MarkLogic WORLD 2017

# Getting the Most from MarkLogic Semantics

John Snelson, Principal Engineer, MarkLogic
Stephen Buxton, Sr. Director, Product Management, MarkLogic

# Introduction

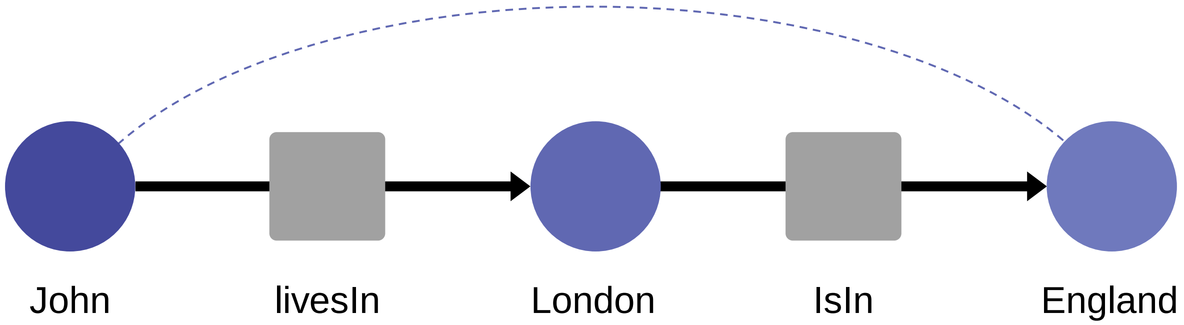
# Semantics Is: A New Way to Organize Data

**Data** is stored in **RDF** triples expressed as

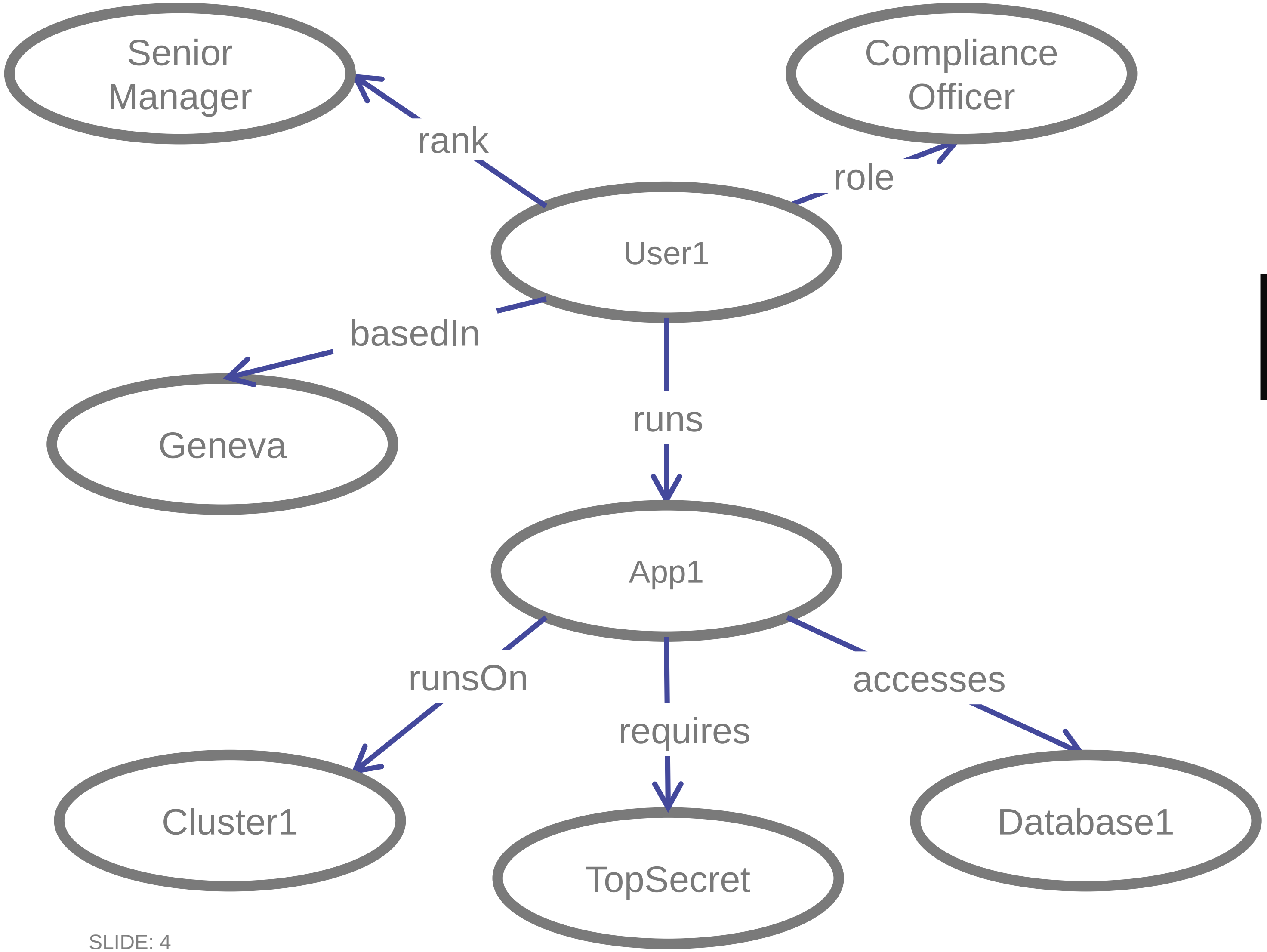
| :**Subject** | : **Predicate** | : **Object** |
|---|---|---|
| John Smith | : livesIn | : London |
| London | : isIn | : England |

**Query** with SPARQL, gives us simple lookup .. and more!

Find people who live in (a place that's in) England



# Triples Alongside Documents



Senior Manager

rank

Compliance Officer

role

User1

basedIn

Geneva

runs

App1

runsOn

requires

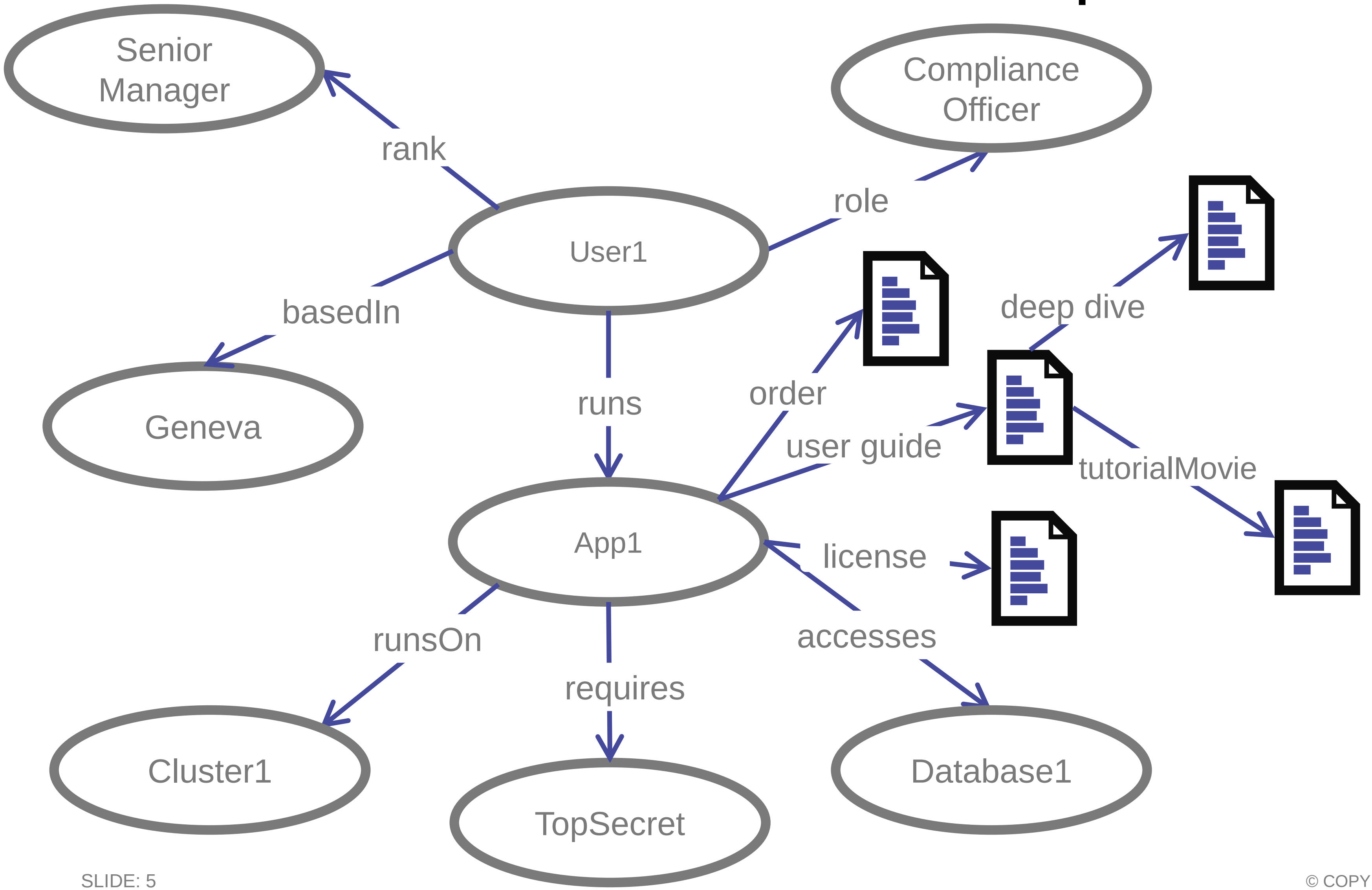
accesses

Cluster1

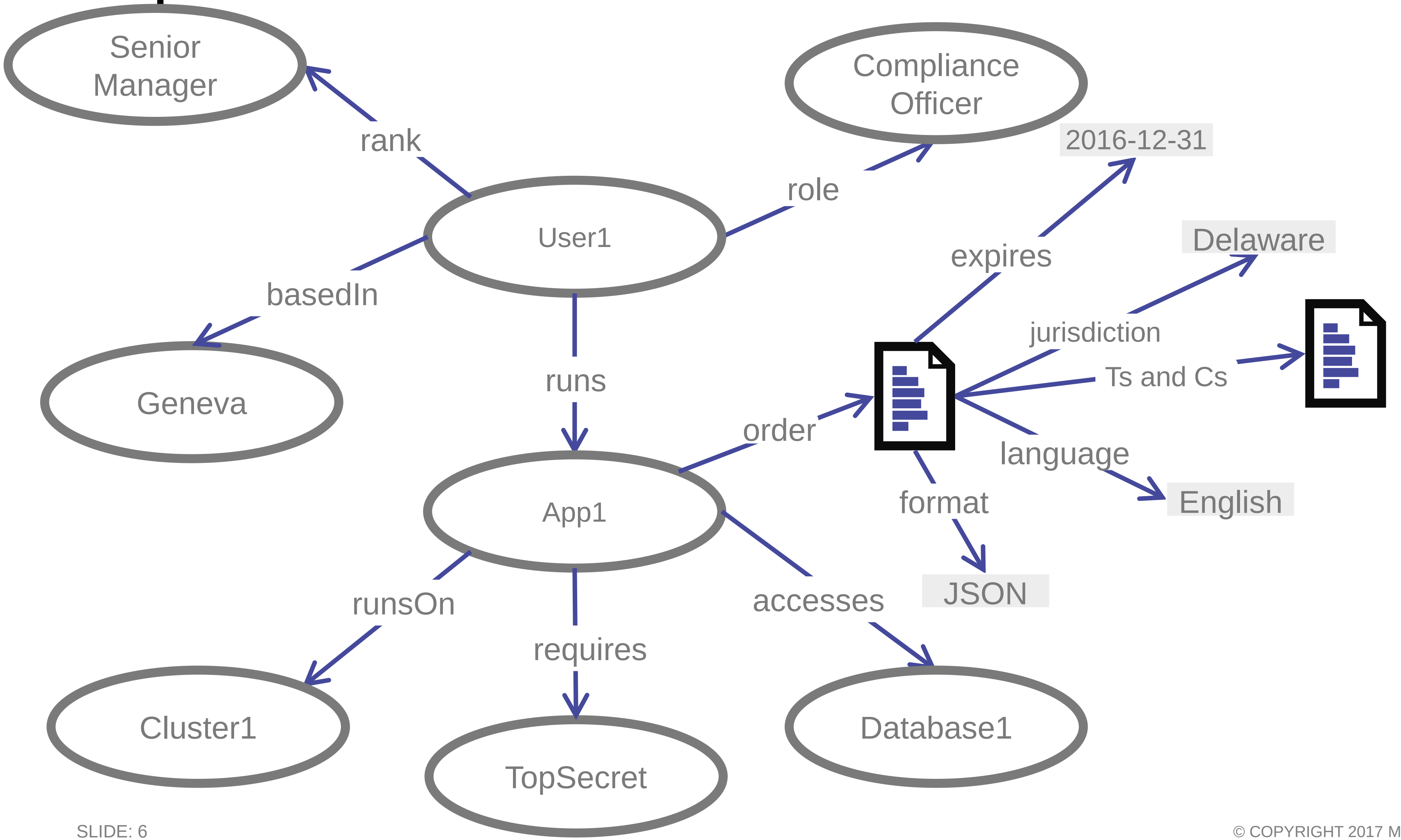
TopSecret

Database1

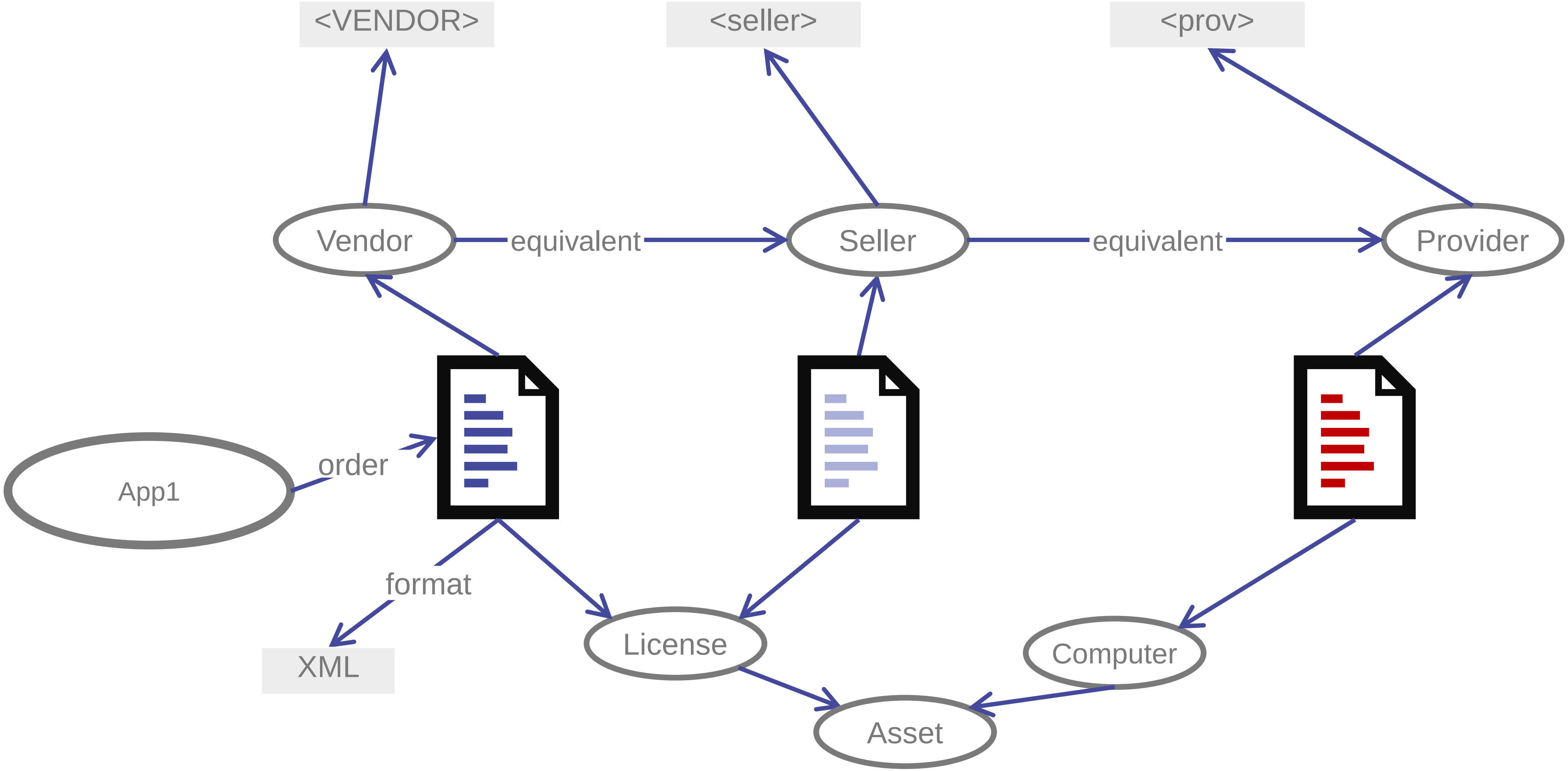
# Documents as Part of the Graph



# Triples About Documents – Extended Metadata



# Triples About Documents - Integration



# XML [JSON] Document With Embedded Triples

```
 <order id="12345">
   <VENDOR>Acme Corp</VENDOR>
   <payment>
     <amount>3427</amount>
     <unit>USD</units>
     <period>annual</period>
   </payment>
   <sem:triple>
      <sem:subject>http://youruri.com/orders/12345</sem:subject>
       <sem:predicate>http://youruri.com/predicates/expires</sem:predicate>
      <sem:object>2016-12-31</sem:object>
   </sem:triple>
   <sem:triple>
      <sem:subject>http://youruri.com/orders/12345</sem:subject>
       <sem:predicate>http://youruri.com/predicates/TsAndCs</sem:predicate>
      <sem:object>http://youruri.com/terms/34567</sem:object>
   </sem:triple>
 <description> .... </description>
</order>
```

