Live Linked Data

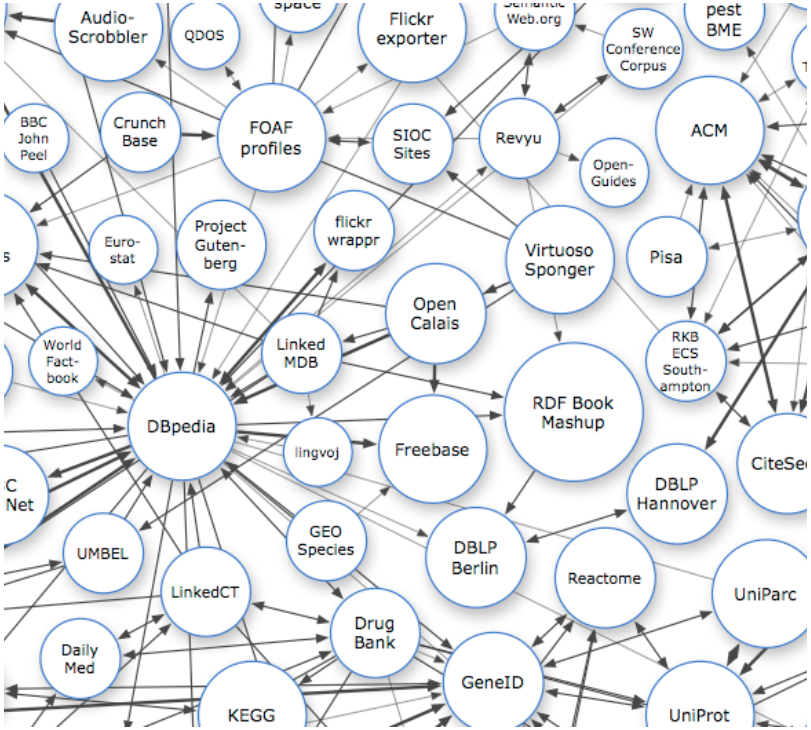
or how to make linked data writable with massive optimistic replication and Conflict-Free Replicated Data Types

Pascal Molli, 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

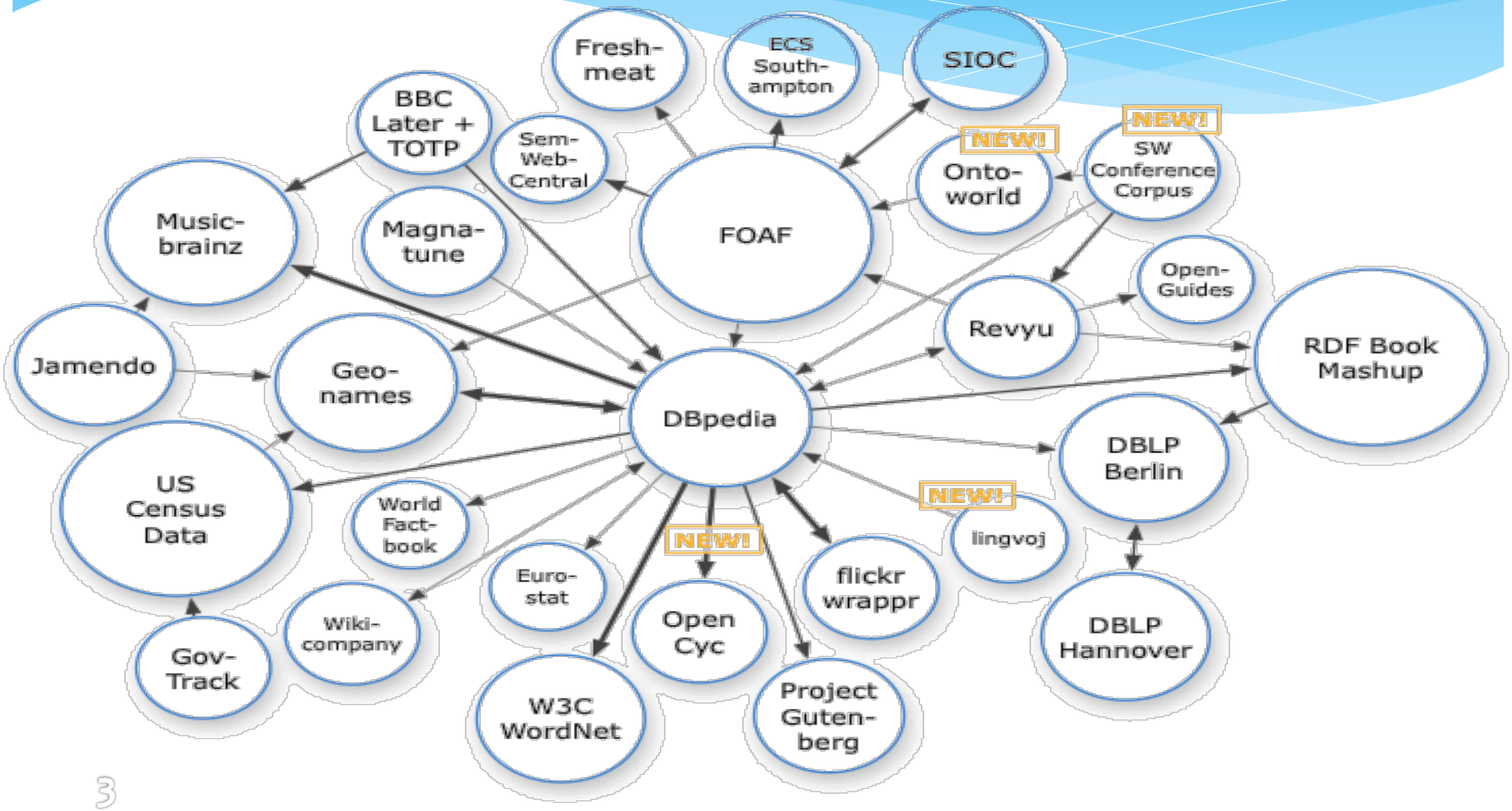
Linked Data

* In 2012, 32 billion RDF triples distributed between ~300 to 700 **autonomous participants**
* **Autonomous participants** agree on a set of principles that allows publishing, querying and browsing of RDF data, distributed across different servers

