



Advanced RDF & SPARQL

Advanced features within Stardog

Taught by:



Joseph Hayes

Senior Software Engineer

Learning Objectives



Use PATHS queries to discover the many ways 2 nodes in your graph can be connected



Use full text-search with Stardog data



Discover where your data comes from using Studio's Provenance feature



Explore your spatial data using Stardog's GeoSPARQL support



Centrally manage your queries using the Stored Query Service (SQS)



Write flexible queries using Named Graph Aliases



Bridge the gap between Property Graphs and a Knowledge Graph by leveraging Edge Properties (a RDF*/SPARQL* extension)





Path Queries



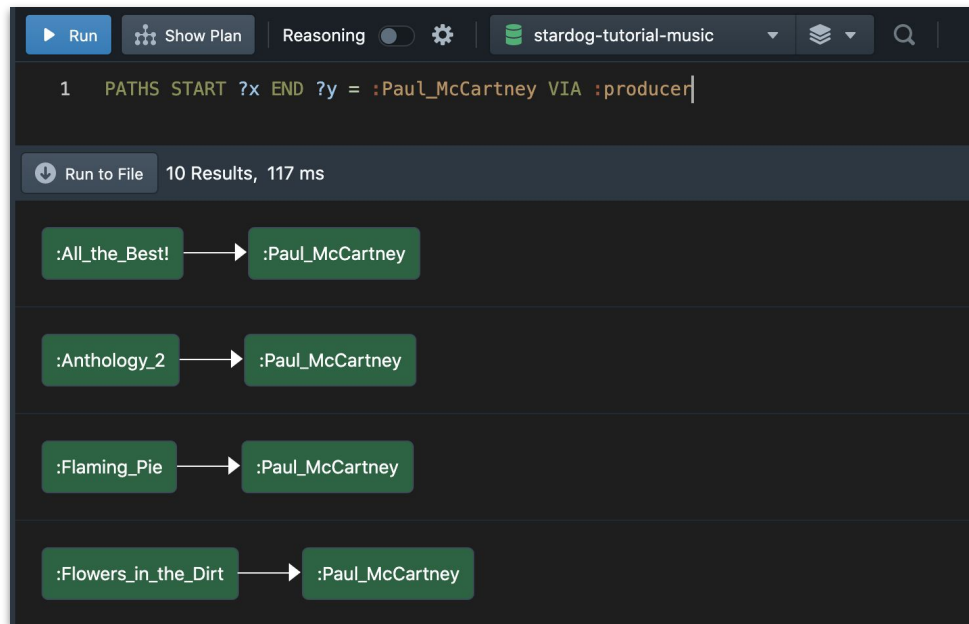
Path Queries

- Path queries find paths between two nodes in a graph
- Similar to SPARQL property paths
- SPARQL property paths return only start and end nodes, Stardog path queries return all the intermediate nodes as well
- Stardog path queries allow arbitrary SPARQL patterns to be used in the query
- By default path queries return the shortest path between two nodes but can be used to also find all paths between two nodes. Paths can be:
 - Simple (a path edge is a graph edge)
 - Complex (a path edge is a subgraph)
- You can also use path queries to find simple cycle paths (where the start node is equal to the end node)



Example: Path Queries

- Using the sample music database we can find all the items where Paul McCartney is the producer by running the query:



The screenshot shows the Stardog web interface with the query editor and results pane. The query is a path query designed to find all items where Paul McCartney is the producer.

```
1 PATHS START ?x END ?y = :Paul_McCartney VIA :producer
```