# Set of Triples with XML [JSON] annotation

```
<userInfo>
    <source>myApp44</source>
    <confidence>100</confidence>
    <location>37.52 -122.25</location>
    <icd9-proc-code>1111</icd9-proc-code>
    <temporal>
      <systemStart/><systemEnd/>
      <validStart>2014-04-03T11:00:00</validStart>
      <validEnd>2014-04-03T16:00:00</validEnd>
    </temporal>
    ...
    <sem:triple>
      <sem:subject>http://youruri.com/users/11111</sem:subject>
      <sem:predicate>http://youruri.com/predicates/runs</sem:predicate>
      <sem:object>http://youruri.com/applications/1111</sem:object>
    </sem:triple>
    <sem:triple>
      <sem:subject>http://youruri.com/users/11111</sem:subject>
      <sem:predicate>http://youruri.com/predicates/manages</sem:predicate>
      <sem:object>http://youruri.com/applications/3333</sem:object>
    </sem:triple>
</userInfo>
```

# Semantics Performance at Scale

CREDITS

Balvinder Dang
Ed Thomas
James Kerr
Tom Ternquist

# Semantics Performance at Scale

- Note: not all Semantics use cases are “at scale”
- Some uses require only a small number of triples with simple queries:
  - Semantic Search – expand search terms, concept search
  - Semantic Integration – expand location of search via a Semantic Model
- Here, we can be more generous with joins, inference, and so on

# Tip: Use MarkLogic!

- Use MarkLogic capabilities:
  - Security
  - Partition: e.g. Collections
  - Search
    - Filtering
    - Projection

- DON’T: pull all possibly-relevant data into the mid-tier
- DO: use the power of MarkLogic
  - distributed operations / Map-Reduce
  - close to the data

# Tip: Scope the query

- Trim results sets early

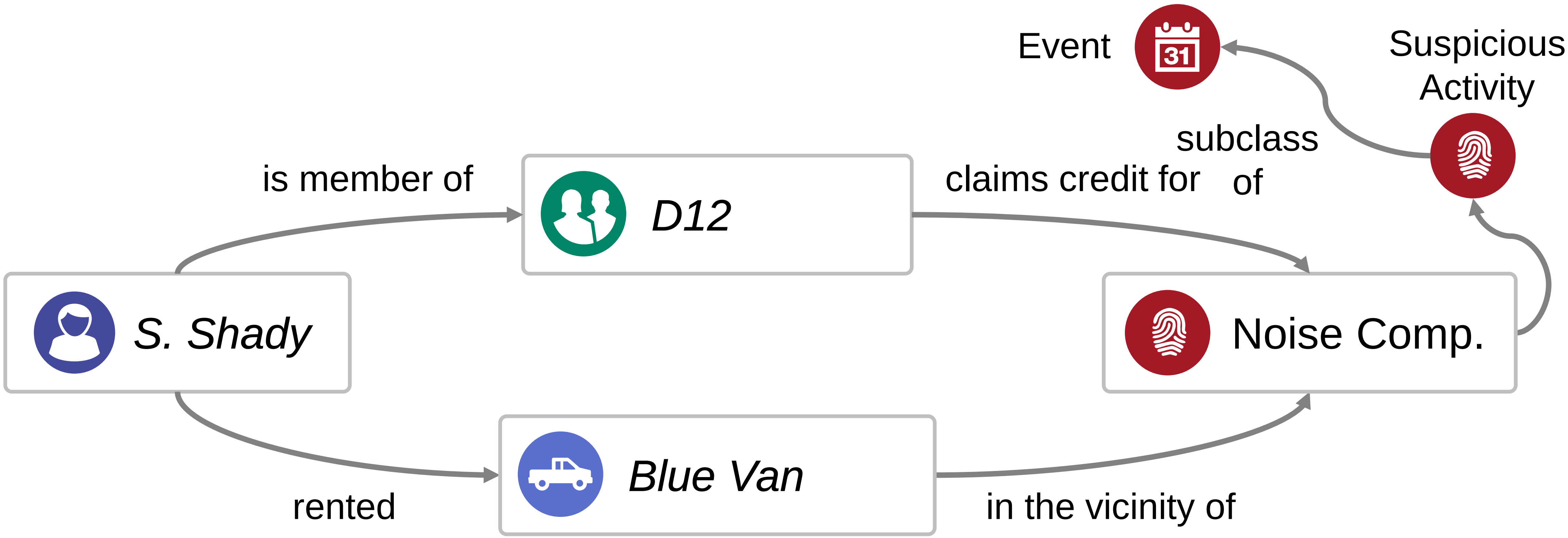
- Compare with SELECT * FROM [table]
- It’s worse than that with a triple store or document store
  - SELECT * FROM [the whole database]

- See “Use MarkLogic”: partitioning
- Advanced: get smart about keeping like-triples in the same document

# Tip: Documents for Entities; Triples for Facts, Relationships

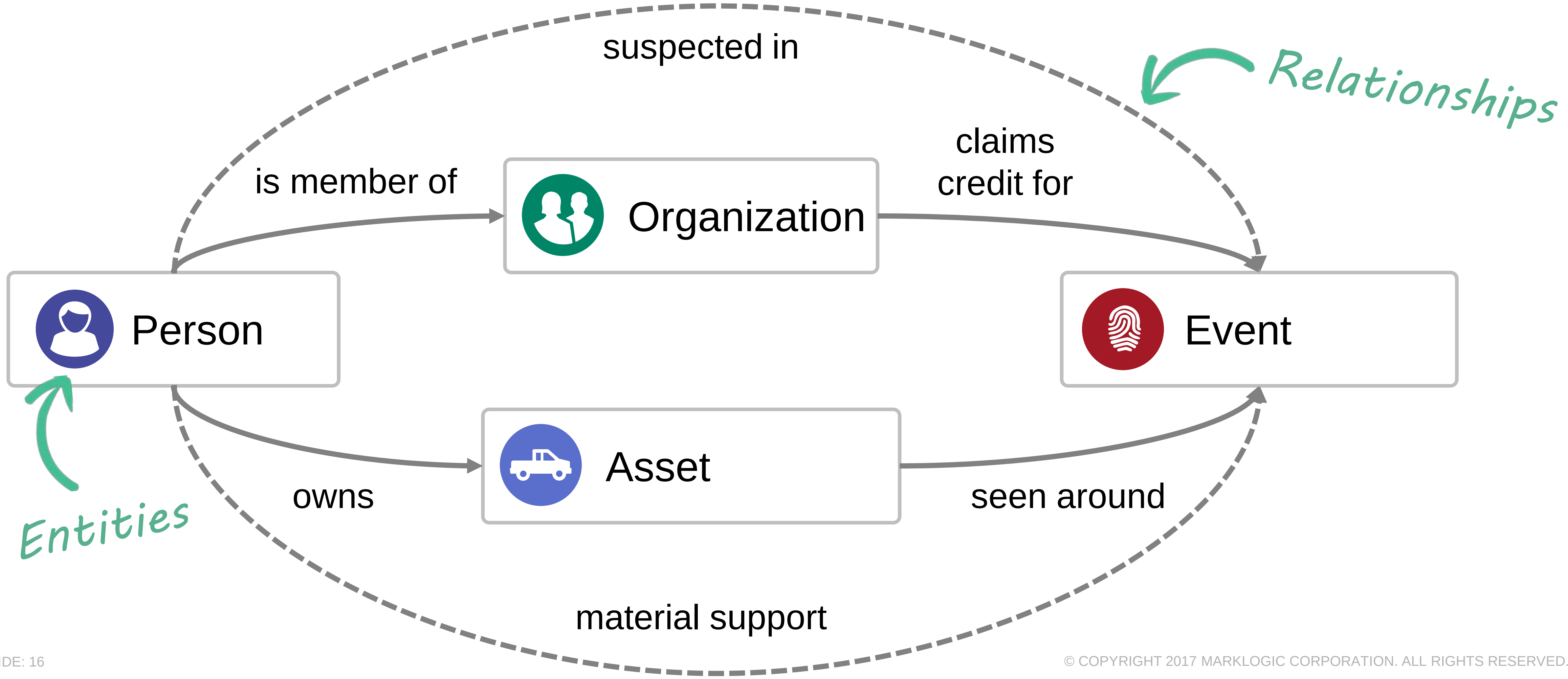
- It’s better that way!
- It’s more efficient
  - Keep entity information together
  - Reduce joins on query AND retrieval
- … some clever tricks if you know where/how triples are stored

# Tip: Documents for Entities; Triples for Facts, Relationships



MarkLogic

# Tip: Documents for Entities; Triples for Facts, Relationships



# Tip: Inference

Inference is *powerful* and *convenient*, but can be *expensive*

- Scope the query
- Consider SPARQL-based inference

# SPARQL Based Inference

- Inference ruleset:
  - Automatically expands rdf:type

- SPARQL Based Inference:
  - Use a property path query

```
//With Inference ruleset
sem.sparql("
  SELECT ?uri  {
    ?s rdf:type fs:Trade .
    ?s rdfs:isDefinedBy ?uri
  } LIMIT 10",
  [],[],
  sem.rulesetStore(
    "subClassOf.rules",sem.store())
  )
```

```
//With SPARQL Based Inference
sem.sparql("
  SELECT ?uri  {
    ?s rdf:type/rdfs:subClassOf* fs:Trade.
    ?s rdfs:isDefinedBy ?uri
} LIMIT 10");
```

# Tip: Inference

Inference is *powerful* and *convenient*, but can be *expensive*

- Scope the query
- Consider SPARQL-based inference
- Consider materialization (SPARQL CONSTRUCT)

# Tips: detailed [1]

- Use MarkLogic indexes to scope a query
  - Collection query (or SPARQL FROM) to partition the RDF space
  - Put ontologies and other lookup/mapping triples into their own graphs/collections
  - Consider pushing-down some SPARQL FILTERs to the document
- Look for joins that can be eliminated by materializing those relationships at load/update time
  - Think of this as *denormalizing* triples
  - “joins are free … conceptually”
- “Materialize” often-queried elements (in documents) consistently for better indexing

# Tips: detailed [2]

- Project the result:
  - From SPARQL for small results sets:
    - Get the set of documents that match your query using search
    - Return the relevant triples *directly from the index*
  - From documents for large results sets:
    - “get me customers/orders/contracts that …”
    - Fetch document in a single read, no joins

# Tips: detailed [3]

- Use the latest version of MarkLogic: perf improvements on minor releases
- Add more memory: allows the optimizer to choose faster plans
- Add more hardware: cluster for parallelization
- Re-use queries: query plan is cached for 5 minutes; use bind variables
- Use MarkLogic built-in functions in SPARQL
- Consider dedup-off option to sem:sparql() [ML9]
    - Avoid dedup processing
    - No effect on results if you have no duplicate triples and/or you use DISTINCT
    - Can make a big difference

# Case Studies

- Company names have been obscured, but these are real projects
- Query timings are given for comparison only

MarkLogic

# Case Study:
# Educational Publisher

# Case Study: Educational Publisher

Central metadata repository to store metadata, product mapping and central rights management using all-RDF

- Semantic enrichment of content: provide bespoke products using intelligent/smart search.
- Easy discovery and re-use of content
- Central rights management
- Use (and extend) standard RDF vocabularies to share metadata, e.g. Dublin Core.
- RDF Multilingual support

- Before: some SPARQL queries were very slow
- Resolution: 4-week exercise to identify and improve slowest queries
- After: performance improvements of **up to 100x**

# Query: find triples where object matches mat?s

Dataset: 6 Million triples

Query: find triples where object matches mat?s

- Regex term filter
- Language filter
- Searchable filter
- UNION Blank Nodes
- Authorization filter based on SHACL

Regex Term Filter

Language Filter

Searchable Filter

Union BNodes

Authorization Filter

```
SELECT DISTINCT ?id
WHERE
{
     ?id a ?__type.
  {{{
      ?id ?_propVar1 ?_o2.
      FILTER regex (?_o2, ”mat?s"^^<http://www.w3.org/2001/XMLSchema#string>, "i")
      FILTER (langmatches(lang(?_o2), "en") || lang(?_o2) = "" )
      FILTER NOT EXISTS { graph <nonsearchable> {?id ?_propVar1 ?obj } } }
    } UNION {
      ?id ?bnodeProp _:b0 .
      _:b0 ?_propVar1 ?_o2.
      FILTER regex (?_o2, ”mat?s"^^<http://www.w3.org/2001/XMLSchema#string>, "i")
      FILTER (langmatches(lang(?_o2), "en") || lang(?_o2) = "" )
      FILTER NOT EXISTS { graph <nonsearchable> {_:b0 ?_propVar1 ?obj } }}
  }
  OPTIONAL { ?id  raf:retrievableBy ?__irole}
  FILTER(!BOUND(?__irole) || ?__irole IN ("metadataReader"))
  OPTIONAL {
        SELECT ?__badShape ?__type {
          ?__badShape sh:scopeClass ?__type.
          MINUS {
            ?__badShape sh:scopeClass ?__type.
            ?__badShape raf:retrievableBy ?userRole
            VALUES ?userRole {"metadataReader"}
          }
          MINUS {
            ?__badShape   sh:scopeClass ?__type.
            ?__badShape raf:sorRole ?sorRole
          }}
}  FILTER (!BOUND(?__badShape))
}
LIMIT 20
```

# Query: find triples where object matches mat?s

Dataset: 6 Million triples

Query: find triples where object matches mat?s

**Timings**:

- Initial: **20 secs**
- Use cts:contains instead of regex() in SPARQL: **7 secs**
- Use collection query to partition by collections/graphs: **3 secs**
- Use a cts:query to partition data further: **0.4 secs**
- Overall improvement: **100x**

# Query: find triples where object matches mat?s

Dataset: 6 Million triples

Query: find triples where object matches mat?s

**Next steps**:

- Replace UNION Blank Nodes with property path (new in MarkLogic 8)
- Look at using MarkLogic security (index-based)
  - Replace SHACL constraints in each query
  - Remove `FILTER NOT EXISTS { graph <nonsearchable> …`