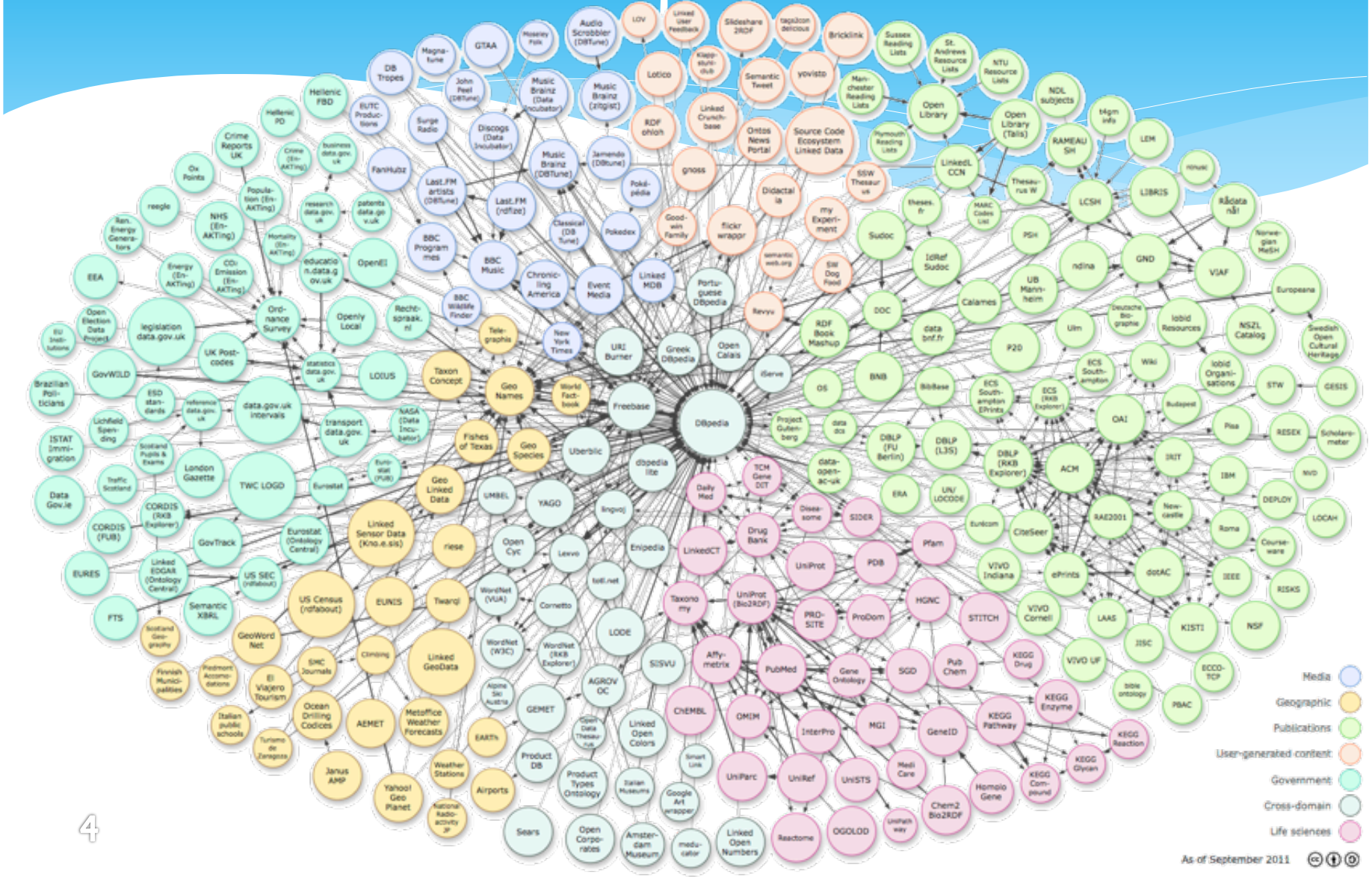


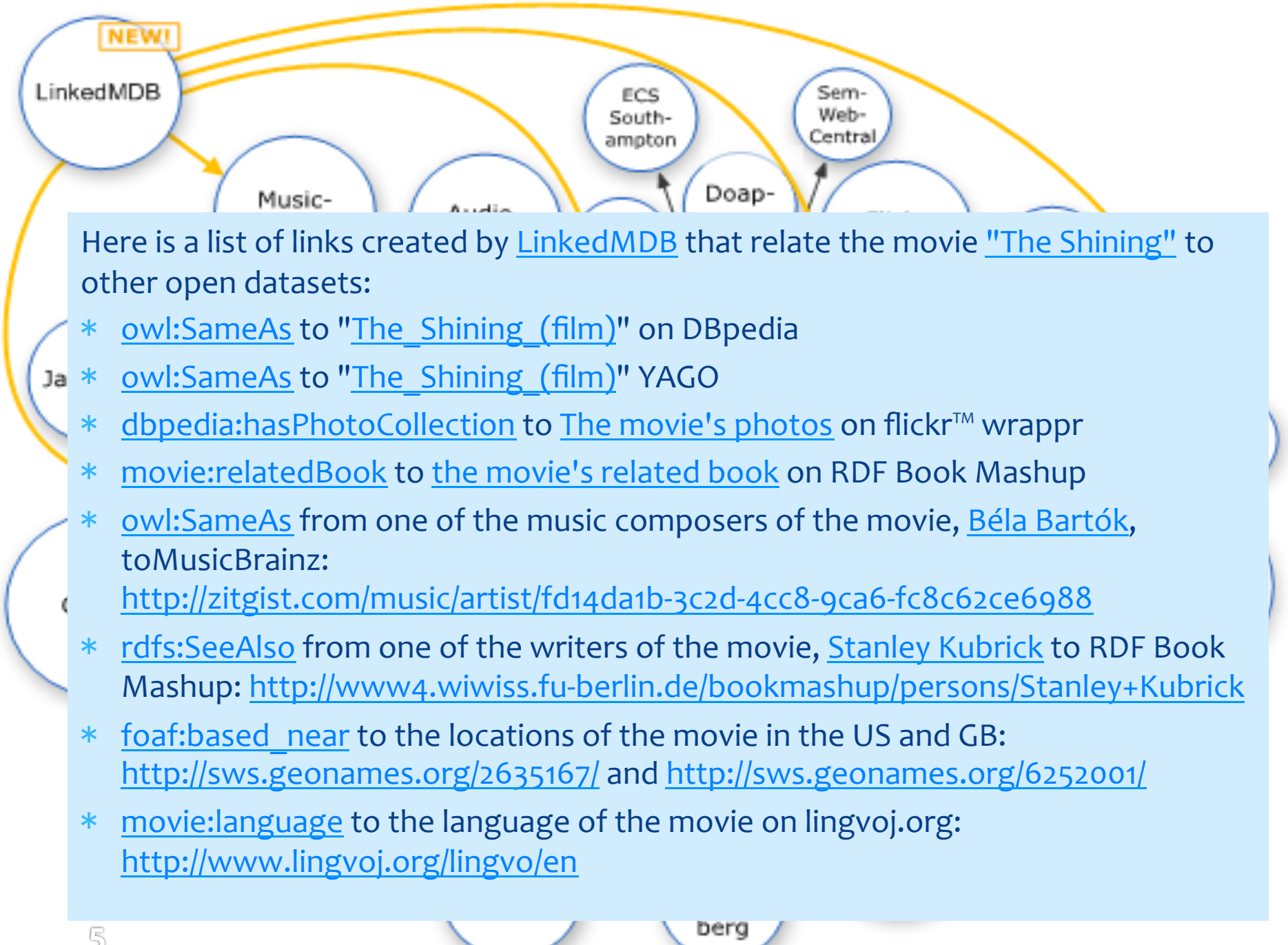
http://events.linkeddata.org/ldow2012/papers/ldow2012-paper-14.pdf