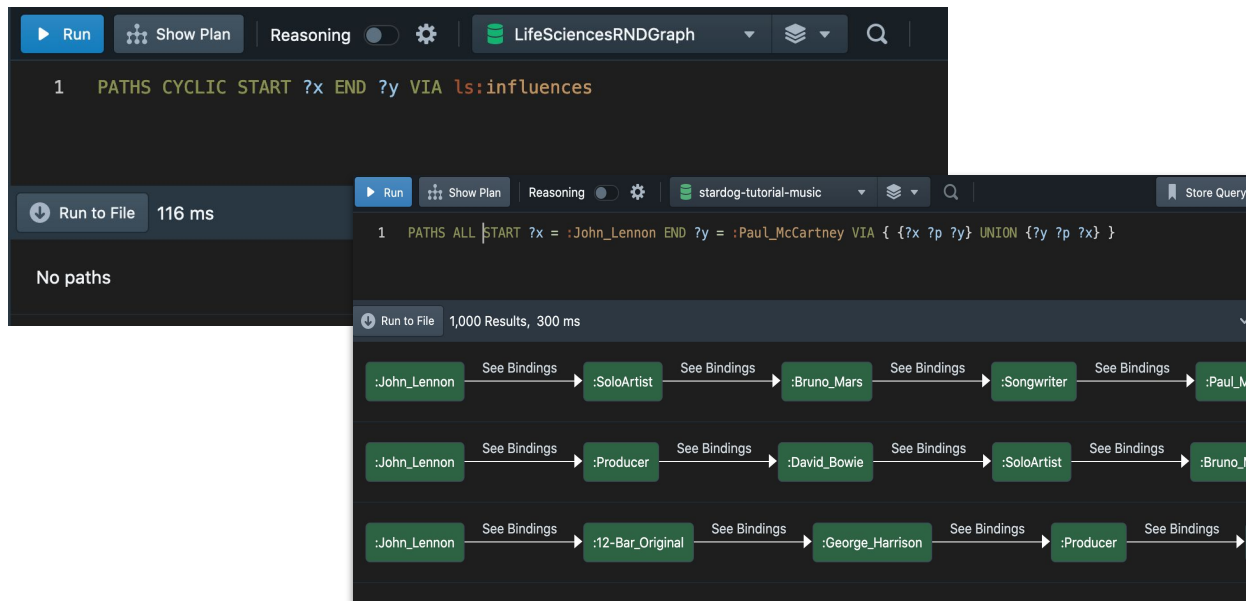
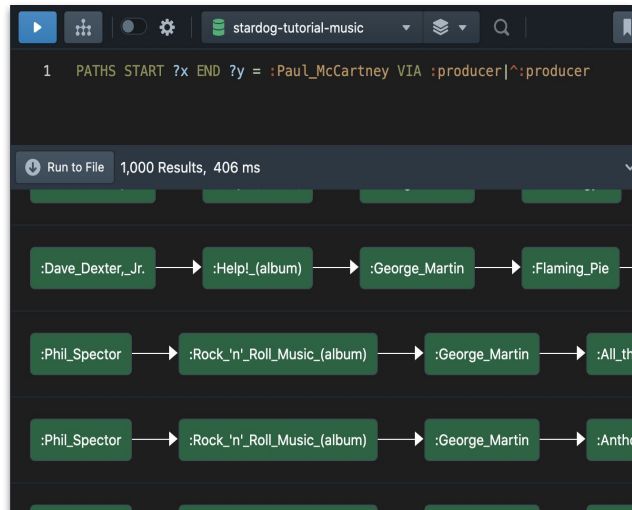
The results pane shows 10 results in 117 ms. The visible results are:

- `:All_the_Best!` → `:Paul_McCartney`
- `:Anthology_2` → `:Paul_McCartney`
- `:Flaming_Pie` → `:Paul_McCartney`
- `:Flowers_in_the_Dirt` → `:Paul_McCartney`



Example: Path Queries

- We can also query complex paths, cyclic paths, and even all paths between nodes
(**NOTE:** Using PATHS ALL can return a very large number of results - use them when necessary):