# Query: GET Description

Dataset: 6 Million triples

Query: Fetch everything you know about X

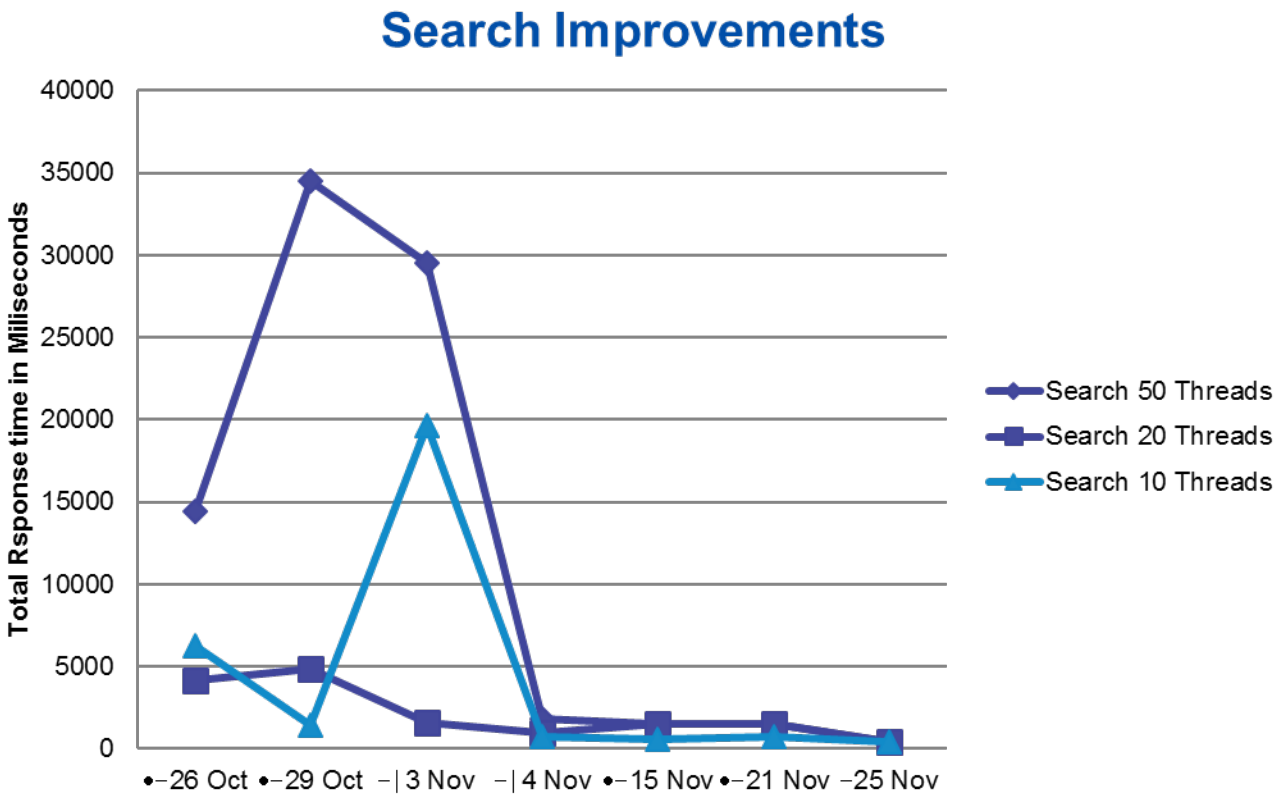
**Timings**:

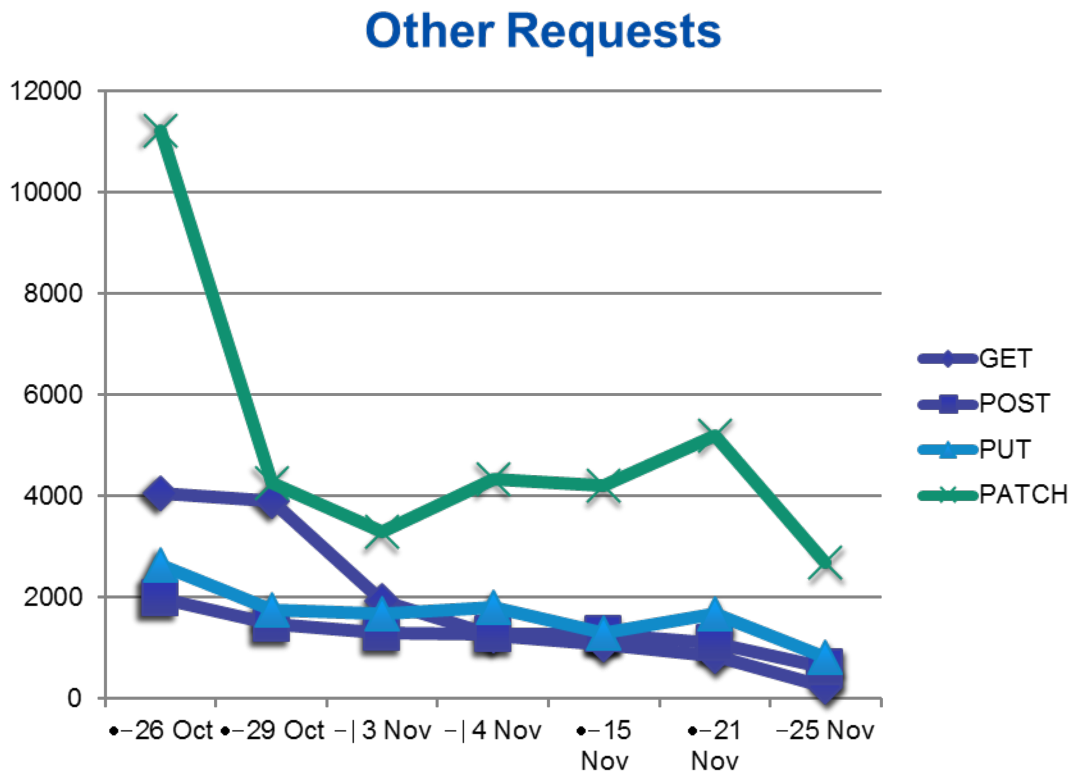
- Initial: **6 secs**
- Use named graph/collection and collection-lexicon
- Use cts:triple-range-query to scope by subjects
- Final: **0.2 secs**
- Overall improvement: **30x**

**Next step**:

- Consider documents for “everything you know about X”
- Use TDE to index (parts of) documents as triples [ML9]

# Performance Exercise





MarkLogic

# Case Study:
# Data Store for Clinical Data

# Case Study: Data Store for Clinical Data

Data store for clinical data - Organizations, patients, encounters, conditions, medications, etc.

- Data comes in in the form of FHIR messages
- Need to track provenance of every data element in the system
- Enrich clinical data
- Need to be able to query based on one or more ontologies
  - Connect concepts, traverse relationships, etc.
- Role-based access to PHI
- Encrypt and audit access of PHI
- Archive older, less used data

# Initial Approach

- All data represented as triples
  - Limited access to document features: security, tiered storage, bitemporal
- Provenance and other metadata about triples via “instantiated predicates”
  - One way to do “reification” in a pure-triples world
  - Every query requires inference OR re-write

# Initial Approach

- All data represented as triples
- Provenance and other metadata about triples via “instantiated predicates”

Issues:

- Inferencing required for every query against the ADR
- Limited use of MarkLogic’s powerful multi-model query and analytic capabilities
- Limited use of MarkLogic’s data management capabilities
  - Security, Tiered Storage, Bitemporal
- Many joins to retrieve back a single resource/record (e.g. Patient)
- Complex ETL from entities to triples

# <Sidetrack: Metadata about triples>

# Metadata about a triple[1]: Reification

- Triple:
  `:John :livesIn :London`
- Reified triple:
  ```
  :triple1234 rdf:type rdf:Statement .
  :triple1234 rdf:subject :John
  :triple1234 rdf:predicate :livesIn
  :triple1234 rdf:object :London
  ```
- Now you can say things about this triple:
  `:triple1234 :source :patient-record-42`
- Downside:
  - 4x triples
  - more complex queries OR inference

# Metadata about a triple[2]: Instantiated Predicates

- Triple:
  ```
  :John :livesIn :London
  ```
- Instantiated Predicate:
  ```
  :JohnLivesIn rdfs:type :livesIn
  :John :JohnLivesIn :London
  ```
- Now you can say things about this triple:
  ```
  :JohnLivesIn :source :patient-record-42
  ```
  - (*need at least 1 more triple to constrain :JohnLivesIn to :John*)
- Downside:
  - At least 3x triples
  - More complex queries OR inference
  - Inference: infer around 100,000 new triples for each query

# Metadata about a triple[3]: Embed triple in a document

- Triple:

```
:John :livesIn :London
```

- Now you can say things about this triple:

```
<doc>
  <triple>
    <subject>John</subject>
    <predicate>livesIn</predicate>
    <object>London</object>
  </triple>
  <source>patient-record-42</source>
</doc>
```

- Embedded triple:

```
<doc>
  <triple>
    <subject>John</subject>
    <predicate>livesIn</predicate>
    <object>London</object>
  </triple>
</doc>
```

Any metadata: source, confidence, bitemporal, etc
Metadata can be structured
Query with combination query or Optic

# </Sidetrack: Metadata about triples>

# Initial Approach

- All data represented as triples
- Provenance and other metadata about triples via “instantiated predicates”

Issues:

- Inferencing required for every query against the ADR
- Limited use of MarkLogic’s powerful multi-model query and analytic capabilities
- Limited use of MarkLogic’s data management capabilities
  - Security, Tiered storage, Bitemporal
- Many joins to retrieve back a single resource/record (e.g. Patient)
- Complex ETL from entities to triples

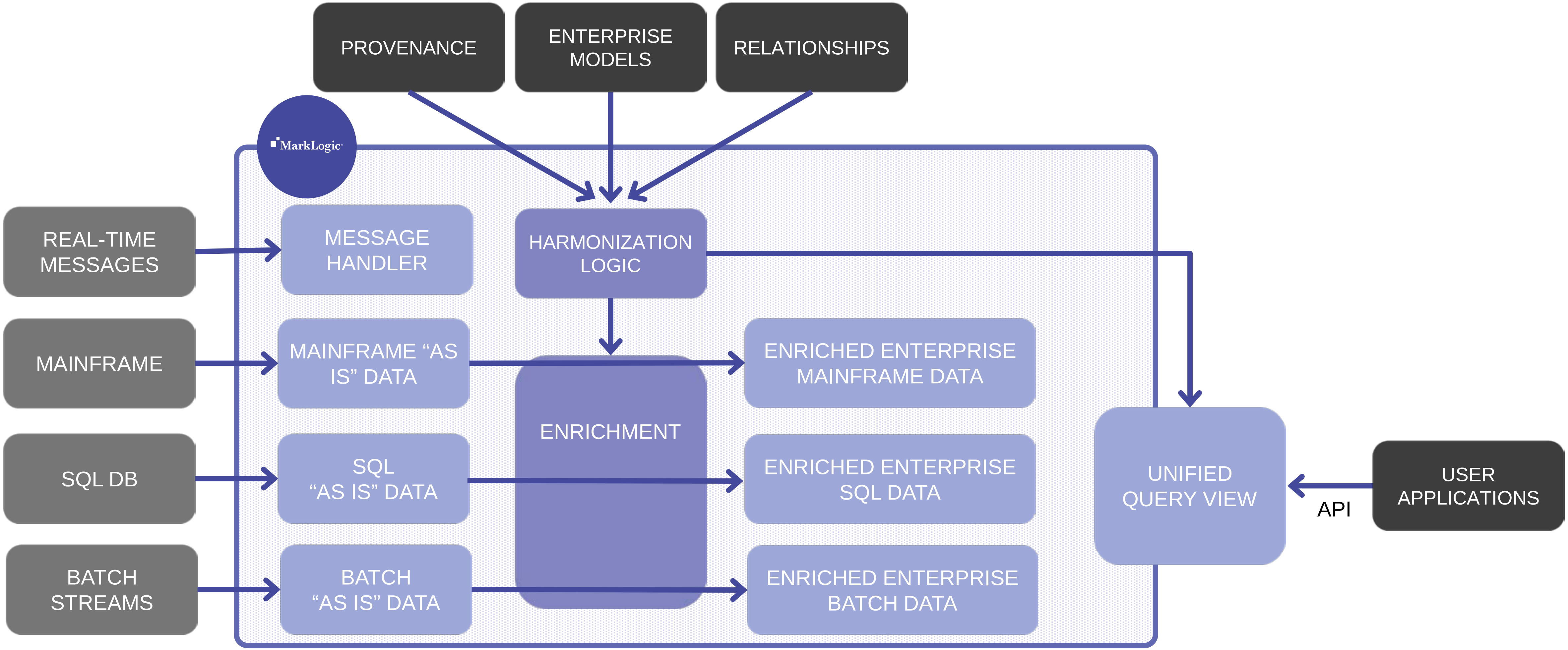
# New Approach

Documents plus triples:

- Store incoming messages as documents
- Map data elements into top-level domain **entities** (Patient, Practitioner, Encounter, etc.) and store these as documents
- Store the triples that go with each entity in the entity document (share management)
- Store enriched / derived triples in the entity documents that they came from
- Capture provenance in XML data structures
- Leverage bitemporal to track changes to entities over time

HEALTHCARE 360

# Architecture



MarkLogic

# Unified Query View example for Patient ID

| Subject | Predicate | Object | Comment |
|---|---|---|---|
| EntPatientID | owl:sameAs | …/recordTarget/patientClinical/id | HL7 v3 patient ID |
| EntPatientID | owl:sameAs | PID-3 | HL7 v2 patient ID |
| EntPatientID | owl:sameAs | .../entry/resource/Patient/id | FHIR patient ID |

- EntPatientID is the enterprise patient ID that the API exposes in queries
- The patient ID is labeled and located differently in HL7 v3, HL7 v2 and FHIR
- Ontology triples can be used to expand the search to all the known ID locations in the combined sources

# Provenance

- Each entity has metadata / provenance / bitemporal information

```
<envelope>
  <metadata>
    <lastUpdatedDateTime>2015-05-25T12:00:03Z</lastUpdatedDateTime>
    <firstCreatedDateTime>2015-05-25T12:00:03Z</firstCreatedDateTime>
    <source>/fhir/message-9876.xml</source>
    <lastUpdatedDateTime>2015-05-25T12:00:03Z</lastUpdatedDateTime>
    <firstCreatedDateTime>2015-05-25T12:00:03Z</firstCreatedDateTime>
  </metadata>
  <original>
    ...
   </original>
  </envelope>
```

- Note: Prior versions of entities are kept via temporal collection

MarkLogic

# Query

- Combination: MarkLogic **CTS** queries to scope **SPARQL** and **Inference**
  - Reduce the set of entities we are interested in by using CTS
  - Use SPARQL to query across entities and concepts
  - Retrieve records as single entity documents without the need for joins
- **SPARQL** for semantic search
  - Query triples (an ontology) to expand a concept search to include related concepts
- **SPARQL** for integration
  - Query triples (an ontology) to expand a search over a canonical patientID
    to a search over all representations of patientID

# Query

- Combiantion: MarkLogic **CTS** queries to scope **SPARQL** and **Inference**
  - Reduce the set of entities we are int
  - Use SPARQL to query across entitie
  - Retrieve records as single entity documents without the need for joins
- **SPARQL** for semantic search
  - Query triples (an ontology) to expand a concept search to include related concepts
- **SPARQL** for integration
  - Query triples (an ontology) to expand a search over a canonical patientID to a search over all representations of patientID

Query expansion:

- Expand the value you’re querying for
- Expand the places you look for that value

# Example Combination Query

- Find **patients** who have **encounters** of type “outpatient” and date after 1/1/2010 with a **physician** with last name of “ROBERT”

Done in one call to the DB

- CTS query to select encounters with type “outpatient” and date after 1/1/2010 and physicians with last name of “ROBERT”
- SPARQL against the triples from the selected entities to join physicians and encounters and return patient IRIs (as contained in the encounters)

Efficient Read

- Use the patient IRIs to retrieve complete or partial patient entities (one by one or in batches)

MarkLogic

# Query: find patient info for some condition code

Dataset: 300 MN triples => (30K documents + 22 MN triples)

Query: find patient info for some condition code

**Timings**:

- Initial, triples-only: **45 secs**
- Triples-only with re-written queries and some materialization: **30 secs**
- Triples + documents, SPARQL with CTS constraints: **0.7 secs**
- overall improvement: **65x perf + bitemporal**

# Query: Aggregate patients by condition code, gender

Dataset: 300 MN triples => (30K documents + 22 MN triples)

Query: Aggregate patients by condition code, gender

**Timings**:

- Initial, triples-only: **1200 secs (8)**
- Triples-only with re-written queries and some materialization: **50 secs**
- Triples + documents, SPARQL with CTS constraints: **0.9 secs**
- overall improvement: **55x perf + bitemporal**

# Case Study:
# Leading Global Bank

# Case Study: Leading Global Bank

Inventory system for tracking technical assets

- 2 Billion triples
- Queries are heavily graph-traversal
- Documents for triples enrichment, provenance
- Bitemporal audit trail

# Performance Summary

- Documents: entities
- Triples: relationships, facts, graphs
- You don’t have to choose just one!
- They go together like ….