2007 October



2011 September





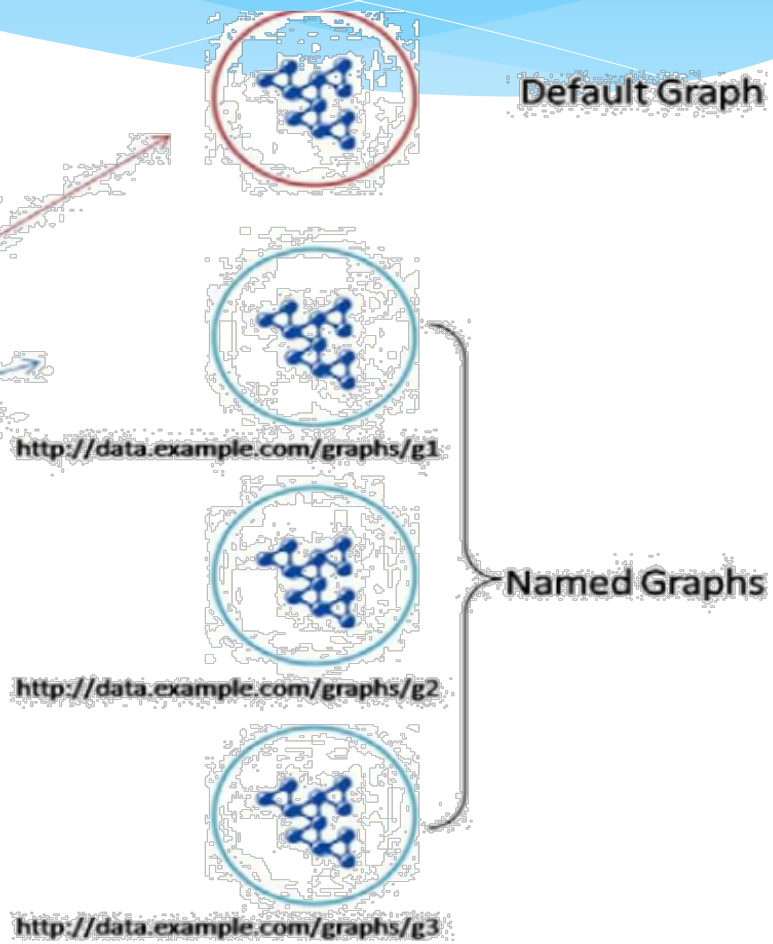
Here is a list of links created by LinkedMDB that relate the movie "The Shining" to other open datasets:

* 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

```
PREFIX g: <http://data.example.com/graphs/>
PREFIX ex: <...>
SELECT *
FROM <...>
FROM NAMED g:g1
FROM NAMED g:g2
FROM NAMED g:g3
WHERE {
    ?s ex:p1 ex:o1 ; ex:p2 ex:o2 .
    GRAPH g:g1 { ?s ex:p3 ex:o3 }
    GRAPH ?g {
        ex:s1 ex:p4 ?s .
        ex:s1 ex:p5 ex:o5 .
    }
}
```



Pb of Freshness

http://www.cambridgesemantics.com/semantic-university/learn-sparql

Queries in Linked Data: Distributed Queries

Find the birth dates of all of the actors in Star Trek: The Motion Picture

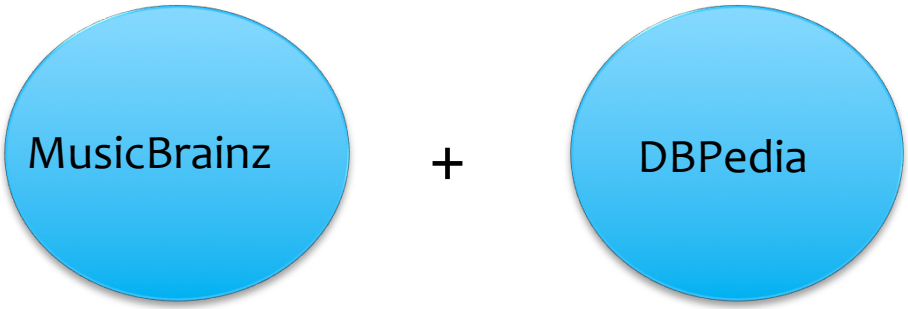
```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?actor_name ?birth_date
FROM <http://www.w3.org/People/Berners-Lee/card> # placeholder graph
WHERE {
  SERVICE <http://data.linkedmdb.org/sparql> {
    <http://data.linkedmdb.org/resource/film/675> movie:actor ?actor .
    ?actor movie:actor_name ?actor_name
  }
  SERVICE <http://dbpedia.org/sparql> {
    ?actor2 a dbpedia:Actor ; foaf:name ?actor_name_en ; dbpedia:birthDate ?birth_date .
    FILTER(STR(?actor_name_en) = ?actor_name)
  }
}
```

Pb of endpoint reliability and performances (and discovery?)

http://www.cambridgesemantics.com/semantic-university/learn-sparql

Querying LOD: Sometimes OK

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

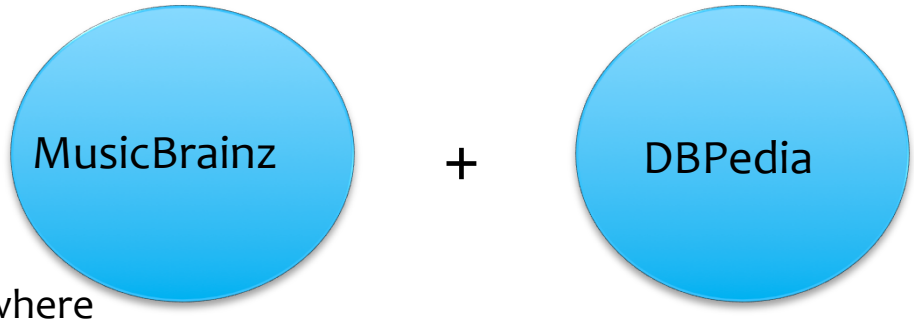


Select ?country ?city where
{http://musicbrainz.org/Vicente_Fernandez http://musicbrainz.org/fromCountry ?country.
?country http://dbpedia.org/ontology/largestCity ?city . }

