENT)? IRIref	GS if iri not in graphNames(GS); otherwise, OpDrop(GS, iri_j) = {DG} union $\{(iri_i, G_i) \mid i \ne j \text{ and } 1 \le i \le n\}$

http://www.w3.org/TR/sparql11-update/#formalModel

Insert Ground Triples

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
INSERT DATA
{ GRAPH <http://example/bookStore>
   { <http://example/book1>  ns:price  42 .
     <http://example/book1> dc:creator “A.N.Other” . } }
```

Data before:

```
# Graph: http://example/bookStore
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example/book1> dc:title "Fundamentals of Compiler Design" .
```

Data after:

```
# Graph: http://example/bookStore
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book1> dc:title "Fundamentals of Compiler Design" .
<http://example/book1> ns:price 42 .
<http://example/book1> dc:creator “A.N.Other” 42 .
```

http://www.w3.org/TR/sparql11-update/

Delete-Insert

```
PREFIX foaf:  http://xmlns.com/foaf/0.1/
WITH <http://example/addresses>
DELETE { ?person foaf:givenName 'Bill' }
  INSERT { ?person foaf:givenName 'William' }
  WHERE { ?person foaf:givenName 'Bill’  }
```

Data before:

```
# Graph: http://example/addresses
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
<http://example/president25> foaf:givenName "Bill" .
<http://example/president25> foaf:familyName "McKinley" .
<http://example/president27> foaf:givenName "Bill" .
<http://example/president27> foaf:familyName "Taft" .
<http://example/president42> foaf:givenName "Bill" .
<http://example/president42> foaf:familyName "Clinton" .
```

Data after:

```
# Graph: http://example/addresses
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
<http://example/president25> foaf:givenName "William" .
<http://example/president25> foaf:familyName "McKinley" .
<http://example/president27> foaf:givenName "William" .
<http://example/president27> foaf:familyName "Taft" .
<http://example/president42> foaf:givenName "William" .
<http://example/president42> foaf:familyName "Clinton" .
```

Enabling Convergence in LLD

- Operations can be received on different sites in different orders.
- If the application of all operations over any state commutes, the object is Conflict-Free
  - $S \circ op_i \circ op_j = S \circ op_j \circ op_i$ => Convergence
- Basic SPARQL update application does not commute

  [INSERT DATA(x); DELETE DATA(x);INSERT DATA(x)] = {x}

  [INSERT DATA(x); INSERT DATA(x);DELETE DATA(x) = { }

  RDF Data-Stores are not CRDTs

Enabling Intentions Preservation in LLD

- Observed effect of SPARQL update queries at generation time should be preserved at re-execution time…
  - If an Sparql query inserts a triple at generation time, this triple will be added at all sites whatever the state of the store at re-execution time…
- Intentions can be impossible to preserve (non-deterministic operations) or impossible without more order constraints (re-execution time evaluation)

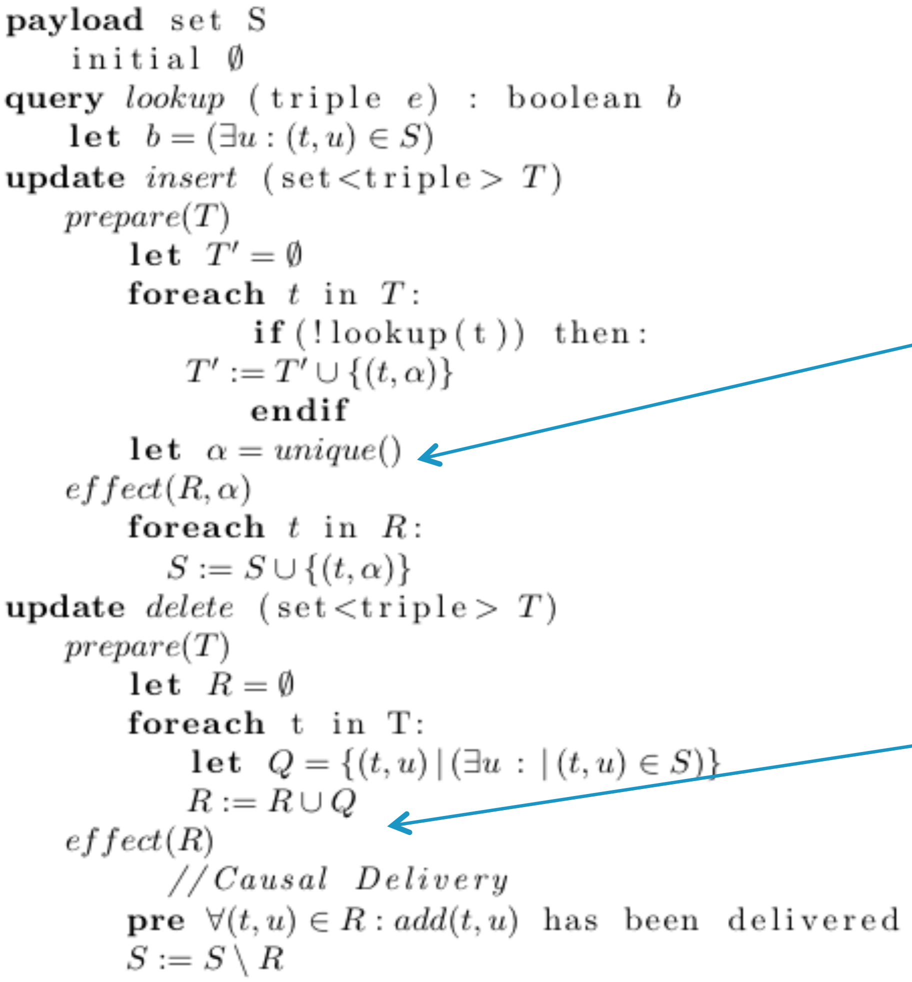
```
PREFIX foaf:  http://xmlns.com/foaf/0.1/
WITH <http://example/addresses>
DELETE { ?person foaf:givenName 'Bill' }
INSERT { ?person foaf:givenName 'William' }
WHERE { ?person foaf:givenName 'Bill’  }
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT DATA {
  _:book1 dc:title   "A new book" ;
              dc:creator "A.N.Other" .
}
```

A CRDT for SPARQL update

- Rewrite SPARQL UPDATE Operations to another type that does commute, while preserving the original semantics.
- Graph Update operations work over RDF-Graphs, and can be expressed in terms of the basic INSERT and DELETE of ground triples.
  - RDF-Graphs are Sets of RDF-Triples, and CRDTs for Sets with Insert and Delete already exists…

SU-Set : A Sparql-Update CRDT