MarkLogic

# Understanding SPARQL Execution

# Well-Behaved Query

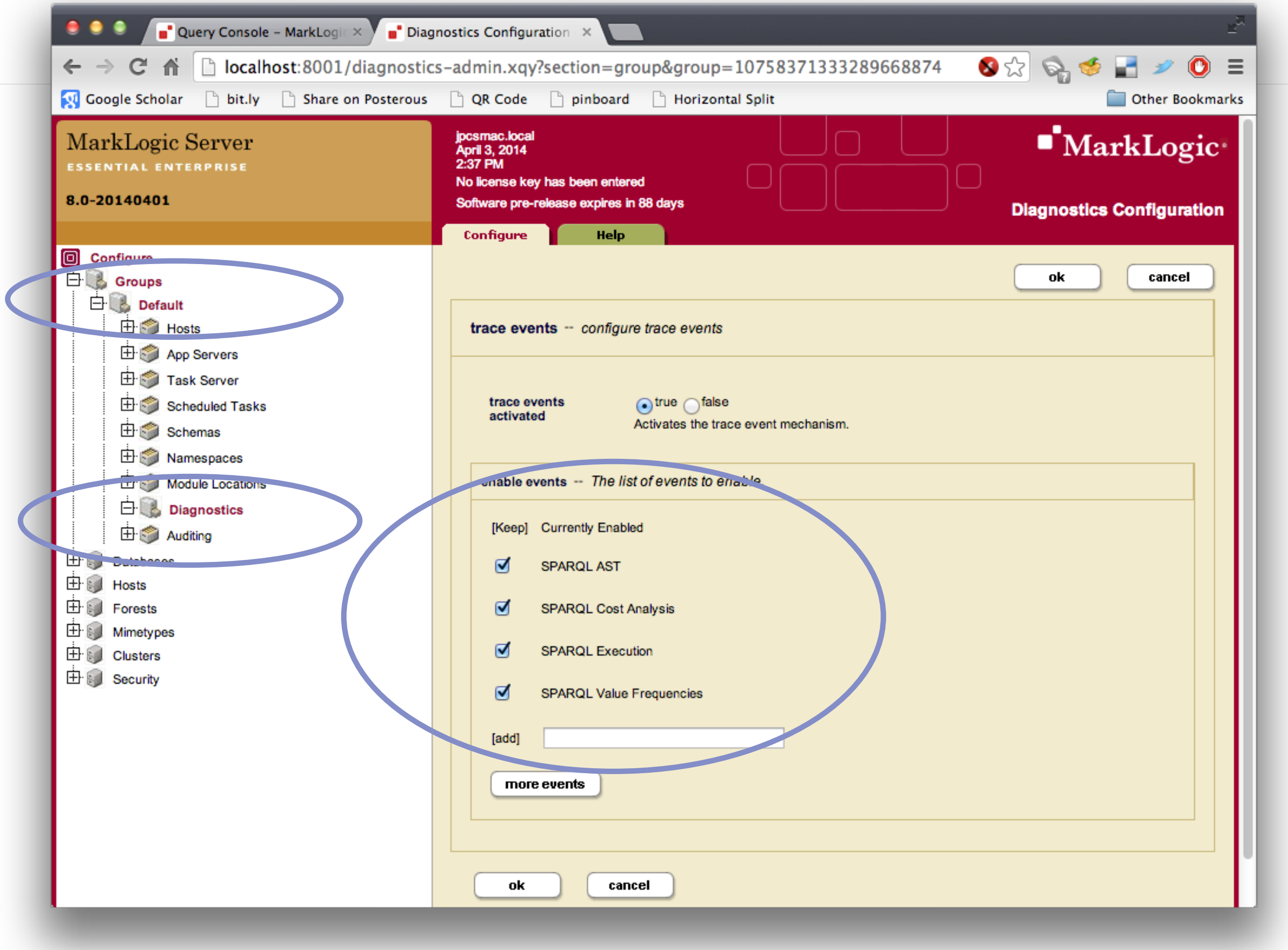
```
prefix : <http://example.org/kennedy/>
select * {
  ?person :first-name ?first .
  ?person :last-name ?last .
  ?person :birth-place [:name ?birthPlace] .
  filter(?birthPlace = 'Wien')
}
order by ?first ?last
```

# Executing a Query

1. Parse
2. Initial query plan
3. Cost-based optimization
4. Execution plan
5. Run the plan

# Query Plan

- Trace flag "SPARQL AST" [ML7,8,9]
- Trace flag "Optic Plan" [ML9]
- Trace option ("trace=XXXX") to sem:sparql() and xdmp:sql() [ML9]
- User friendly query plan functions: sem:sparql-plan(), and xdmp:sql-plan() [ML9.0-2]



jpcsmac.local
April 3, 2014
2:37 PM
No license key has been entered
Software pre-release expires in 88 days

**trace events** -- *configure trace events*

**trace events activated** ● true ○ false
Activates the trace event mechanism.

**enable events** -- *The list of events to enable*

[Keep] Currently Enabled

☑ SPARQL AST

☑ SPARQL Cost Analysis

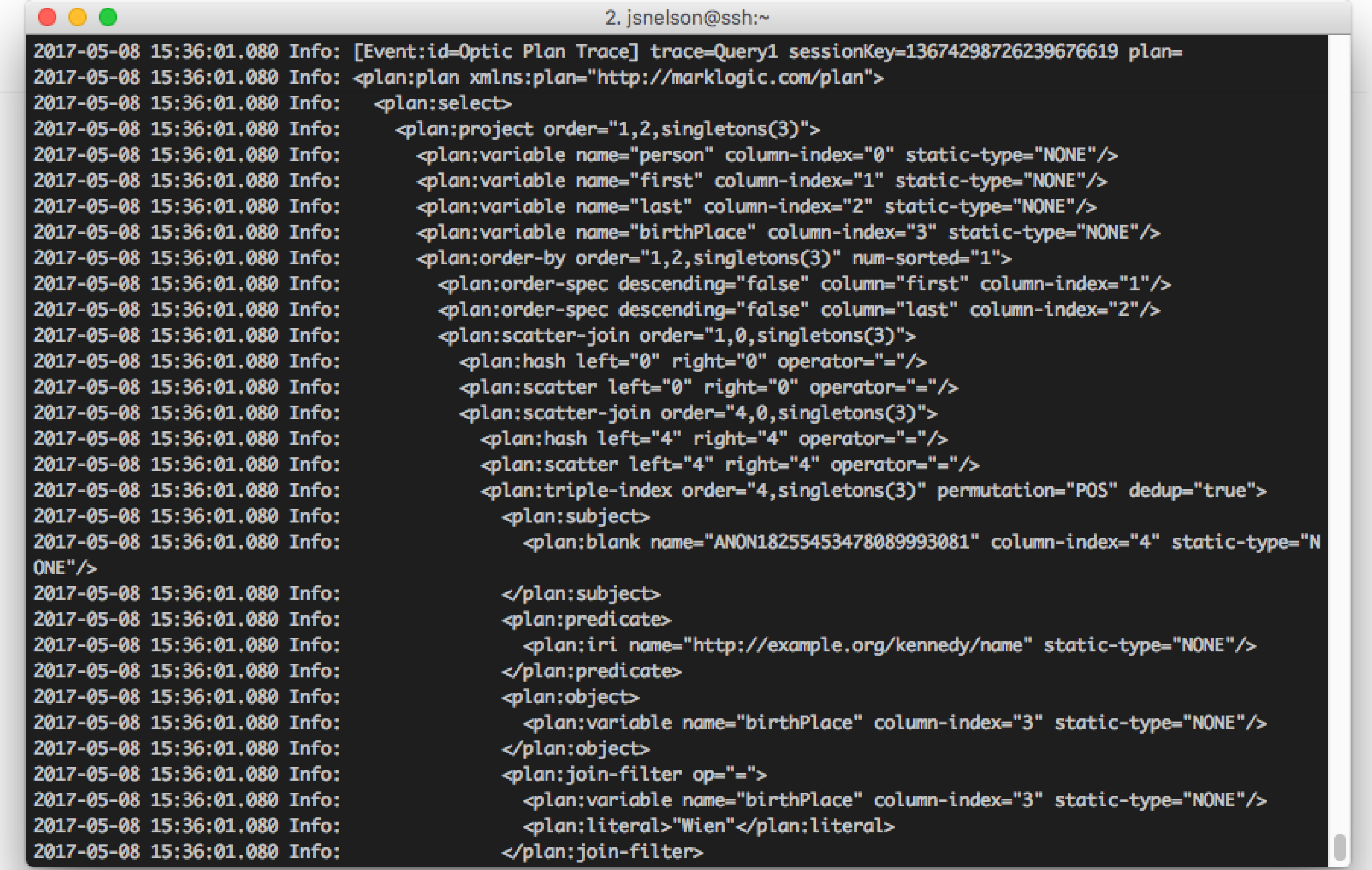
☑ SPARQL Execution

☑ SPARQL Value Frequencies

[add]

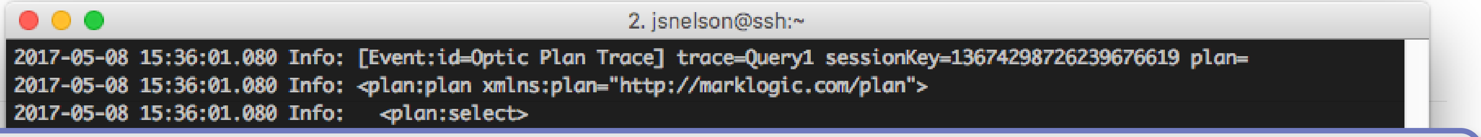
# Query Plan

- Trace flag "SPARQL AST" [ML7,8,9]
- Trace flag "Optic Plan" [ML9]
- Trace option ("trace=XXXX") to sem:sparql() and xdmp:sql() [ML9]
- User friendly query plan functions: sem:sparql-plan(), and xdmp:sql-plan() [ML9]



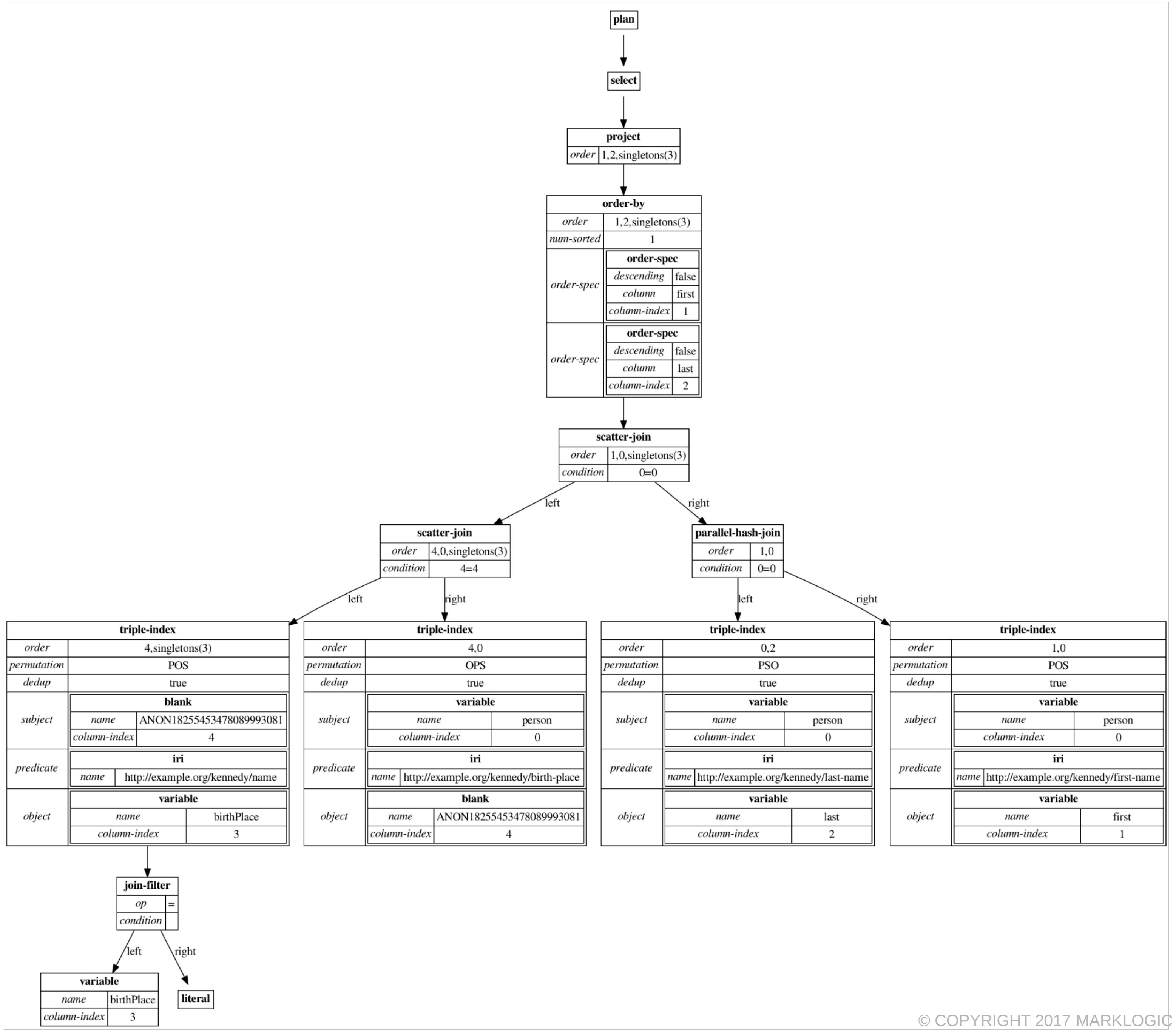
```
2017-05-08 15:36:01.080 Info: [Event:id=Optic Plan Trace] trace=Query1 sessionKey=13674298726239676619 plan=
2017-05-08 15:36:01.080 Info: <plan:plan xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.080 Info:   <plan:select>
2017-05-08 15:36:01.080 Info:     <plan:project order="1,2,singletons(3)">
2017-05-08 15:36:01.080 Info:       <plan:variable name="person" column-index="0" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:       <plan:variable name="first" column-index="1" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:       <plan:variable name="last" column-index="2" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:       <plan:variable name="birthPlace" column-index="3" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:       <plan:order-by order="1,2,singletons(3)" num-sorted="1">
2017-05-08 15:36:01.080 Info:         <plan:order-spec descending="false" column="first" column-index="1"/>
2017-05-08 15:36:01.080 Info:         <plan:order-spec descending="false" column="last" column-index="2"/>
2017-05-08 15:36:01.080 Info:         <plan:scatter-join order="1,0,singletons(3)">
2017-05-08 15:36:01.080 Info:           <plan:hash left="0" right="0" operator="="/>
2017-05-08 15:36:01.080 Info:           <plan:scatter left="0" right="0" operator="="/>
2017-05-08 15:36:01.080 Info:           <plan:scatter-join order="4,0,singletons(3)">
2017-05-08 15:36:01.080 Info:             <plan:hash left="4" right="4" operator="="/>
2017-05-08 15:36:01.080 Info:             <plan:scatter left="4" right="4" operator="="/>
2017-05-08 15:36:01.080 Info:             <plan:triple-index order="4,singletons(3)" permutation="POS" dedup="true">
2017-05-08 15:36:01.080 Info:               <plan:subject>
2017-05-08 15:36:01.080 Info:                 <plan:blank name="ANON18255453478089993081" column-index="4" static-type="N
ONE"/>
2017-05-08 15:36:01.080 Info:               </plan:subject>
2017-05-08 15:36:01.080 Info:               <plan:predicate>
2017-05-08 15:36:01.080 Info:                 <plan:iri name="http://example.org/kennedy/name" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:               </plan:predicate>
2017-05-08 15:36:01.080 Info:               <plan:object>
2017-05-08 15:36:01.080 Info:                 <plan:variable name="birthPlace" column-index="3" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:               </plan:object>
2017-05-08 15:36:01.080 Info:               <plan:join-filter op="=">
2017-05-08 15:36:01.080 Info:                 <plan:variable name="birthPlace" column-index="3" static-type="NONE"/>
2017-05-08 15:36:01.080 Info:                 <plan:literal>"Wien"</plan:literal>
2017-05-08 15:36:01.080 Info:               </plan:join-filter>
```