?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 . }

?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

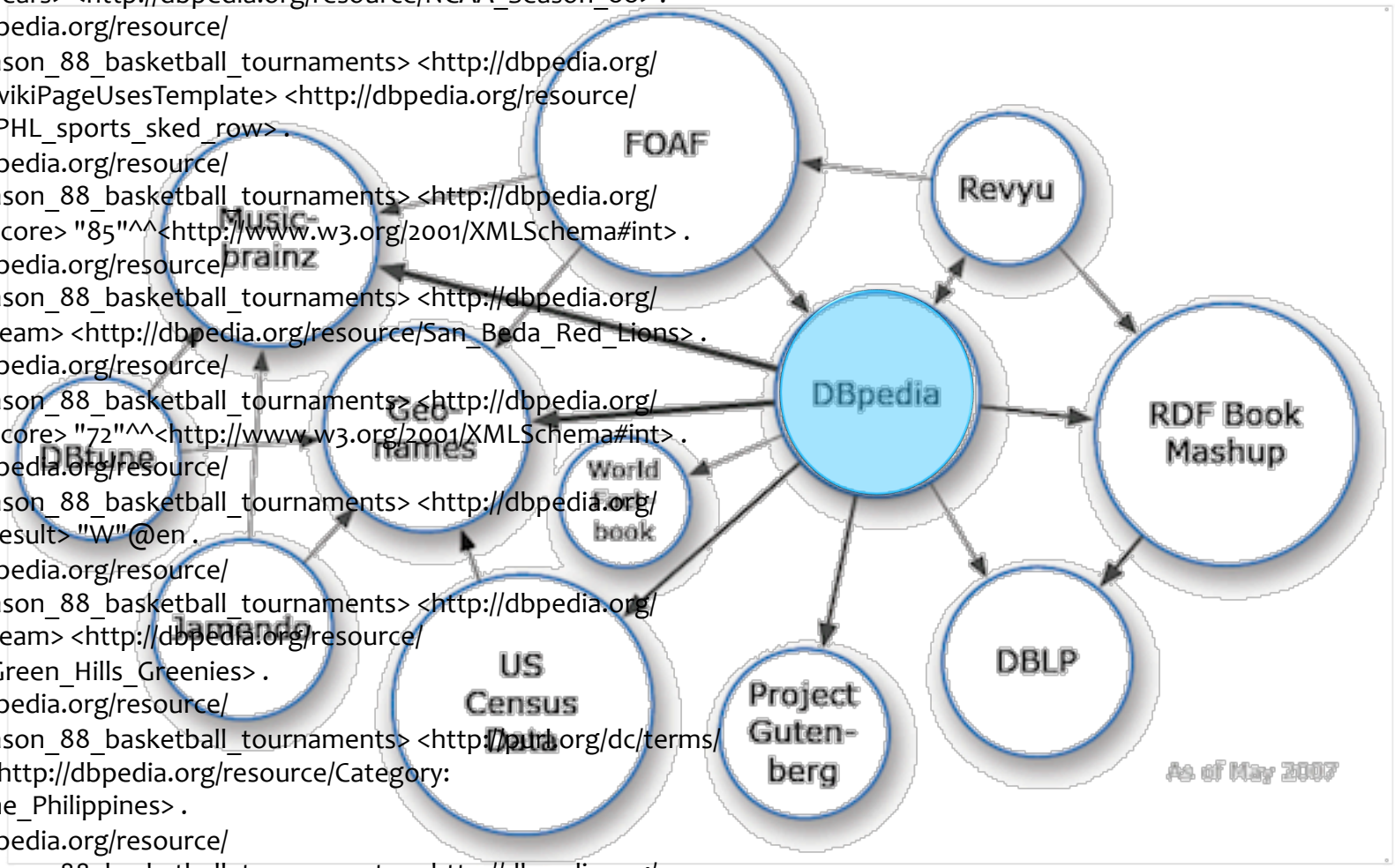
* Enabling Massive Optimistic Replication in Linked Data:
  * SPARQL update 1.1 allows to update RDF data on one .
  * Update feeds to provide streams of updates (as in DBPedia Live)
  * Conflict-Free Replicated Datatypes (CRDT) to manage consistency of replicates

- M. Shapiro, N. Preguiça, C. Baquero, M. Zawirski.Conflict-free Replicated Data Types. 13th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS). Grenoble, France, 10-12 October 2011.
- Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot-undo: Distributed collaborative editing system on p2p networks. IEEE Transactions on Parallel Distributed Systems, 21(8):1162–1174, 2010