Demo





Full Text Search

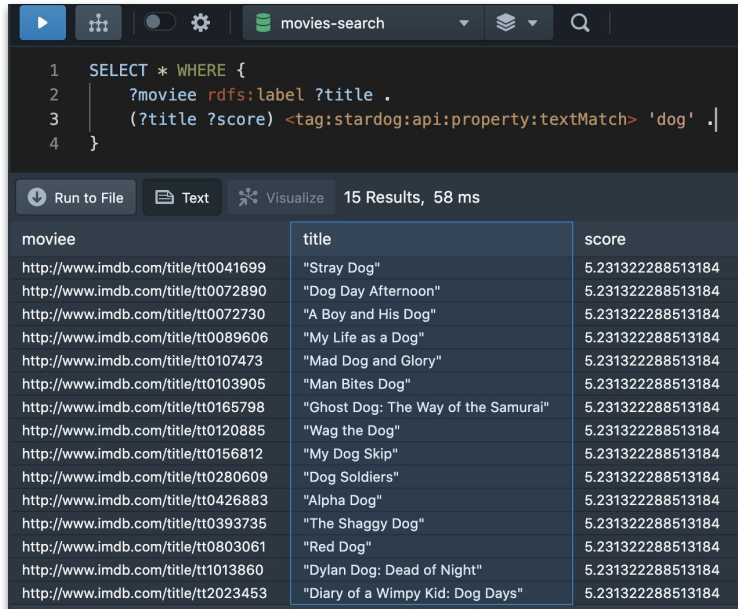
Full Text-Search

- You can enable full text-search on any Stardog database; this will create a “search document” for each RDF literal
- Search enabled databases provide a special predicate for use in simple queries
- A Service Search form is also provided for queries with multiple input constants and output variables
- Search results can be highlighted
- You can extend Stardog’s native full text capabilities by creating a custom Analyzer



Example: Full Text-Search

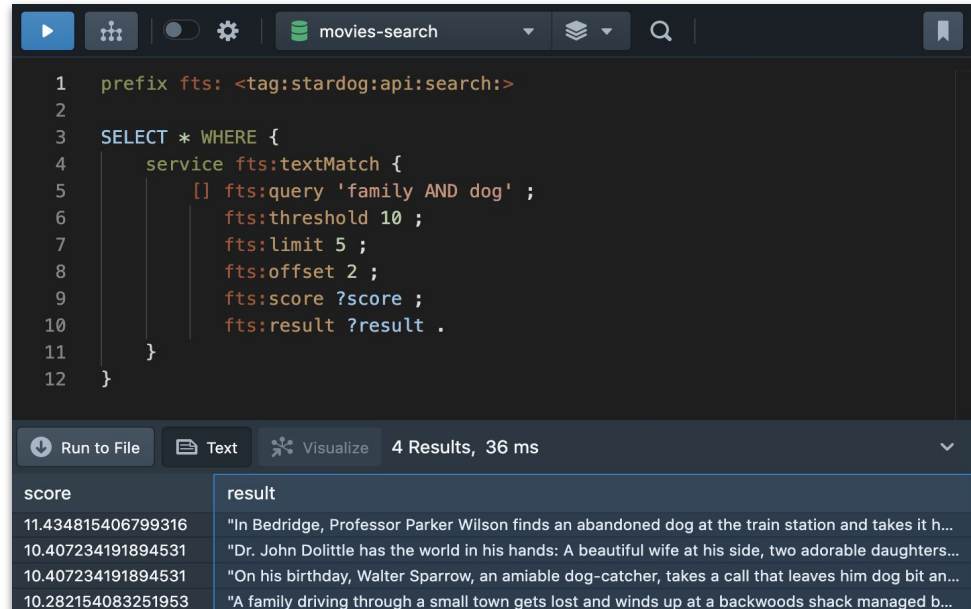
- Stardog provides a `textMatch` predicate and a Service form of Search



```
1 SELECT * WHERE {
2   ?moviee rdfs:label ?title .
3   (?title ?score) <tag:stardog:api:property:textMatch> 'dog' .
4 }
```

Run to File Text Visualize 15 Results, 58 ms

moviee	title	score
http://www.imdb.com/title/tt0041699	"Stray Dog"	5.231322288513184
http://www.imdb.com/title/tt0072890	"Dog Day Afternoon"	5.231322288513184
http://www.imdb.com/title/tt0072730	"A Boy and His Dog"	5.231322288513184
http://www.imdb.com/title/tt0089606	"My Life as a Dog"	5.231322288513184
http://www.imdb.com/title/tt0107473	"Mad Dog and Glory"	5.231322288513184
http://www.imdb.com/title/tt0103905	"Man Bites Dog"	5.231322288513184
http://www.imdb.com/title/tt0165798	"Ghost Dog: The Way of the Samurai"	5.231322288513184
http://www.imdb.com/title/tt0120885	"Wag the Dog"	5.231322288513184
http://www.imdb.com/title/tt0156812	"My Dog Skip"	5.231322288513184
http://www.imdb.com/title/tt0280609	"Dog Soldiers"	5.231322288513184
http://www.imdb.com/title/tt0426883	"Alpha Dog"	5.231322288513184
http://www.imdb.com/title/tt0393735	"The Shaggy Dog"	5.231322288513184
http://www.imdb.com/title/tt0803061	"Red Dog"	5.231322288513184
http://www.imdb.com/title/tt1013860	"Dylan Dog: Dead of Night"	5.231322288513184
http://www.imdb.com/title/tt2023453	"Diary of a Wimpy Kid: Dog Days"	5.231322288513184