# Query Plan



```
2017-05-08 15:36:01.080 Info: [Event:id=Optic Plan Trace] trace=Query1 sessionKey=13674298726239676619 plan=
2017-05-08 15:36:01.080 Info: <plan:plan xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.080 Info:   <plan:select>
```

```
<plan:plan xmlns:plan="http://marklogic.com/plan">
  <plan:select>
    <plan:project order="1,2,singletons(3)">
      <plan:variable name="person" column-index="0" static-type="NONE"/>
      <plan:variable name="first" column-index="1" static-type="NONE"/>
      <plan:variable name="last" column-index="2" static-type="NONE"/>
      <plan:variable name="birthPlace" column-index="3" static-type="NONE"/>
      <plan:order-by order="1,2,singletons(3)" num-sorted="1">
        <plan:order-spec descending="false" column="first" column-index="1"/>
        <plan:order-spec descending="false" column="last" column-index="2"/>
        <plan:scatter-join order="1,0,singletons(3)">
          <plan:hash left="0" right="0" operator="="/>
          <plan:scatter left="0" right="0" operator="="/>
          <plan:scatter-join order="4,0,singletons(3)">
            <plan:hash left="4" right="4" operator="="/>
            <plan:scatter left="4" right="4" operator="="/>
            <plan:triple-index order="4,singletons(3)" permutation="POS" dedup="true">
              <plan:subject><plan:blank name="ANON18255453478089993081" column-index="4" static-type="NONE"/></plan:subject>
              <plan:predicate><plan:iri name="http://example.org/kennedy/name" static-type="NONE"/></plan:predicate>
              <plan:object><plan:variable name="birthPlace" column-index="3" static-type="NONE"/></plan:object>
              <plan:join-filter op="=">
                <plan:variable name="birthPlace" column-index="3" static-type="NONE"/>
                <plan:literal>"Wien"</plan:literal>
              </plan:join-filter>
            </plan:triple-index>
            <plan:triple-index order="4,0" permutation="OPS" dedup="true">
             …
```



| project | |
|---|---|
| *order* | 1,2,singletons(3) |

| order-by | |
|---|---|
| *order* | 1,2,singletons(3) |
| *num-sorted* | 1 |
| *order-spec* | **order-spec**<br>*descending* false<br>*column* first<br>*column-index* 1 |
| *order-spec* | **order-spec**<br>*descending* false<br>*column* last<br>*column-index* 2 |

| scatter-join | |
|---|---|
| *order* | 1,0,singletons(3) |
| *condition* | 0=0 |

| scatter-join | |
|---|---|
| *order* | 4,0,singletons(3) |
| *condition* | 4=4 |

| parallel-hash-join | |
|---|---|
| *order* | 1,0 |
| *condition* | 0=0 |

| triple-index | |
|---|---|
| *order* | 4,singletons(3) |
| *permutation* | POS |
| *dedup* | true |
| *subject* | **blank**<br>*name* ANON18255453478089993081<br>*column-index* 4 |
| *predicate* | **iri**<br>*name* http://example.org/kennedy/name |
| *object* | **variable**<br>*name* birthPlace<br>*column-index* 3 |

| triple-index | |
|---|---|
| *order* | 4,0 |
| *permutation* | OPS |
| *dedup* | true |
| *subject* | **variable**<br>*name* person<br>*column-index* 0 |
| *predicate* | **iri**<br>*name* http://example.org/kennedy/birth-place |
| *object* | **blank**<br>*name* ANON18255453478089993081<br>*column-index* 4 |

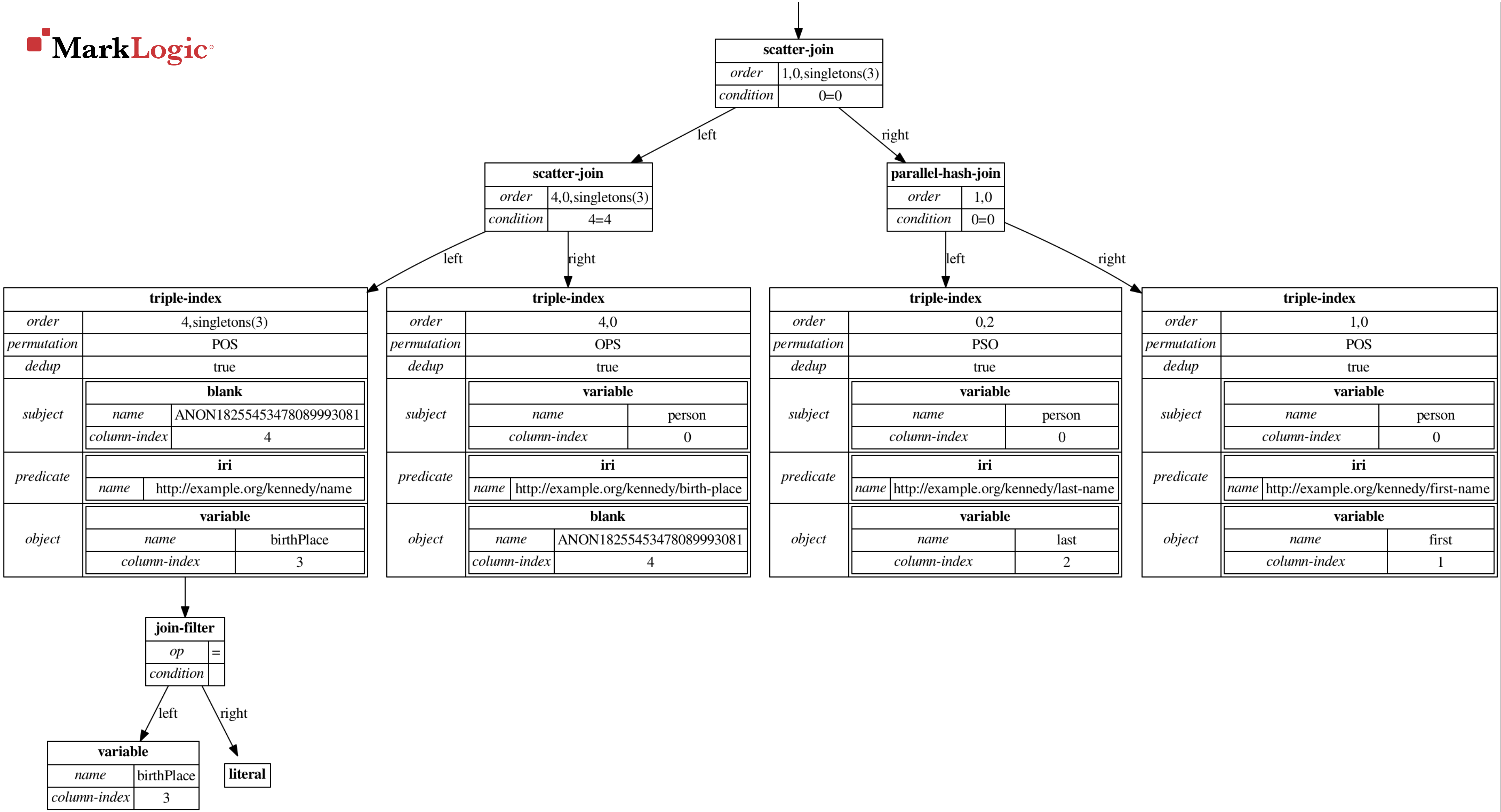
| triple-index | |
|---|---|
| *order* | 0,2 |
| *permutation* | PSO |
| *dedup* | true |
| *subject* | **variable**<br>*name* person<br>*column-index* 0 |
| *predicate* | **iri**<br>*name* http://example.org/kennedy/last-name |
| *object* | **variable**<br>*name* last<br>*column-index* 2 |

| triple-index | |
|---|---|
| *order* | 1,0 |
| *permutation* | POS |
| *dedup* | true |
| *subject* | **variable**<br>*name* person<br>*column-index* 0 |
| *predicate* | **iri**<br>*name* http://example.org/kennedy/first-name |
| *object* | **variable**<br>*name* first<br>*column-index* 1 |

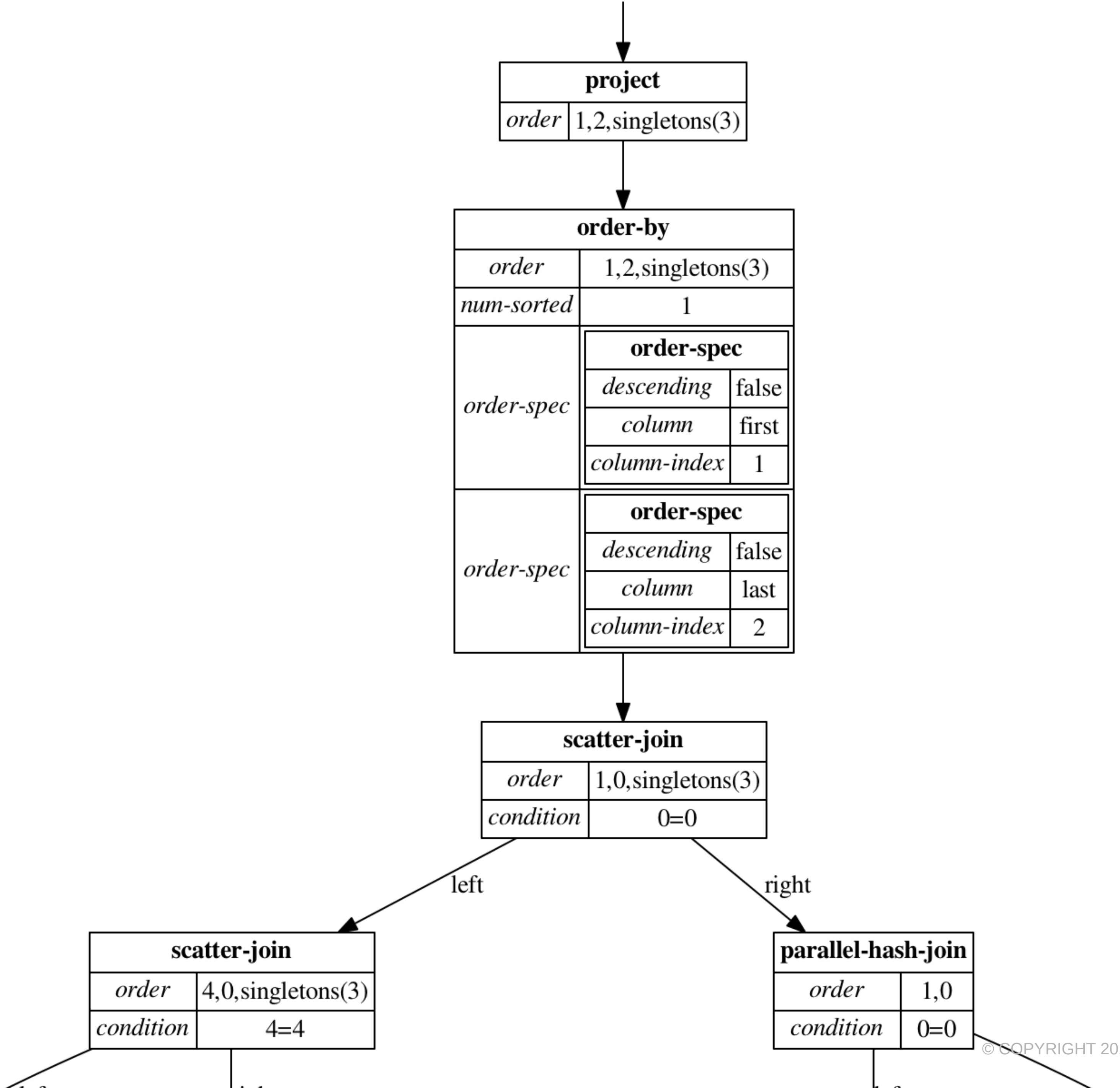
| join-filter | |
|---|---|
| *op* | = |
| *condition* | |

| variable | |
|---|---|
| *name* | birthPlace |
| *column-index* | 3 |

**literal**



- **scatter-join** — *order*: 1,0,singletons(3); *condition*: 0=0
  - left: **scatter-join** — *order*: 4,0,singletons(3); *condition*: 4=4
    - left: **triple-index** — *order*: 4,singletons(3); *permutation*: POS; *dedup*: true
      - *subject*: **blank** — *name*: ANON18255453478089993081; *column-index*: 4
      - *predicate*: **iri** — *name*: http://example.org/kennedy/name
      - *object*: **variable** — *name*: birthPlace; *column-index*: 3
      - **join-filter** — *op*: =; *condition*:
        - left: **variable** — *name*: birthPlace; *column-index*: 3
        - right: **literal**
    - right: **triple-index** — *order*: 4,0; *permutation*: OPS; *dedup*: true
      - *subject*: **variable** — *name*: person; *column-index*: 0
      - *predicate*: **iri** — *name*: http://example.org/kennedy/birth-place
      - *object*: **blank** — *name*: ANON18255453478089993081; *column-index*: 4
  - right: **parallel-hash-join** — *order*: 1,0; *condition*: 0=0
    - left: **triple-index** — *order*: 0,2; *permutation*: PSO; *dedup*: true
      - *subject*: **variable** — *name*: person; *column-index*: 0
      - *predicate*: **iri** — *name*: http://example.org/kennedy/last-name
      - *object*: **variable** — *name*: last; *column-index*: 2
    - right: **triple-index** — *order*: 1,0; *permutation*: POS; *dedup*: true
      - *subject*: **variable** — *name*: person; *column-index*: 0
      - *predicate*: **iri** — *name*: http://example.org/kennedy/first-name
      - *object*: **variable** — *name*: first; *column-index*: 1



| project | |
|---|---|
| *order* | 1,2,singletons(3) |

| order-by | |
|---|---|
| *order* | 1,2,singletons(3) |
| *num-sorted* | 1 |
| *order-spec* | **order-spec**<br>*descending* false<br>*column* first<br>*column-index* 1 |
| *order-spec* | **order-spec**<br>*descending* false<br>*column* last<br>*column-index* 2 |

| scatter-join | |
|---|---|
| *order* | 1,0,singletons(3) |
| *condition* | 0=0 |

| scatter-join | |
|---|---|
| *order* | 4,0,singletons(3) |
| *condition* | 4=4 |

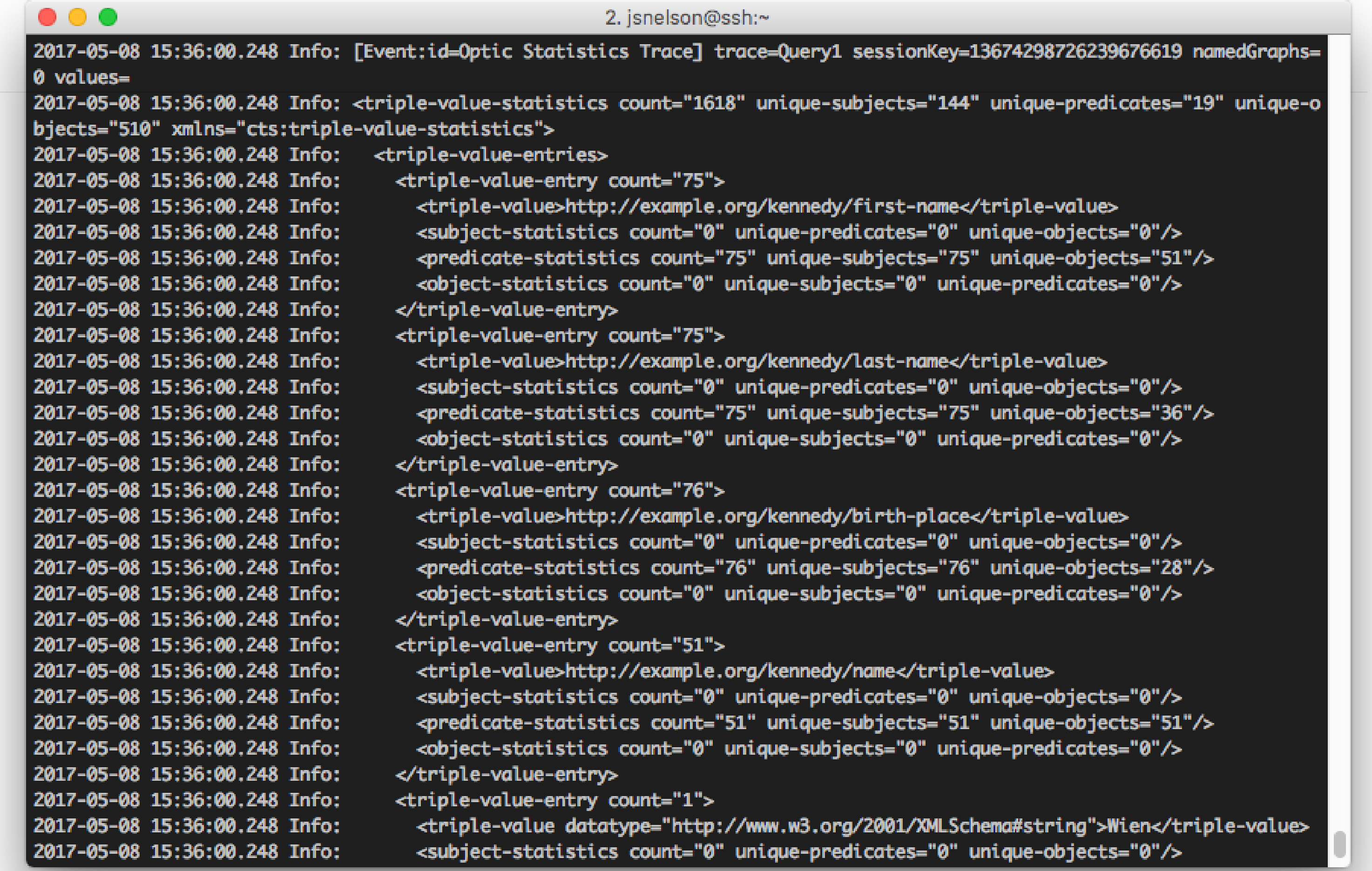
| parallel-hash-join | |
|---|---|
| *order* | 1,0 |
| *condition* | 0=0 |