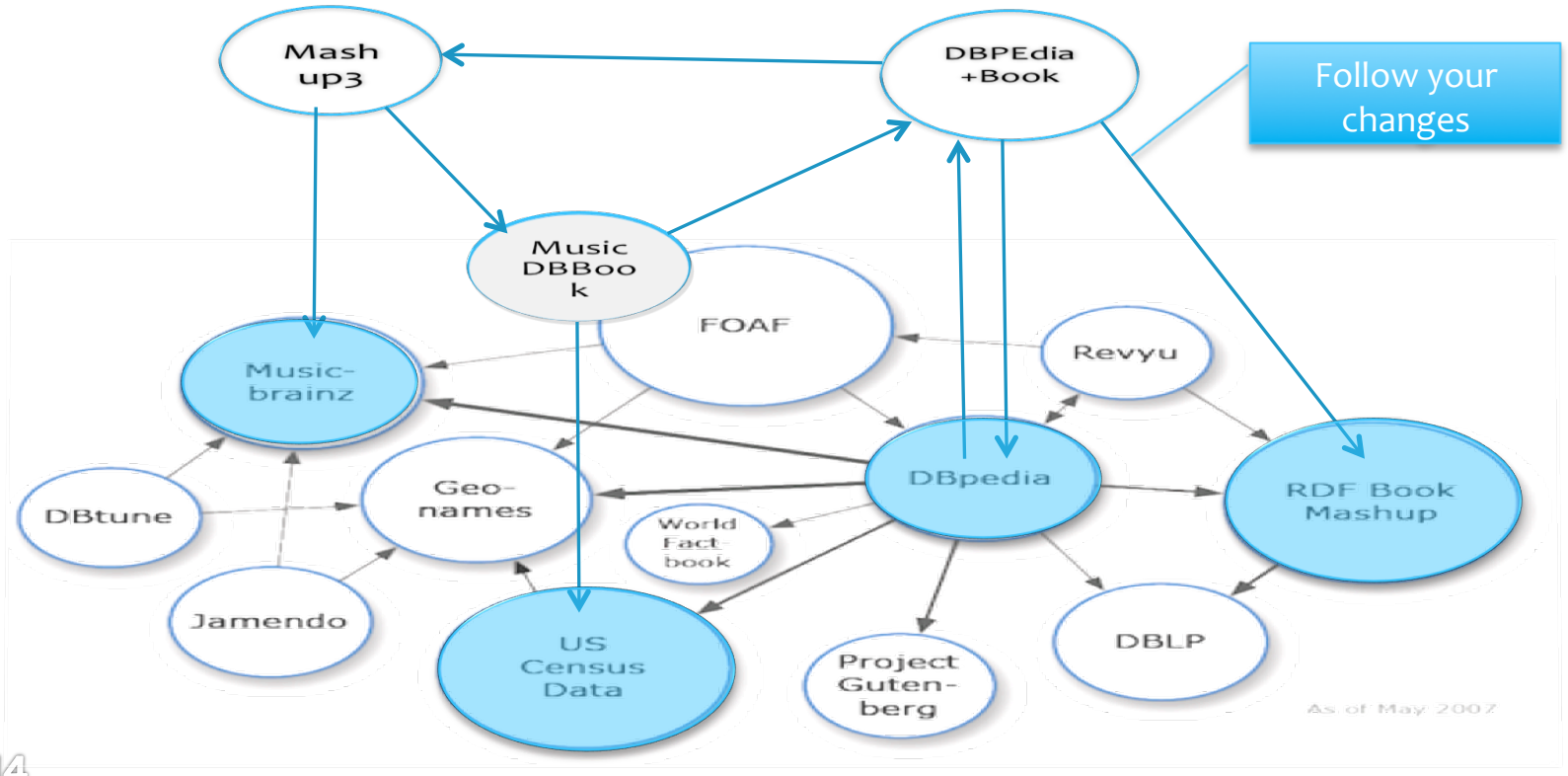
Related Works

* **Database: Multi-Master Replication and Eventual Consistency**
  * Duplicated Database (Johnson 75), base of Usenet, Multi-Master Database Replication, Eventual Consistency in NoSQL stores
  * P. Johnson and R. Thomas. Rfc677: The maintenance of duplicate databases, 1976.
* **Distributed System: Optimistic Replication and Eventual Consistency**
  * Yasushi Saito and Marc Shapiro. Optimistic replication. ACM Comput. Surv., 37(1):42–81, 2005.
  * Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot-undo: Distributed collaborative editing system on p2p networks. IEEE Transactions on Parallel Distributed Systems, 21(8):1162–1174, 2010
* **CSCW: Multi-Synchronous Collaboration Model and Eventual Consistency**
  * Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, and David Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. ACM Transactions on Computer-Human Interaction, 5(1):63–108, 1998.
  * Gérald Oster, Pascal Urso, Pascal Molli, and Abdessamad Imine. 2006. Data consistency for P2P collaborative editing. In Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (CSCW '06). ACM, New York, NY, USA, 259-268. DOI=10.1145/1180875.1180916 http://doi.acm.org/10.1145/1180875.1180916
* **Semantic Web: Synchronization of RDF stores**
  * Tim Berners-Lee and Dan Connolly. Delta: an ontology for the distribution of differences between rdf graphs. http://www.w3.org/DesignIssues/ Diff, 2004
  * Giovanni Tummarello, Christian Morbidoni, Reto Bachmann-Gmür, and Orri Erling. Rdfsync: Efficient remote synchronization of rdf models. In 6th International and 2nd Asian Semantic Web Conference (ISWC + ASWC), pages 537–551, 2007.
  * SD Share Group W3C, http://www.sdshare.org

<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/
NCAA_Season_88_basketball_tournaments> <http://dbpedia.org/
property/team> <http://dbpedia.org/resource/FAC_Generals> .



Live Linked Data Overview



Live Linked Data Overview

* A social network for linked data participant 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 Optimistic Replication Model

* Let a social network of unknown number of linked data nodes linked by a *"follow your change"* relation
* Each linked data node:
  * Executes SPARQL Update queries locally.
  * Publishes these operations in "Live Streams" (using W3C standards as SD Share group), hypothesis: Operations are eventually delivered (reliable broadcast).
  * Other nodes consume and re-execute them.
* The system is correct if:
  * Eventual consistency (Convergence) and Intentions

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,* $\text{term}(G)$ *is the set of elements of T appearing in the triples of G, and* $\text{blank}(G)$ *is the set of blank nodes appearing in G, i.e.* $\text{blank}(G) = \text{term}(G) \cap B$.

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

Datasets examples

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book2> ns:price 42 .
<http://example/book2> dc:title "David Copperfield" .
<http://example/book2> dc:creator "Edmund Wells" .

 # with blank nodes
 ex:John   foaf:knows      _:p1
 _:p1     foaf:birthDate   04-21
```

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) | 1 ≤ i ≤ 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) | i ≠ j and 1 ≤ i ≤ 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; 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/

Insert Data with Blank Node

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

Data before:

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

Data after:

```
# Default Graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example/book1> dc:title "Fundamentals of Compiler Design" .
_:b1 dc:title   "A new book"
_:b1 dc:creator "A.N.Other"
```

Delete Ground Triples

```
PREFIX dc: http://purl.org/dc/elements/1.1/
DELETE DATA {
  <http://example/book2> dc:title "David Copperfield" ;  dc:creator "Edmund Wells" .
}
```

Data before:

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book2> ns:price 42 .
<http://example/book2> dc:title "David Copperfield" .
<http://example/book2> dc:creator "Edmund Wells" .
```

Data after:

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book2> ns:price 42 .
```

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
* RDF stores will converge if all SPARQL update operations are commutative
  * $S \circ op_i \circ op_j = S \circ op_i \circ op_j$
* Basic SPARQL update are not commutative
  * [INSERT DATA(x); DELETE DATA(x);INSERT DATA(x)]
  * <>[INSERT DATA(x); INSERT DATA(x);DELETE DATA(x)

Enabling Intentions Preservation in LLD

* Observed effect of SPARQL update queries at generation time should be preserved at integration time…
  * If a Sparql query insert a triple at generation time, this triple will be added at all sites whatever the state of the store at integration time…
* Intentions can be impossible to preserve (non-deterministic operations) or impossible without more order constraints (integration 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" .
}
```

SU-SET: a CRDT for SPARQL update

* Rewrite SPARQL Update queries into SU-SET commutative operations that will exchanged on the social editing network…
* SU-SET must ensure eventual consistency and preserve semantic of original SPARQL update queries (at generation time)

SU-Set : A Sparql-Update Conflict-free Replicated Data Type

```
payload set S
    initial ∅
query lookup (triple t) : boolean b
    let b = (∃u : (t,u) ∈ S)
update insert (set<triple> T)
    atSource(T)
        let α = unique()
    downstream(T, α)
        let R = {(t,α) : t ∈ T}
        S := S ∪ R
update delete (set<triple> T)
    atSource(T)
        let R = ∅
        foreach t in T:
            let Q = {(t,u) | (∃u : | (t,u) ∈ S)}
            R := R ∪ Q
    downstream(R)
        // Causal Reception
        pre All add(t,u) delivered
        S := S \ R
```

Abstract operation is Sparql Update

Same id for all triples inserted together saves communication

Delete all pairs associated to each triple. Can be expensive.