```
payload set S
  initial $\emptyset$
query $lookup$ (triple $e$) : boolean $b$
  let $b = (\exists u : (t,u) \in S)$
update $insert$ (set<triple> $T$)
  $prepare(T)$
    let $T' = \emptyset$
    foreach $t$ in $T$:
        if(!lookup(t)) then:
      $T' := T' \cup \{(t,\alpha)\}$
        endif
    let $\alpha = unique()$
  $effect(R,\alpha)$
    foreach $t$ in $R$:
      $S := S \cup \{(t,\alpha)\}$
update $delete$ (set<triple> $T$)
  $prepare(T)$
    let $R = \emptyset$
    foreach t in T:
      let $Q = \{(t,u) | (\exists u : | (t,u) \in S)\}$
      $R := R \cup Q$
  $effect(R)$
      $//Causal\ Delivery$
    pre $\forall(t,u) \in R : add(t,u)$ has been delivered
    $S := S \setminus R$
```

Same id for all triples inserted together, unicity is preserved

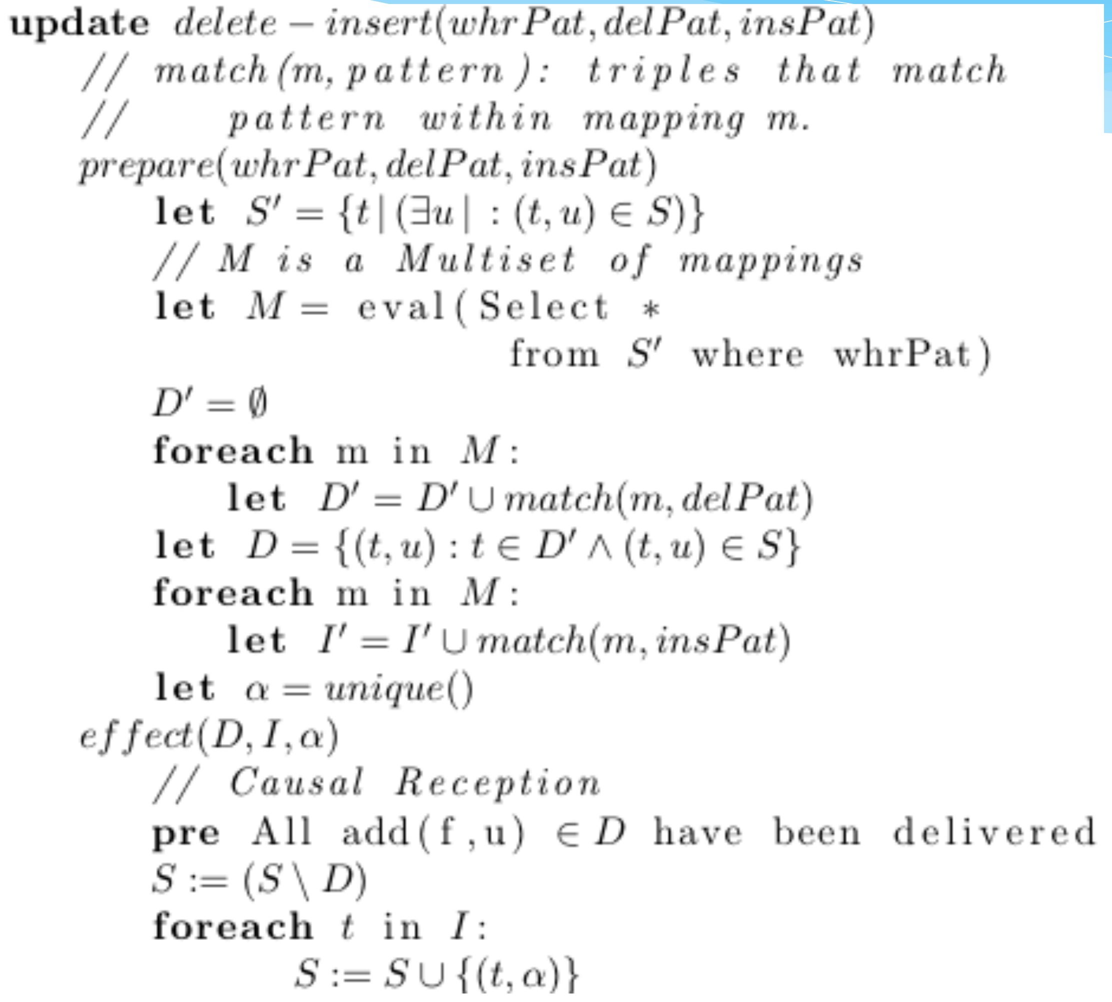
Less expensive in Communication

Need to send (triple,id) when deleting

More expensive in communication

Better for LLD as knowledge grows…

SU-Set