# Constraining Condition

```
select * {
  ?person :first-name ?first .
  ?person :last-name ?last .
  ?person :birth-place ?p .
  ?p :name ?birthPlace .
  filter(?birthPlace = 'Wien')
}
order by ?first ?last
```

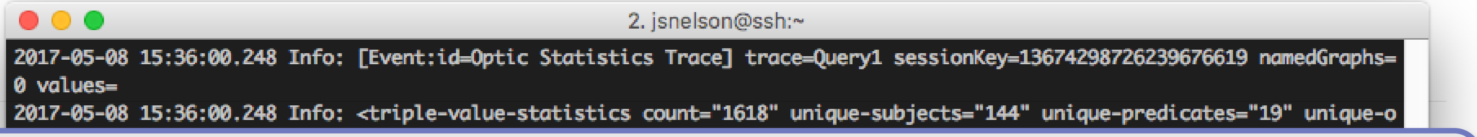
# Statistics

- Trace flag "SPARQL Value Frequencies" [ML7,8,9]
- Trace flag "Optic Statistics" [ML9]
- Trace option ("trace=XXXX") to sem:sparql() and xdmp:sql() [ML9]
- cts:triple-value-statistics() [ML8,9]

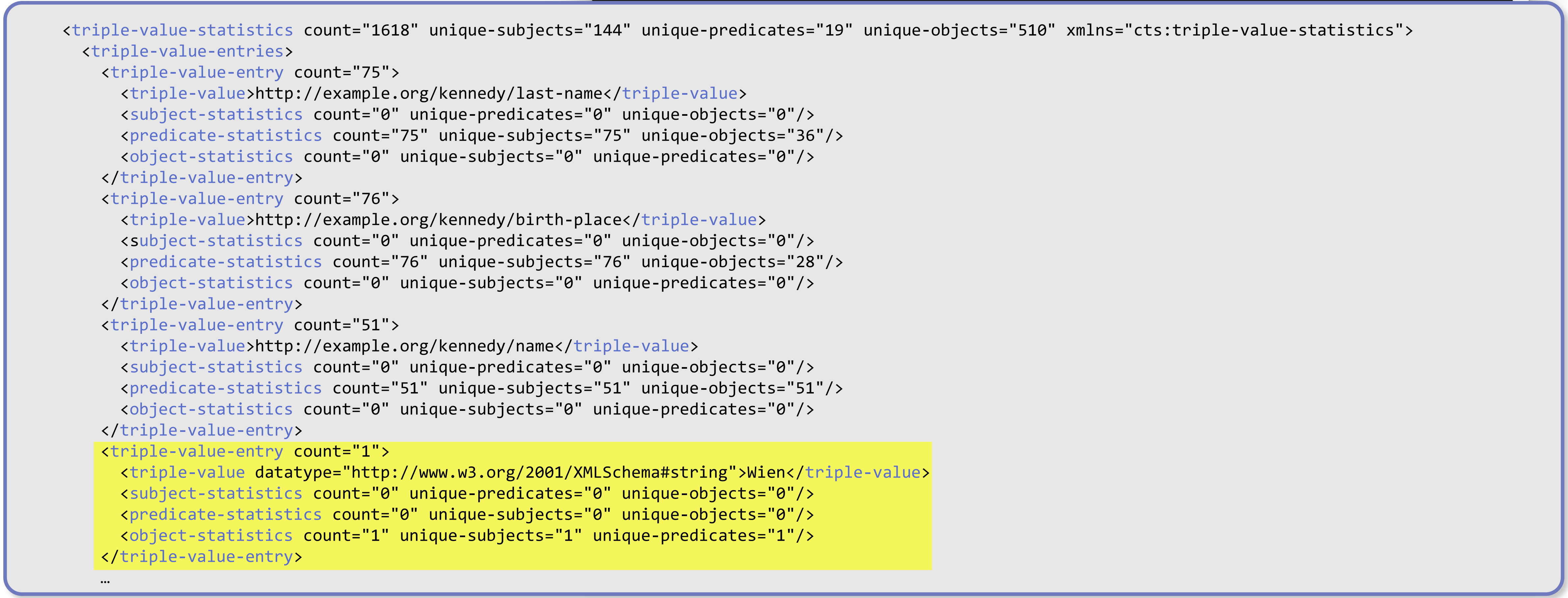


```
2017-05-08 15:36:00.248 Info: [Event:id=Optic Statistics Trace] trace=Query1 sessionKey=13674298726239676619 namedGraphs=
0 values=
2017-05-08 15:36:00.248 Info: <triple-value-statistics count="1618" unique-subjects="144" unique-predicates="19" unique-o
bjects="510" xmlns="cts:triple-value-statistics">
2017-05-08 15:36:00.248 Info:   <triple-value-entries>
2017-05-08 15:36:00.248 Info:     <triple-value-entry count="75">
2017-05-08 15:36:00.248 Info:       <triple-value>http://example.org/kennedy/first-name</triple-value>
2017-05-08 15:36:00.248 Info:       <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
2017-05-08 15:36:00.248 Info:       <predicate-statistics count="75" unique-subjects="75" unique-objects="51"/>
2017-05-08 15:36:00.248 Info:       <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
2017-05-08 15:36:00.248 Info:     </triple-value-entry>
2017-05-08 15:36:00.248 Info:     <triple-value-entry count="75">
2017-05-08 15:36:00.248 Info:       <triple-value>http://example.org/kennedy/last-name</triple-value>
2017-05-08 15:36:00.248 Info:       <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
2017-05-08 15:36:00.248 Info:       <predicate-statistics count="75" unique-subjects="75" unique-objects="36"/>
2017-05-08 15:36:00.248 Info:       <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
2017-05-08 15:36:00.248 Info:     </triple-value-entry>
2017-05-08 15:36:00.248 Info:     <triple-value-entry count="76">
2017-05-08 15:36:00.248 Info:       <triple-value>http://example.org/kennedy/birth-place</triple-value>
2017-05-08 15:36:00.248 Info:       <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
2017-05-08 15:36:00.248 Info:       <predicate-statistics count="76" unique-subjects="76" unique-objects="28"/>
2017-05-08 15:36:00.248 Info:       <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
2017-05-08 15:36:00.248 Info:     </triple-value-entry>
2017-05-08 15:36:00.248 Info:     <triple-value-entry count="51">
2017-05-08 15:36:00.248 Info:       <triple-value>http://example.org/kennedy/name</triple-value>
2017-05-08 15:36:00.248 Info:       <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
2017-05-08 15:36:00.248 Info:       <predicate-statistics count="51" unique-subjects="51" unique-objects="51"/>
2017-05-08 15:36:00.248 Info:       <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
2017-05-08 15:36:00.248 Info:     </triple-value-entry>
2017-05-08 15:36:00.248 Info:     <triple-value-entry count="1">
2017-05-08 15:36:00.248 Info:       <triple-value datatype="http://www.w3.org/2001/XMLSchema#string">Wien</triple-value>
2017-05-08 15:36:00.248 Info:       <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
```

# Statistics



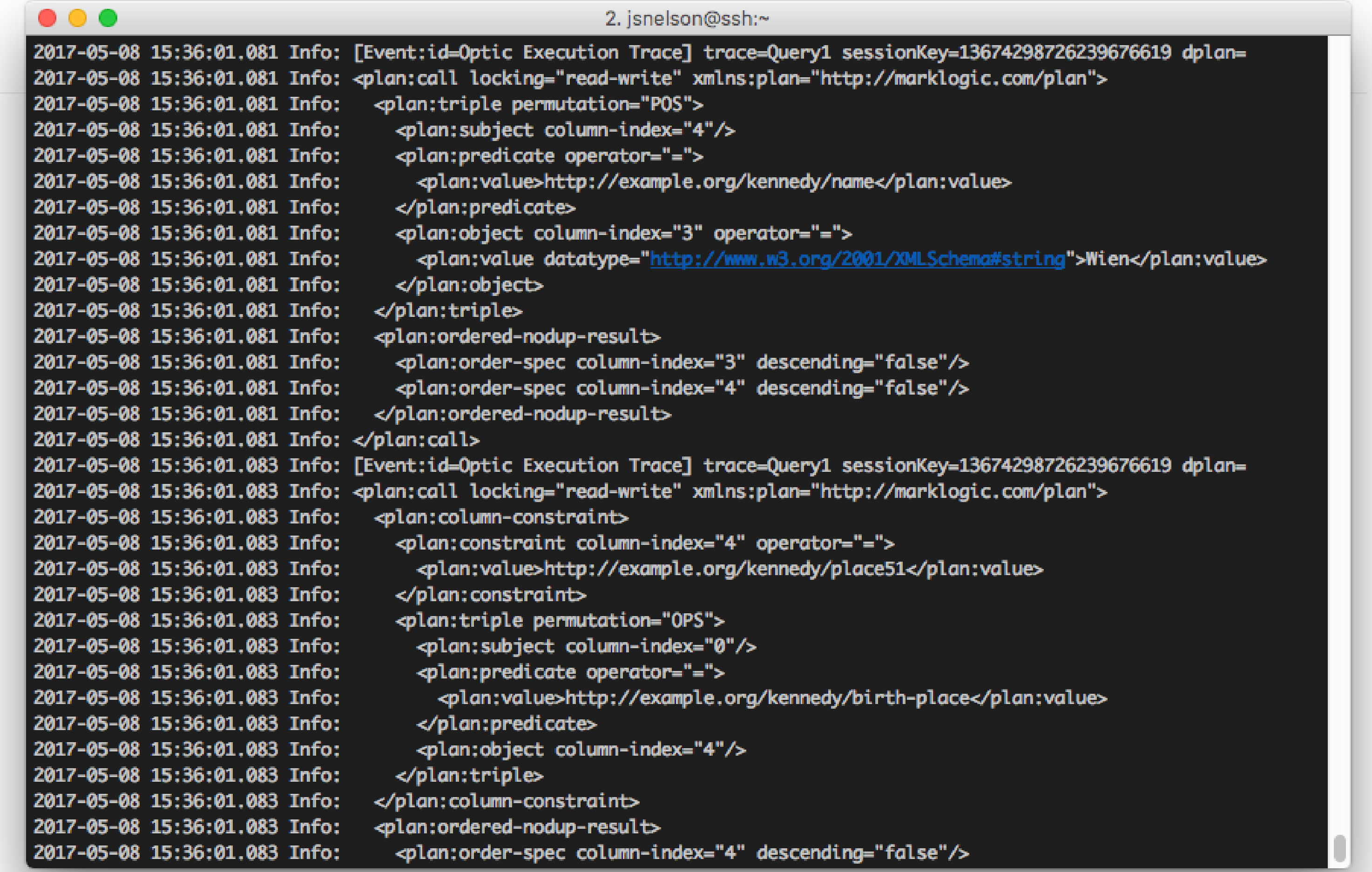
```
2017-05-08 15:36:00.248 Info: [Event:id=Optic Statistics Trace] trace=Query1 sessionKey=13674298726239676619 namedGraphs=
0 values=
2017-05-08 15:36:00.248 Info: <triple-value-statistics count="1618" unique-subjects="144" unique-predicates="19" unique-o
```



```
<triple-value-statistics count="1618" unique-subjects="144" unique-predicates="19" unique-objects="510" xmlns="cts:triple-value-statistics">
  <triple-value-entries>
    <triple-value-entry count="75">
      <triple-value>http://example.org/kennedy/last-name</triple-value>
      <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
      <predicate-statistics count="75" unique-subjects="75" unique-objects="36"/>
      <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
    </triple-value-entry>
    <triple-value-entry count="76">
      <triple-value>http://example.org/kennedy/birth-place</triple-value>
      <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
      <predicate-statistics count="76" unique-subjects="76" unique-objects="28"/>
      <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
    </triple-value-entry>
    <triple-value-entry count="51">
      <triple-value>http://example.org/kennedy/name</triple-value>
      <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
      <predicate-statistics count="51" unique-subjects="51" unique-objects="51"/>
      <object-statistics count="0" unique-subjects="0" unique-predicates="0"/>
    </triple-value-entry>
    <triple-value-entry count="1">
      <triple-value datatype="http://www.w3.org/2001/XMLSchema#string">Wien</triple-value>
      <subject-statistics count="0" unique-predicates="0" unique-objects="0"/>
      <predicate-statistics count="0" unique-subjects="0" unique-objects="0"/>
      <object-statistics count="1" unique-subjects="1" unique-predicates="1"/>
    </triple-value-entry>
    …
```

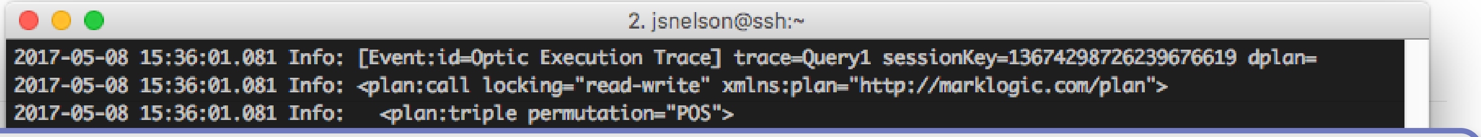
# Query Execution

- Trace flag "SPARQL Execution" [ML7,8,9]
- Trace flag "Optic Execution" [ML9]
- Trace option ("trace=XXXX") to sem:sparql() and xdmp:sql() [ML9]



```
2017-05-08 15:36:01.081 Info: [Event:id=Optic Execution Trace] trace=Query1 sessionKey=13674298726239676619 dplan=
2017-05-08 15:36:01.081 Info: <plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.081 Info:   <plan:triple permutation="POS">
2017-05-08 15:36:01.081 Info:     <plan:subject column-index="4"/>
2017-05-08 15:36:01.081 Info:     <plan:predicate operator="=">
2017-05-08 15:36:01.081 Info:       <plan:value>http://example.org/kennedy/name</plan:value>
2017-05-08 15:36:01.081 Info:     </plan:predicate>
2017-05-08 15:36:01.081 Info:     <plan:object column-index="3" operator="=">
2017-05-08 15:36:01.081 Info:       <plan:value datatype="http://www.w3.org/2001/XMLSchema#string">Wien</plan:value>
2017-05-08 15:36:01.081 Info:     </plan:object>
2017-05-08 15:36:01.081 Info:   </plan:triple>
2017-05-08 15:36:01.081 Info:   <plan:ordered-nodup-result>
2017-05-08 15:36:01.081 Info:     <plan:order-spec column-index="3" descending="false"/>
2017-05-08 15:36:01.081 Info:     <plan:order-spec column-index="4" descending="false"/>
2017-05-08 15:36:01.081 Info:   </plan:ordered-nodup-result>
2017-05-08 15:36:01.081 Info: </plan:call>
2017-05-08 15:36:01.083 Info: [Event:id=Optic Execution Trace] trace=Query1 sessionKey=13674298726239676619 dplan=
2017-05-08 15:36:01.083 Info: <plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.083 Info:   <plan:column-constraint>
2017-05-08 15:36:01.083 Info:     <plan:constraint column-index="4" operator="=">
2017-05-08 15:36:01.083 Info:       <plan:value>http://example.org/kennedy/place51</plan:value>
2017-05-08 15:36:01.083 Info:     </plan:constraint>
2017-05-08 15:36:01.083 Info:     <plan:triple permutation="OPS">
2017-05-08 15:36:01.083 Info:       <plan:subject column-index="0"/>
2017-05-08 15:36:01.083 Info:       <plan:predicate operator="=">
2017-05-08 15:36:01.083 Info:         <plan:value>http://example.org/kennedy/birth-place</plan:value>
2017-05-08 15:36:01.083 Info:       </plan:predicate>
2017-05-08 15:36:01.083 Info:       <plan:object column-index="4"/>
2017-05-08 15:36:01.083 Info:     </plan:triple>
2017-05-08 15:36:01.083 Info:   </plan:column-constraint>
2017-05-08 15:36:01.083 Info:   <plan:ordered-nodup-result>
2017-05-08 15:36:01.083 Info:     <plan:order-spec column-index="4" descending="false"/>
```