SU-Set

```
update delete−insert(whrPat, delPat, insPat)
    // match(m, pattern): triples matching
    //     pattern within mapping m.
    atSource(whrPat, delPat, insPat)
        let S' = {t | (∃u | : (t, u) ∈ S)}
        // M is a Multiset of mappings
        let M = eval(Select *
                     from S' where whrPat)
        let D' = ∅
        foreach m in M:
            D' = D' ∪ match(m, delPat)
        let D = {(t, u) : t ∈ D' ∧ (t, u) ∈ S}
        let I' = ∅
        foreach m in M:
            let I' = I' ∪ match(m, insPat)
        let α = unique()
    downstream(D, I', α)
        // Causal Reception
        pre All add(f,u) ∈ D delivered
        let I = {(i, α) : i ∈ I'}
        S := (S \ D) ∪ I
```

SU-SET

* With this specification, all SU-SET operations are commutative except:
  * Ins(x);Del(x) (with same x)
* Requiring the causality property (causal broadcast) avoid this problem, but increase communication cost
* Forcing Tombstones also avoid the problem but increase space complexity in case of concurrent delete…

SU-Set on Live Linked Data

Insert Data
{France in Asia}

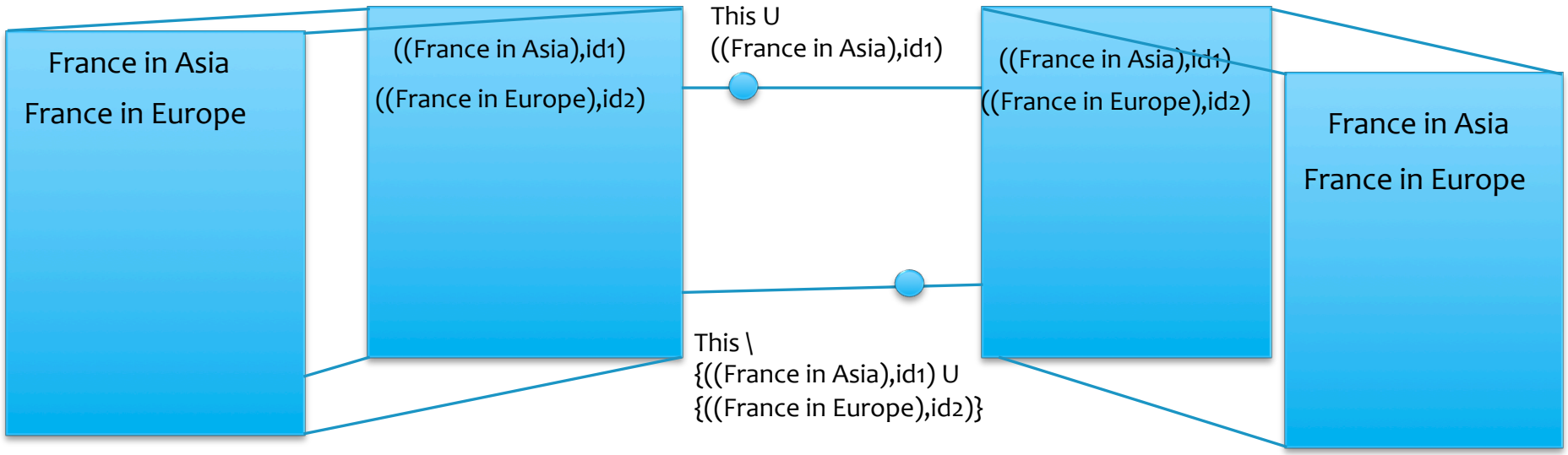
Insert({(France in Asia)})

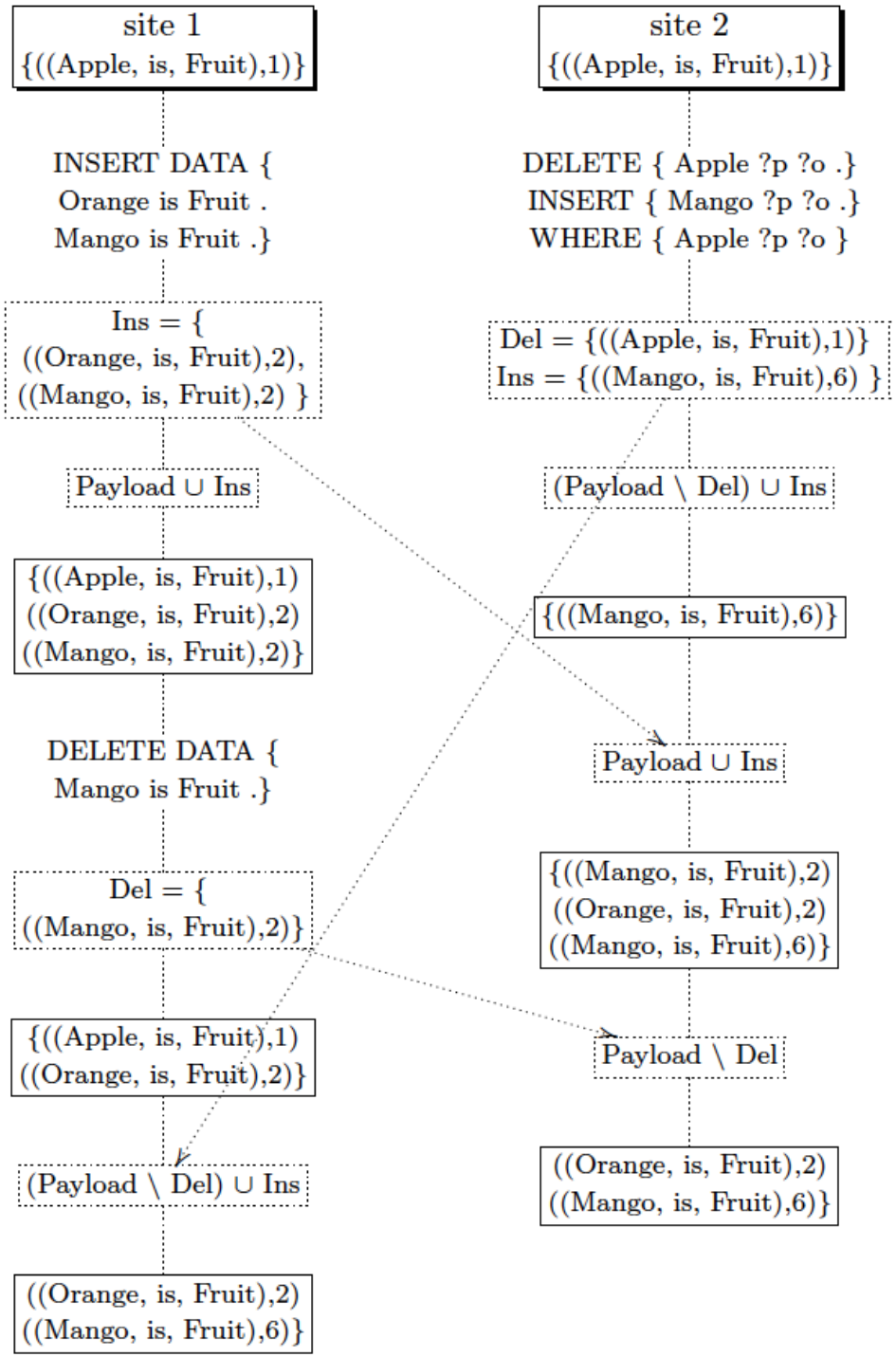
This U
{((France in Asia),id1)}

Delete(
{(France in Asia),
(France in Europe)})