```
update $delete - insert(whrPat, delPat, insPat)$
    // match(m, pattern): triples that match
    //     pattern within mapping m.
    $prepare(whrPat, delPat, insPat)$
        let $S' = \{t \,|\, (\exists u \,|\, : (t, u) \in S)\}$
        // M is a Multiset of mappings
        let $M =$ eval(Select *
                           from $S'$ where whrPat)
        $D' = \emptyset$
        foreach m in $M$:
            let $D' = D' \cup match(m, delPat)$
        let $D = \{(t, u) : t \in D' \wedge (t, u) \in S\}$
        foreach m in $M$:
            let $I' = I' \cup match(m, insPat)$
        let $\alpha = unique()$
    $effect(D, I, \alpha)$
        // Causal Reception
        pre All add(f,u) $\in D$ have been delivered
        $S := (S \setminus D)$
        foreach $t$ in $I$:
            $S := S \cup \{(t, \alpha)\}$
```

Implementation

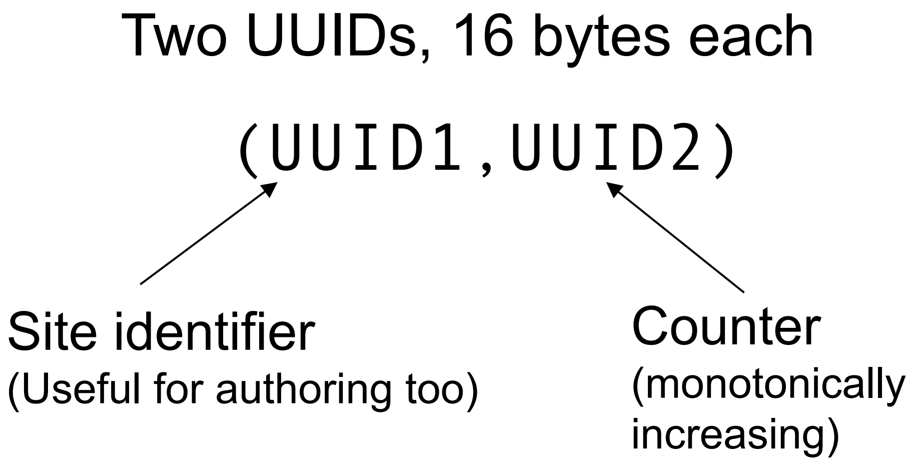
- SU-Set embedded into Corese, a reference implementation of RDF-Stores and SPARQL Update by Wimmics team.
  - Rewrite Sparql Update into SUSet, execute and log.
  - "Follow your change" implemented with anti-entropy over logs and version vectors.

How much we need to pay to have eventual consistency ?

- Time Overhead :
  - Adding an id to each element is linear.
  - Selection and lookup is not affected by many pairs with the same triple.
- Round and # of messages Overhead :
  - Convergence after one round, one message per operation → Optimal

Validation – Space Overhead

- 32 bytes per 1 billion triples = 32 GB → 1 Ipod
- Semantic Stores already use an internal id → Reuse it
- Extra pairs produced by concurrent insertions could cause problems…
- A version vector for each site: Max size= number of participants (~300-800).



Two UUIDs, 16 bytes each

(UUID1,UUID2)

Site identifier
(Useful for authoring too)

Counter
(monotonically increasing)

Dynamicity: DBPedia Live

- Framework to register changes in DBPedia in near real-time.
- Generates one file with inserted triples and one with deleted triples approximately each 10 seconds.
  - No pattern operations logged ➔ No Overhead here.
- **Many more insertions than deletions**
  - Good for SU-Set.
- Many triples inserted per operation
  - Even better for SU-Set

SU-Set Communication overhead on DBPedia Live

7 days of streaming
No concurrent insertions
IdSize[1]: 0,096 Kb
Avg triple size[1]: 0,155 Kb

		Size (MB)		
Operation	# of Triples	Without ids	1 id per triple	1 id per operation
21957 Inserts	21762190	3294,08	5334,29	3296,6
21957 Deletes	1755888	265,78	164,61	430,4
Overhead			54,47%	4,68%

[1] Measured with 'wc –c' over DBPedia Live log files

Conclusion

- Live Linked Data makes linked data “writable” and allows a new query paradigm
- SU-Set is a CRDT for RDF-Graphs updated with SPARQL-Update 1.1 that ensure eventual consistency and intentions on Live Linked Data.
- Future work:
  - Finish and release LLD-Corese.
  - Write the composed CRDT for RDF-Datasets
  - Benchmark Live Linked Data.