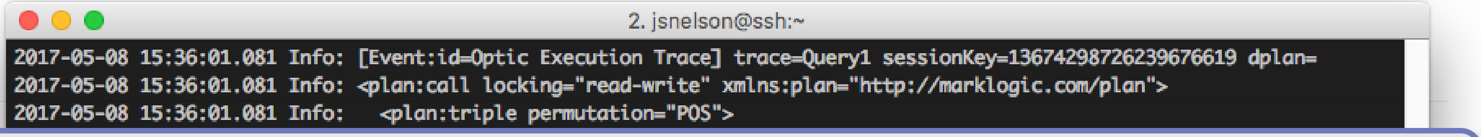
# Query Execution



```
2017-05-08 15:36:01.081 Info: [Event:id=Optic Execution Trace] trace=Query1 sessionKey=13674298726239676619 dplan=
2017-05-08 15:36:01.081 Info: <plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.081 Info:   <plan:triple permutation="POS">
```

```
<plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
  <plan:triple permutation="POS">
    <plan:subject column-index="4"/>
    <plan:predicate operator="=">
      <plan:value>http://example.org/kennedy/name</plan:value>
    </plan:predicate>
    <plan:object column-index="3" operator="=">
      <plan:value datatype="http://www.w3.org/2001/XMLSchema#string">Wien</plan:value>
    </plan:object>
  </plan:triple>
  <plan:ordered-nodup-result>
    <plan:order-spec column-index="3" descending="false"/>
    <plan:order-spec column-index="4" descending="false"/>
  </plan:ordered-nodup-result>
</plan:call>
```

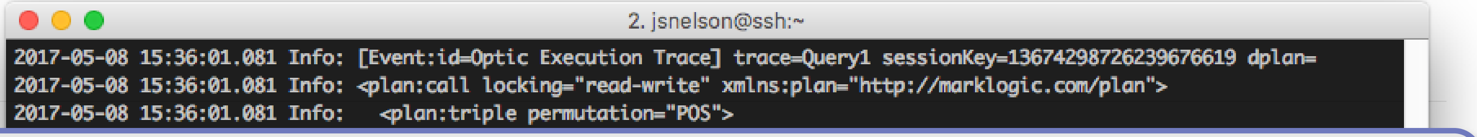
# Query Execution



```
2017-05-08 15:36:01.081 Info: [Event:id=Optic Execution Trace] trace=Query1 sessionKey=13674298726239676619 dplan=
2017-05-08 15:36:01.081 Info: <plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.081 Info:   <plan:triple permutation="POS">
```

```
<plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
  <plan:column-constraint>
    <plan:constraint column-index="4" operator="=">
      <plan:value>http://example.org/kennedy/place51</plan:value>
    </plan:constraint>
    <plan:triple permutation="OPS">
      <plan:subject column-index="0"/>
      <plan:predicate operator="=">
        <plan:value>http://example.org/kennedy/birth-place</plan:value>
      </plan:predicate>
      <plan:object column-index="4"/>
    </plan:triple>
  </plan:column-constraint>
  <plan:ordered-nodup-result>
    <plan:order-spec column-index="4" descending="false"/>
    <plan:order-spec column-index="0" descending="false"/>
  </plan:ordered-nodup-result>
</plan:call>
```

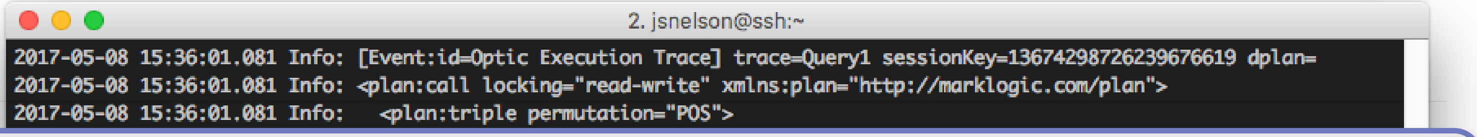
# Query Execution



```
2017-05-08 15:36:01.081 Info: [Event:id=Optic Execution Trace] trace=Query1 sessionKey=13674298726239676619 dplan=
2017-05-08 15:36:01.081 Info: <plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.081 Info:   <plan:triple permutation="POS">
```

```
<plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
  <plan:column-constraint>
    <plan:constraint column-index="0" operator="=">
      <plan:value>http://example.org/kennedy/person27</plan:value>
    </plan:constraint>
    <plan:triple permutation="PSO">
      <plan:subject column-index="0"/>
      <plan:predicate operator="=">
        <plan:value>http://example.org/kennedy/last-name</plan:value>
      </plan:predicate>
      <plan:object column-index="2"/>
    </plan:triple>
  </plan:column-constraint>
  <plan:ordered-nodup-result>
    <plan:order-spec column-index="0" descending="false"/>
    <plan:order-spec column-index="2" descending="false"/>
  </plan:ordered-nodup-result>
</plan:call>
```

# Query Execution

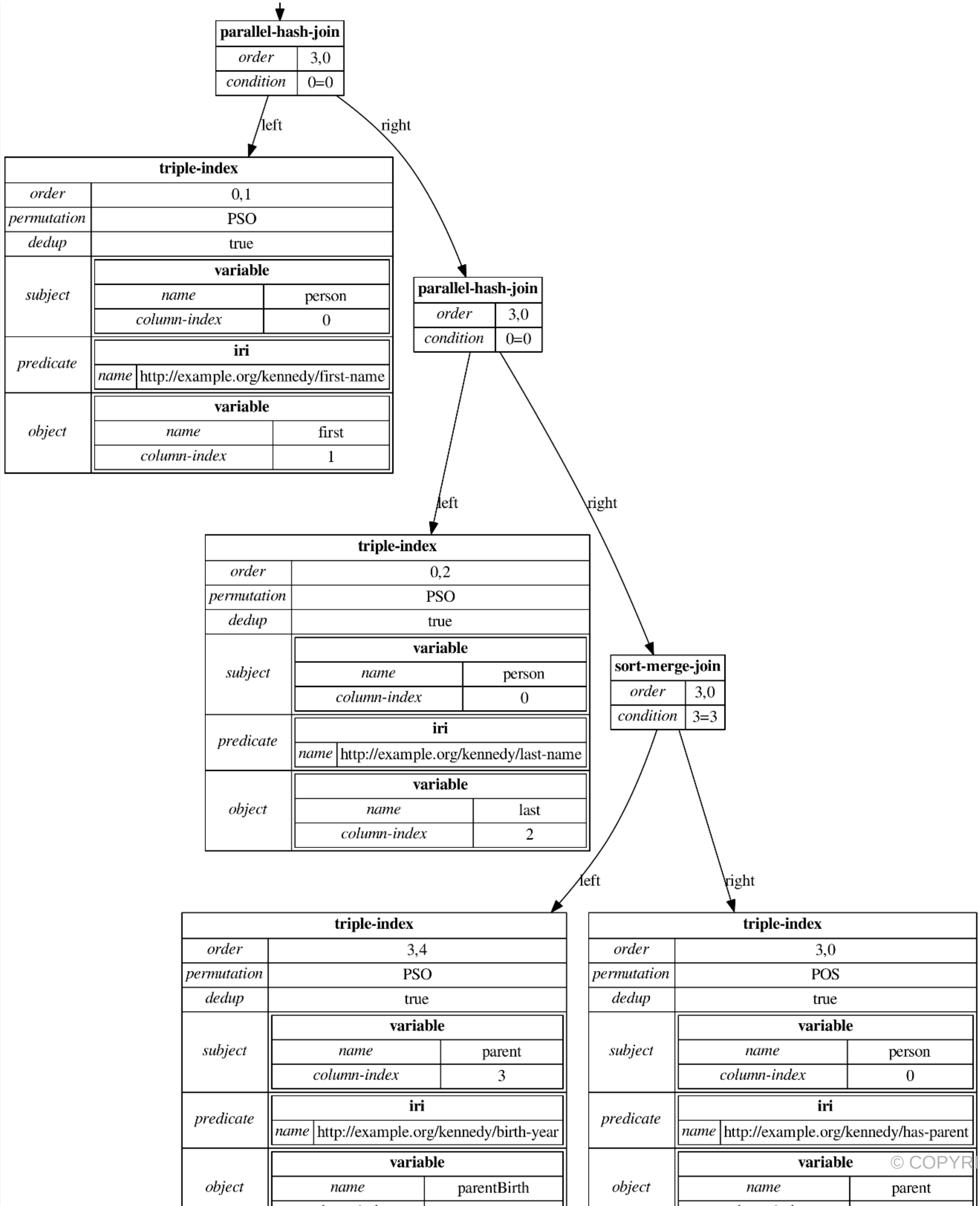


```
2017-05-08 15:36:01.081 Info: [Event:id=Optic Execution Trace] trace=Query1 sessionKey=13674298726239676619 dplan=
2017-05-08 15:36:01.081 Info: <plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
2017-05-08 15:36:01.081 Info:   <plan:triple permutation="POS">
```

```
<plan:call locking="read-write" xmlns:plan="http://marklogic.com/plan">
  <plan:column-constraint>
    <plan:constraint column-index="0" operator="=">
      <plan:value>http://example.org/kennedy/person27</plan:value>
    </plan:constraint>
    <plan:triple permutation="POS">
      <plan:subject column-index="0"/>
      <plan:predicate operator="=">
        <plan:value>http://example.org/kennedy/first-name</plan:value>
      </plan:predicate>
      <plan:object column-index="1"/>
    </plan:triple>
  </plan:column-constraint>
  <plan:ordered-nodup-result>
    <plan:order-spec column-index="1" descending="false"/>
    <plan:order-spec column-index="0" descending="false"/>
  </plan:ordered-nodup-result>
</plan:call>
```

# Problem Query

```
select * {
  ?person :first-name ?first .
  ?person :last-name ?last .
  ?person :has-parent ?parent .
  ?parent :birth-year ?parentBirth .
  filter(?parentBirth < '1890')
}
```



parallel-hash-join
order 3,0
condition 0=0

left

triple-index
order 0,1
permutation PSO
dedup true
subject: variable — name person, column-index 0
predicate: iri — name http://example.org/kennedy/first-name
object: variable — name first, column-index 1

right

parallel-hash-join
order 3,0
condition 0=0

left

triple-index
order 0,2
permutation PSO
dedup true
subject: variable — name person, column-index 0
predicate: iri — name http://example.org/kennedy/last-name
object: variable — name last, column-index 2

right

sort-merge-join
order 3,0
condition 3=3

left

triple-index
order 3,4
permutation PSO
dedup true
subject: variable — name parent, column-index 3
predicate: iri — name http://example.org/kennedy/birth-year
object: variable — name parentBirth

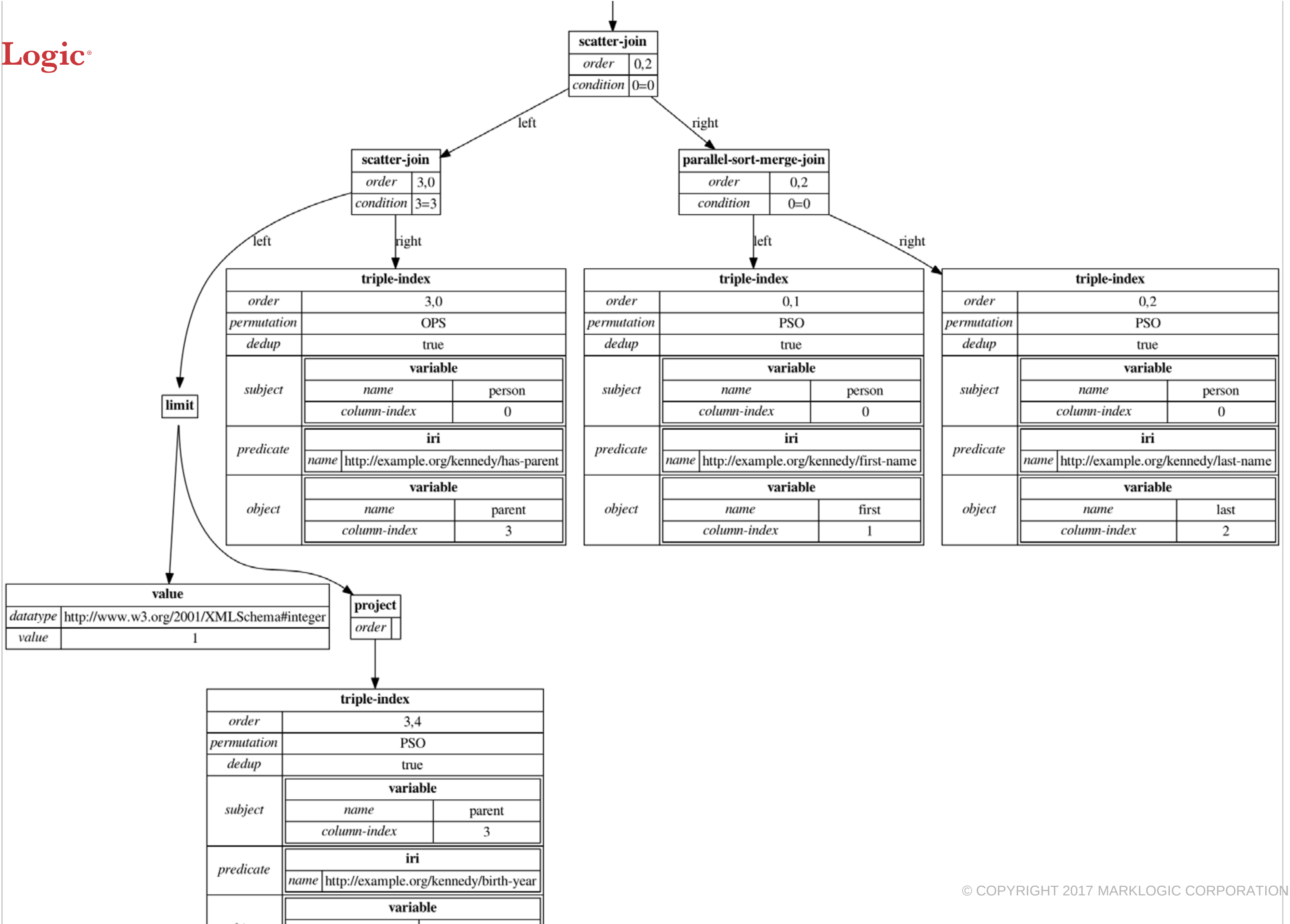
right

triple-index
order 3,0
permutation POS
dedup true
subject: variable — name person, column-index 0
predicate: iri — name http://example.org/kennedy/has-parent
object: variable — name parent

MarkLogic

# Improvement: Add Cardinality Hints

```
select * {
  ?person :first-name ?first .
  ?person :last-name ?last .
  ?person :has-parent ?parent .
  { select * {
    ?parent :birth-year ?parentBirth .
    filter(?parentBirth < '1890')
  } limit 1 }
}
```



| scatter-join | |
|---|---|
| *order* | 0,2 |
| *condition* | 0=0 |

| scatter-join | |
|---|---|
| *order* | 3,0 |
| *condition* | 3=3 |

| parallel-sort-merge-join | |
|---|---|
| *order* | 0,2 |
| *condition* | 0=0 |

| triple-index | |
|---|---|
| *order* | 3,0 |
| *permutation* | OPS |
| *dedup* | true |
| *subject* | variable<br>*name* person<br>*column-index* 0 |
| *predicate* | iri<br>*name* http://example.org/kennedy/has-parent |
| *object* | variable<br>*name* parent<br>*column-index* 3 |

| triple-index | |
|---|---|
| *order* | 0,1 |
| *permutation* | PSO |
| *dedup* | true |
| *subject* | variable<br>*name* person<br>*column-index* 0 |
| *predicate* | iri<br>*name* http://example.org/kennedy/first-name |
| *object* | variable<br>*name* first<br>*column-index* 1 |

| triple-index | |
|---|---|
| *order* | 0,2 |
| *permutation* | PSO |
| *dedup* | true |
| *subject* | variable<br>*name* person<br>*column-index* 0 |
| *predicate* | iri<br>*name* http://example.org/kennedy/last-name |
| *object* | variable<br>*name* last<br>*column-index* 2 |

limit

| value | |
|---|---|
| *datatype* | http://www.w3.org/2001/XMLSchema#integer |
| *value* | 1 |