This \
{((France in Asia),id1)
{((France in Europe),id2)}

DELETE {?c in Asia}
INSERT {?c in Europe}
WHERE {?c in Asia}




site 1
{((Apple, is, Fruit),1)}
INSERT DATA {
Orange is Fruit .
Mango is Fruit .}
Ins = {
((Orange, is, Fruit),2),
((Mango, is, Fruit),2) }
Payload ∪ Ins
{((Apple, is, Fruit),1)
((Orange, is, Fruit),2)
((Mango, is, Fruit),2)}
DELETE DATA {
Mango is Fruit .}
Del = {
((Mango, is, Fruit),2)}
{((Apple, is, Fruit),1)
((Orange, is, Fruit),2)}
(Payload \ Del) ∪ Ins
((Orange, is, Fruit),2)
((Mango, is, Fruit),6)}
site 2
{((Apple, is, Fruit),1)}
DELETE { Apple ?p ?o .}
INSERT { Mango ?p ?o .}
WHERE { Apple ?p ?o }
Del = {((Apple, is, Fruit),1)}
Ins = {((Mango, is, Fruit),6) }
(Payload \ Del) ∪ Ins
{((Mango, is, Fruit),6)}
Payload ∪ Ins
{((Mango, is, Fruit),2)
((Orange, is, Fruit),2)
((Mango, is, Fruit),6)}
Payload \ Del
((Orange, is, Fruit),2)
((Mango, is, Fruit),6)}

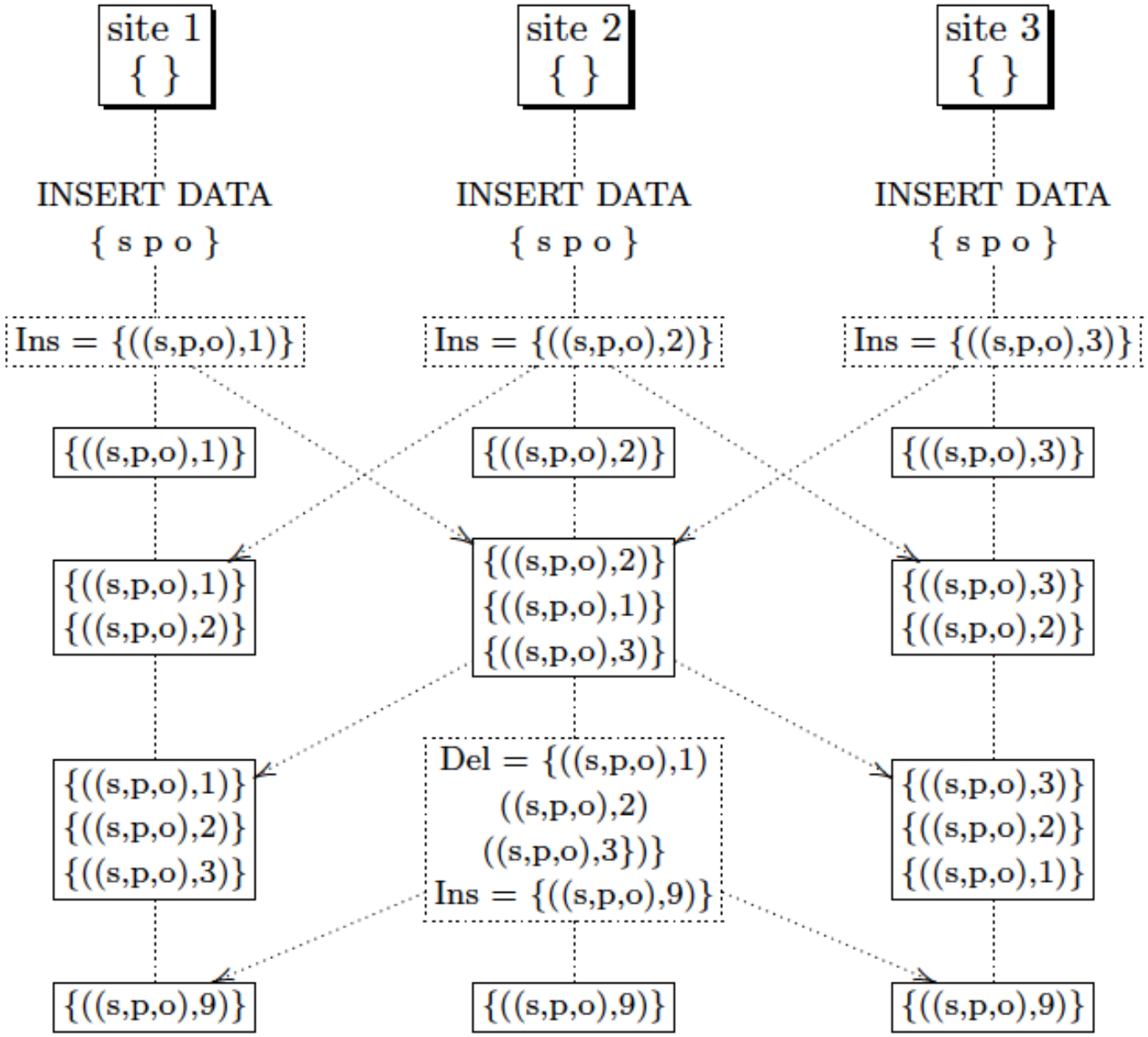
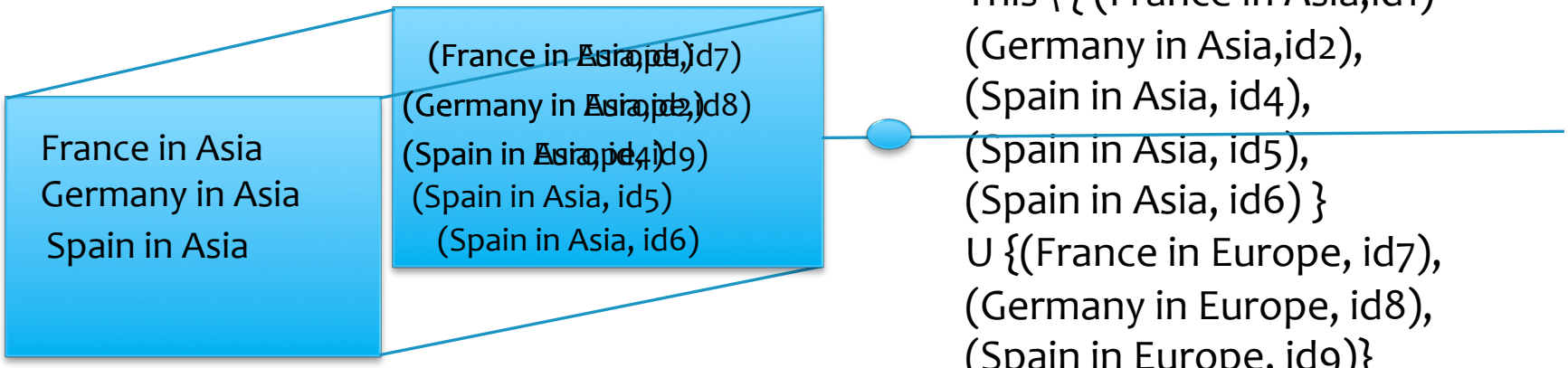


Figure 7 Generation and garbage collection of pairs referring to the same triple

SparQL-Update Pattern Queries



Delete {?c in Asia}
Insert {?c in Europe}
Where {?c in Asia}

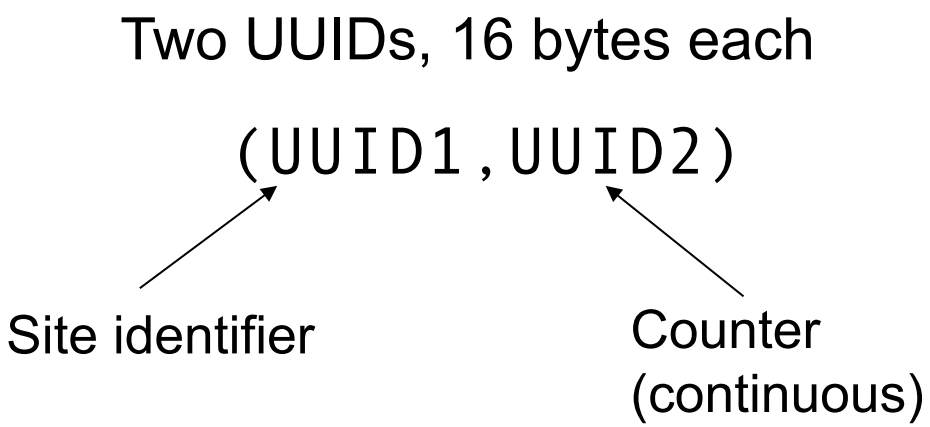
This \ { (France in Asia,id1)
(Germany in Asia,id2),
(Spain in Asia, id4),
(Spain in Asia, id5),
(Spain in Asia, id6) }
U {(France in Europe, id7),
(Germany in Europe, id8),
(Spain in Europe, id9)}

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…



Communication Cost with DBPedia Live with SU-Set

* DBPedia Live generates one file with triples inserted and one with triples deleted approximately each 10 seconds
  * No pattern operations → No overhead here.
* **Many more insertions than deletions**
  * Insertions are cheap, they only need one id.
* Many triples per insertion
  * More triples inserted at a time is cheaper.

SU-Set Communication overhead on DBPedia Live

7 days of streaming
No concurrent insertions

Operation	# of Triples	Size (MB)		
		No ids	1 id per triple	1 id per operation
21957 Inserts	21762190	3403,4	4469,89	3404,6
21957 Deletes	1755888	238,46	324,5	324,5
Overhead			31,64%	2,39%

Communication Complexity

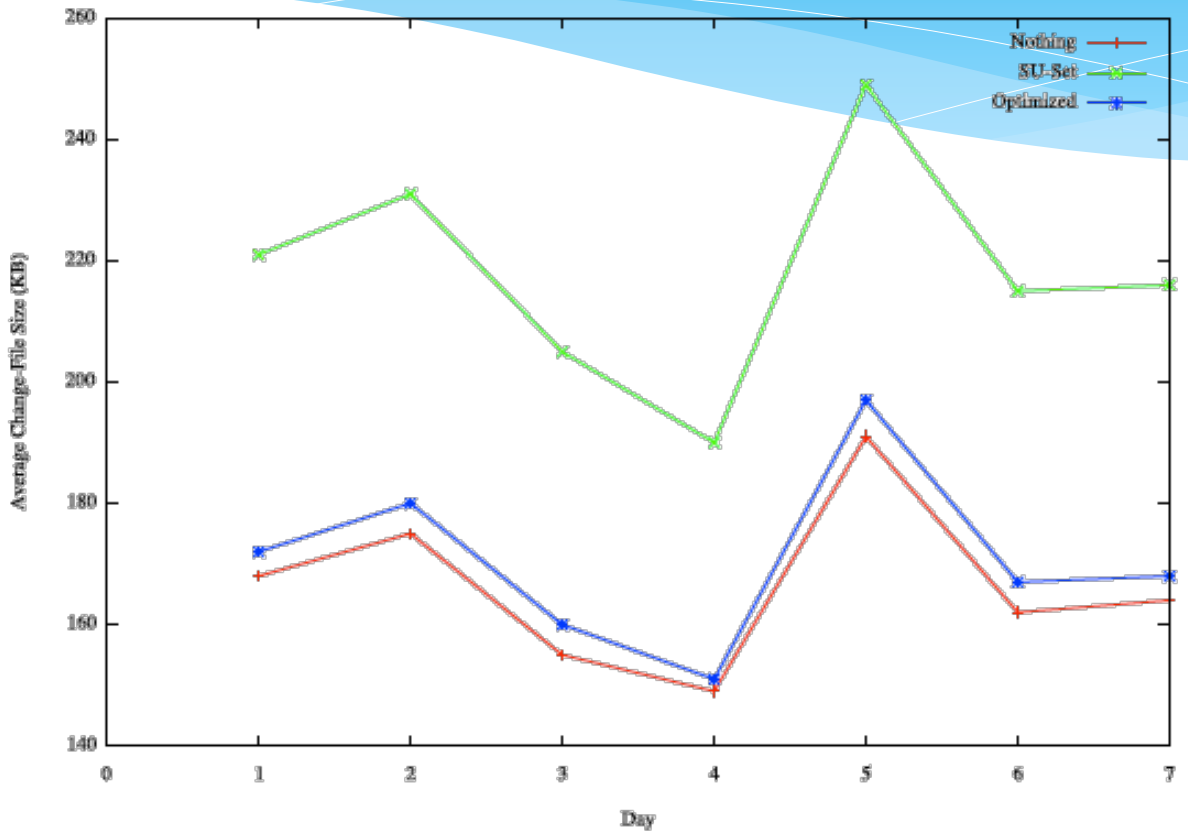


Figure 9 Communication complexity for SU-Set and its optimized version in DBPedia Live case.

Conclusion

* Live Linked Data makes linked data “writable” and allow a new query paradigm
* SU-Set is a CRDT for RDF-datasets updated with SPARQL-Update 1.1 that ensure eventual consistency and intentions on live linked data
* Time to embed in real system
  * Embed SU-Set in “Corese” engine of Wimmics
  * In dev…

Future work

* Divergence – How to evaluate divergence between node in live linked data
* Provenance – How to collect it efficiently in LLD
* Resource discovery in LLD
* Queries with weakly consistent replicated datasets

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