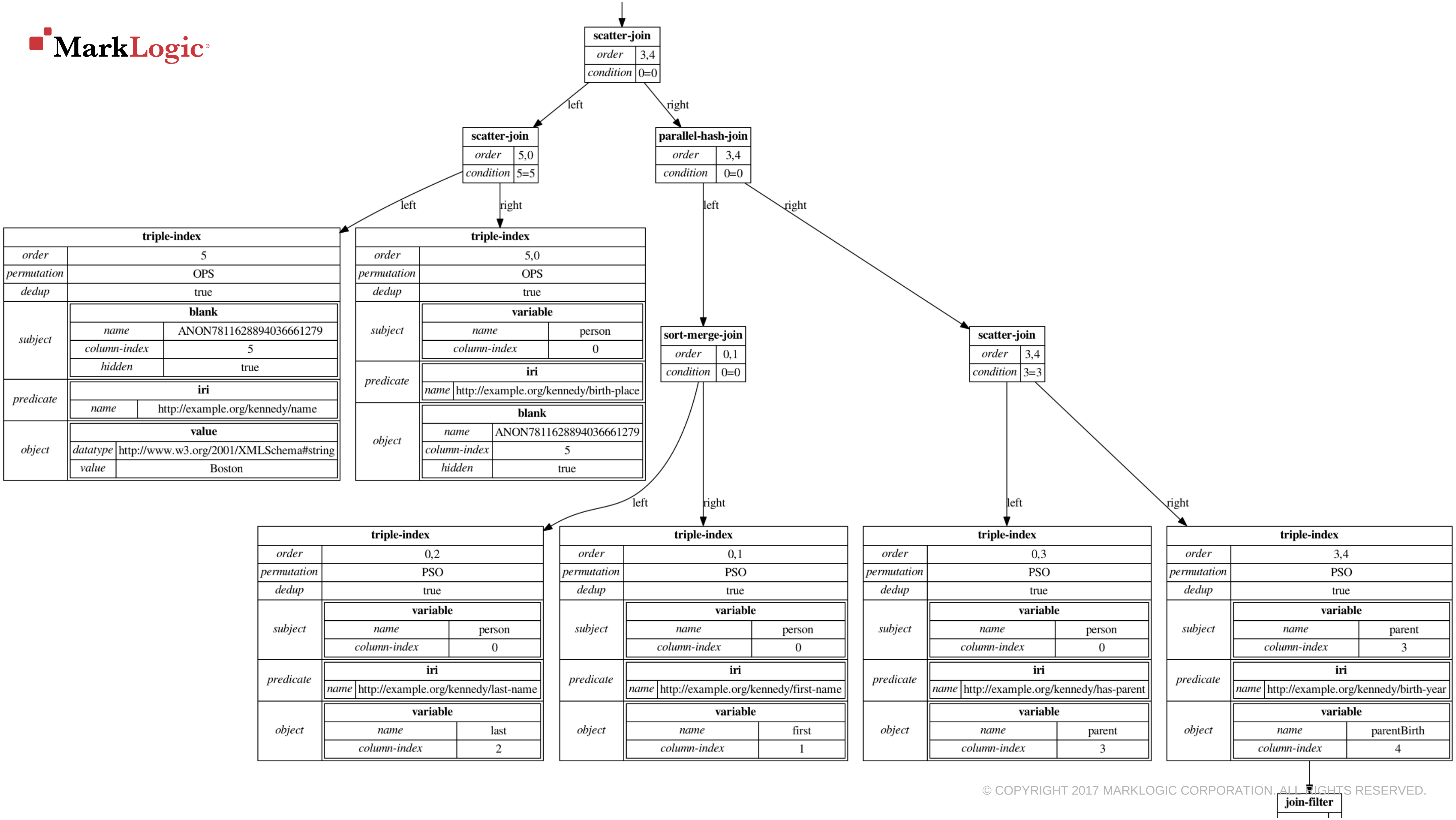
| project | |
|---|---|
| *order* | |

| triple-index | |
|---|---|
| *order* | 3,4 |
| *permutation* | PSO |
| *dedup* | true |
| *subject* | variable<br>*name* parent<br>*column-index* 3 |
| *predicate* | iri<br>*name* http://example.org/kennedy/birth-year |
| | variable |

MarkLogic®

# Improvement: Add Constraining Condition

```
select * {
  ?person :first-name ?first .
  ?person :last-name ?last .
  ?person :birth-place [:name 'Boston'] .
  ?person :has-parent ?parent .
  ?parent :birth-year ?parentBirth .
  filter(?parentBirth < '1890')
}
```



| scatter-join | |
|---|---|
| order | 3,4 |
| condition | 0=0 |

| scatter-join | |
|---|---|
| order | 5,0 |
| condition | 5=5 |

| parallel-hash-join | |
|---|---|
| order | 3,4 |
| condition | 0=0 |

| triple-index | |
|---|---|
| order | 5 |
| permutation | OPS |
| dedup | true |
| subject | blank<br>name ANON7811628894036661279<br>column-index 5<br>hidden true |
| predicate | iri<br>name http://example.org/kennedy/name |
| object | value<br>datatype http://www.w3.org/2001/XMLSchema#string<br>value Boston |

| triple-index | |
|---|---|
| order | 5,0 |
| permutation | OPS |
| dedup | true |
| subject | variable<br>name person<br>column-index 0 |
| predicate | iri<br>name http://example.org/kennedy/birth-place |
| object | blank<br>name ANON7811628894036661279<br>column-index 5<br>hidden true |

| sort-merge-join | |
|---|---|
| order | 0,1 |
| condition | 0=0 |

| scatter-join | |
|---|---|
| order | 3,4 |
| condition | 3=3 |

| triple-index | |
|---|---|
| order | 0,2 |
| permutation | PSO |
| dedup | true |
| subject | variable<br>name person<br>column-index 0 |
| predicate | iri<br>name http://example.org/kennedy/last-name |
| object | variable<br>name last<br>column-index 2 |

| triple-index | |
|---|---|
| order | 0,1 |
| permutation | PSO |
| dedup | true |
| subject | variable<br>name person<br>column-index 0 |
| predicate | iri<br>name http://example.org/kennedy/first-name |
| object | variable<br>name first<br>column-index 1 |

| triple-index | |
|---|---|
| order | 0,3 |
| permutation | PSO |
| dedup | true |
| subject | variable<br>name person<br>column-index 0 |
| predicate | iri<br>name http://example.org/kennedy/has-parent |
| object | variable<br>name parent<br>column-index 3 |

| triple-index | |
|---|---|
| order | 3,4 |
| permutation | PSO |
| dedup | true |
| subject | variable<br>name parent<br>column-index 3 |
| predicate | iri<br>name http://example.org/kennedy/birth-year |
| object | variable<br>name parentBirth<br>column-index 4 |

join-filter

# Understanding the Triple Caches

# Triple Patterns use the Triple Index

- Designed to look up triple patterns
- Facilitates fast joins
- 4 triple permutations
- Not memory mapped - cached for performance
- Works seamlessly with other indexes

# Triple Index

| subject | predicate | object | doc ID | position |
|---|---|---|---|---|
| :person4 | :first-name | “John” | 11 | 5 - 9 |
| :person5 | :alma-mater | :Brown | 4 | 25 - 40 |
| :person5 | :birth-year | 1929 | 9 | 13 - 17 |
| … | | | | |

# Triple Data and Triple Values

| subject | predicate | object | doc ID | position |
|---|---|---|---|---|
| 4 | 3 | 6 | 11 | 5 - 9 |
| 5 | 0 | 2 | 4 | 25 - 40 |
| 5 | 1 | 7 | 9 | 13 - 17 |
| … | | | | |

| ordinal | tag | value |
|---|---|---|
| … | | |
| 3 | IRI | :first-name |
| 4 | IRI | :person4 |
| 5 | IRI | :person5 |
| 6 | STRING | “John” |
| 7 | DECIMAL | 1929 |
| … | | |

# Triple and Triple Value Cache

- D-node caches
- Partitioned for lock contention
- Configurable maximum size
- Grows and shrinks
  - No up-front memory allocation
  - Trickle removed after 30s inactivity (user configurable)
- Flexibility
  - Size according to importance of triple based queries
  - Size for working set
  - Size big “in case”, but rely on it normally being small

# Triple and Triple Value Cache

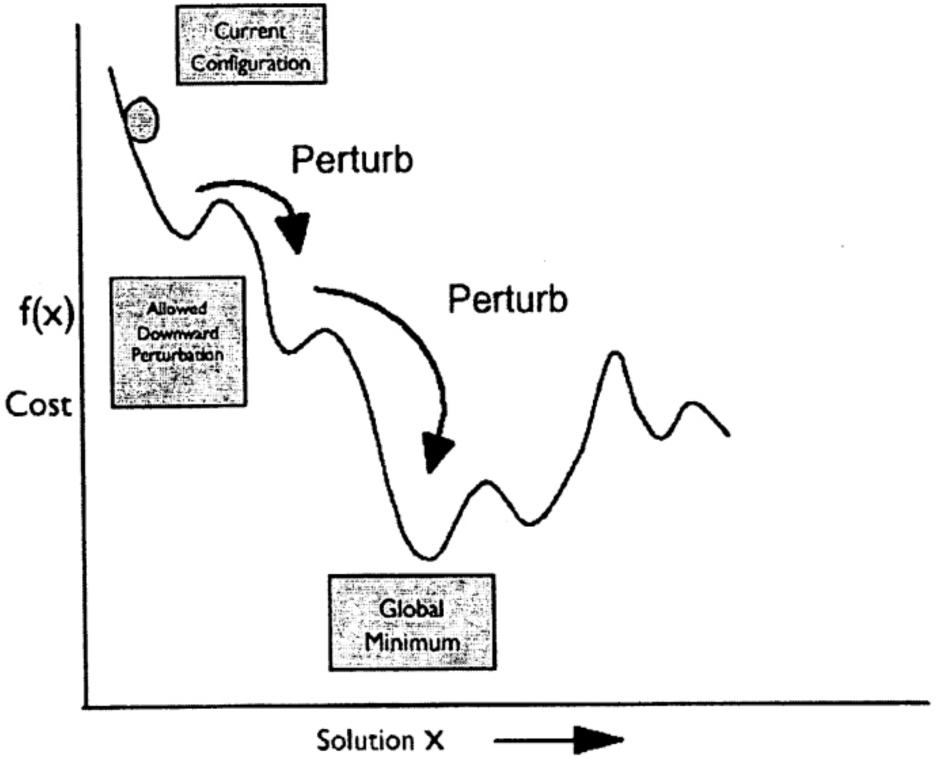
- xdmp:query-meters()
  - cache hits/misses for a query
- xdmp:forest-status()
  - cache hits/misses/hit rate/miss rate for each stand
- xdmp:cache-status()
  - Percentage busy/used/free by cache partition
- Admin UI database status and forest status
- Metering

MarkLogic

# Understanding Optimization

# SPARQL Optimization

- Cost estimation:
  - Column cardinality estimates
  - Sort order static analysis
- Query plan mutations:
  - Multiple orders available in the triple index
  - Multiple join implementations
  - Join re-ordering
- Simulated annealing:
  - Guided randomized search for a good query plan



# Optimization Levels

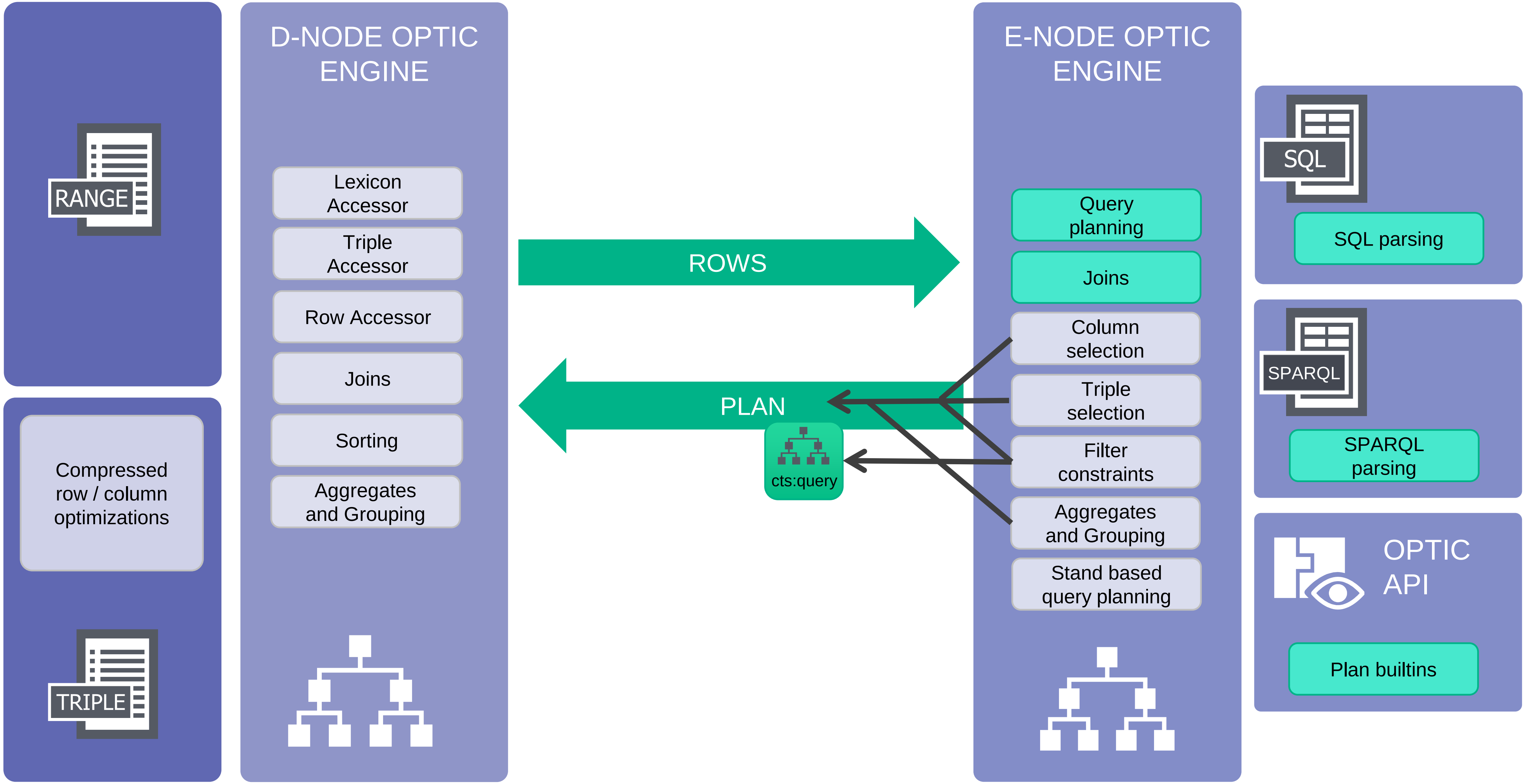
- Default optimization level is 1
- Larger queries may need a longer optimization process
- Optimization levels of 2, 3, 4 etc. are possible
- Optimization level of 0 only uses simple heuristic based optimization
- Trade off between planning and doing

```
sem:sparql("…",(),"optimize=2",
 cts:directory-query("/triples/")
)
```

credit: http://commons.wikimedia.org/wiki/User:Morio

# Improvements in MarkLogic 9

# OPTIC ENGINE ARCHITECTURE



Compressed row / column optimizations

## D-NODE OPTIC ENGINE

Lexicon Accessor

Triple Accessor

Row Accessor

Joins

Sorting

Aggregates and Grouping

ROWS

PLAN

cts:query

## E-NODE OPTIC ENGINE

Query planning

Joins

Column selection

Triple selection

Filter constraints

Aggregates and Grouping

Stand based query planning

SQL parsing

SPARQL parsing

OPTIC API

Plan builtins

# What the New Optimizations Do For You!

- Faster ORDER BY from the index with a known predicate
  - POS permutation in the triple index
- Faster descending ORDER BY
  - descending order triple index access
- Faster multi-column ORDER BY
  - partial sort uses major sort order from the triple index
- Faster range-based triple index access
  - both upper and lower bound
- Faster grouping
  - hash based grouping
- Faster disk reads
  - especially on Windows

# Getting the Most from MarkLogic Semantics

- Introduction
- Performance at scale
  - Tips – general, detailed
  - Case Studies – Educational publisher, Data store for clinical data, Global bank
- Under the hood
  - Understanding SPARQL execution
  - Understanding the Triple Caches
  - Understanding optimization
  - Improvements in MarkLogic 9

# Questions?