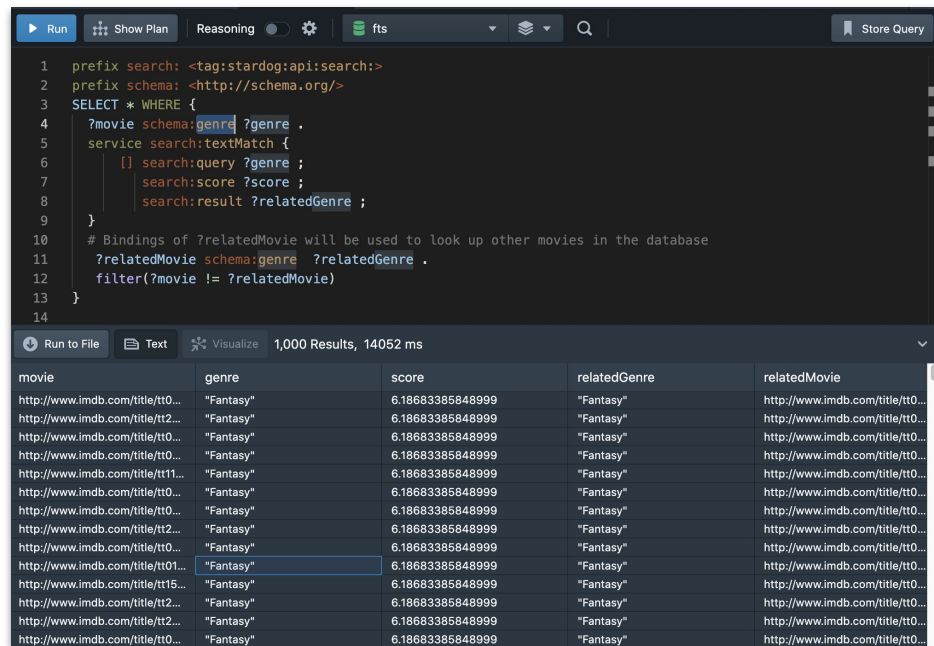
```
1 prefix fts: <tag:stardog:api:search:>
2
3 SELECT * WHERE {
4   service fts:textMatch {
5     [] fts:query 'family AND dog' ;
6     fts:threshold 10 ;
7     fts:limit 5 ;
8     fts:offset 2 ;
9     fts:score ?score ;
10    fts:result ?result .
11  }
12 }
```

Run to File Text Visualize 4 Results, 36 ms

score	result
11.434815406799316	"In Bedbridge, Professor Parker Wilson finds an abandoned dog at the train station and takes it h...
10.407234191894531	"Dr. John Dolittle has the world in his hands: A beautiful wife at his side, two adorable daughters...
10.407234191894531	"On his birthday, Walter Sparrow, an amiable dog-catcher, takes a call that leaves him dog bit an...
10.282154083251953	"A family driving through a small town gets lost and winds up at a backwoods shack managed b...

Example: Full Text-Search

- Stardog's Search capabilities can use a variable bound in the same scope (or provided externally) as input



The screenshot displays the Stardog Query Editor interface. The top toolbar includes buttons for 'Run', 'Show Plan', 'Reasoning', and a settings icon. A dropdown menu shows 'fts' selected. The query editor contains the following SPARQL code:

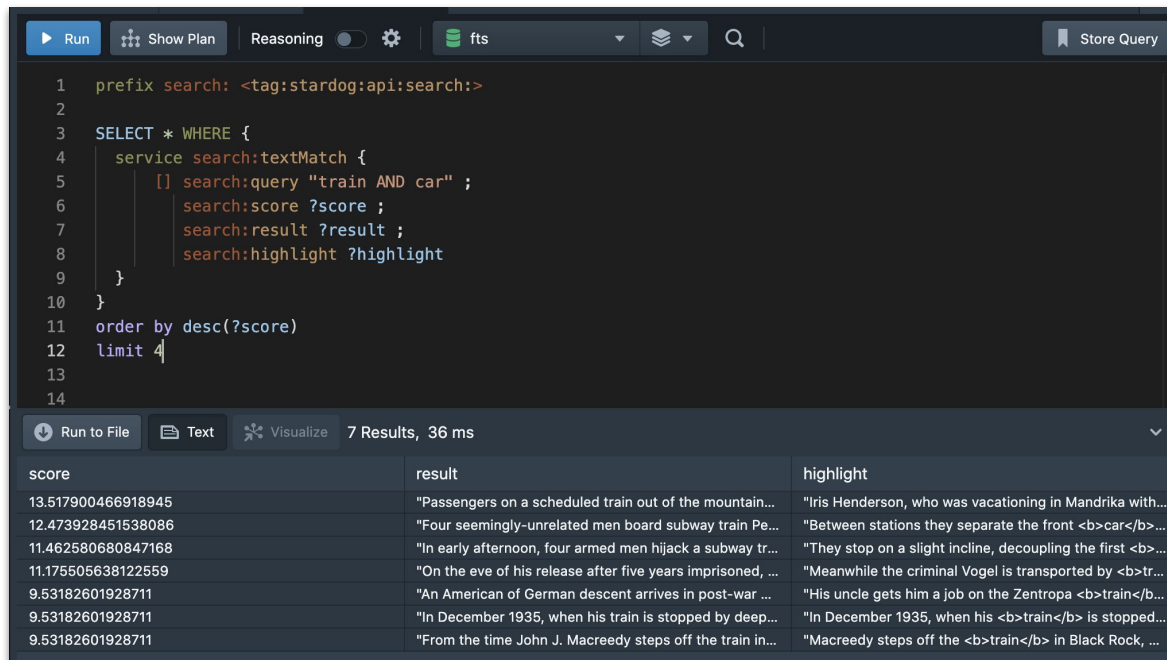
```
1 prefix search: <tag:stardog:api:search:>
2 prefix schema: <http://schema.org/>
3 SELECT * WHERE {
4   ?movie schema:genre ?genre .
5   service search:textMatch {
6     [] search:query ?genre ;
7       search:score ?score ;
8       search:result ?relatedGenre ;
9   }
10  # Bindings of ?relatedMovie will be used to look up other movies in the database
11  ?relatedMovie schema:genre ?relatedGenre .
12  filter(?movie != ?relatedMovie)
13 }
14
```

Below the query editor, the results are displayed in a table with 5 columns: 'movie', 'genre', 'score', 'relatedGenre', and 'relatedMovie'. The table shows 1,000 results in 14,052 ms. The first few rows all show 'Fantasy' as the genre and relatedGenre, with various IMDB movie URLs in the 'movie' and 'relatedMovie' columns. The 10th row is highlighted.

movie	genre	score	relatedGenre	relatedMovie
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt2...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt11...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt2...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt01...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt15...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt2...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt2...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...
http://www.imdb.com/title/tt0...	"Fantasy"	6.18683385848999	"Fantasy"	http://www.imdb.com/title/tt0...

Example: Full Text-Search

- You can also have Stardog return results with search hits highlighted



The screenshot shows the Stardog query editor interface. The top bar includes buttons for 'Run', 'Show Plan', 'Reasoning', and a settings icon. The 'fts' (Full Text Search) extension is selected. The query editor contains the following SPARQL query:

```
1 prefix search: <tag:stardog:api:search:>
2
3 SELECT * WHERE {
4   service search:textMatch {
5     [] search:query "train AND car" ;
6       search:score ?score ;
7       search:result ?result ;
8       search:highlight ?highlight
9   }
10 }
11 order by desc(?score)
12 limit 4
13
14
```

Below the query editor, the results are displayed in a table. The table has three columns: 'score', 'result', and 'highlight'. The results are sorted by score in descending order, and the 'highlight' column shows the search results with the matched terms highlighted in blue.

score	result	highlight
13.517900466918945	"Passengers on a scheduled train out of the mountain..."	"Iris Henderson, who was vacationing in Mandrika with..."
12.473928451538086	"Four seemingly-unrelated men board subway train Pe..."	"Between stations they separate the front car..."
11.462580680847168	"In early afternoon, four armed men hijack a subway tr..."	"They stop on a slight incline, decoupling the first ..."
11.175505638122559	"On the eve of his release after five years imprisoned, ..."	"Meanwhile the criminal Vogel is transported by tr..."
9.53182601928711	"An American of German descent arrives in post-war ..."	"His uncle gets him a job on the Zentropa train..."
9.53182601928711	"In December 1935, when his train is stopped by deep..."	"In December 1935, when his train is stopped..."
9.53182601928711	"From the time John J. Macreeedy steps off the train in..."	"Macreeedy steps off the train in Black Rock, ..."



Demo





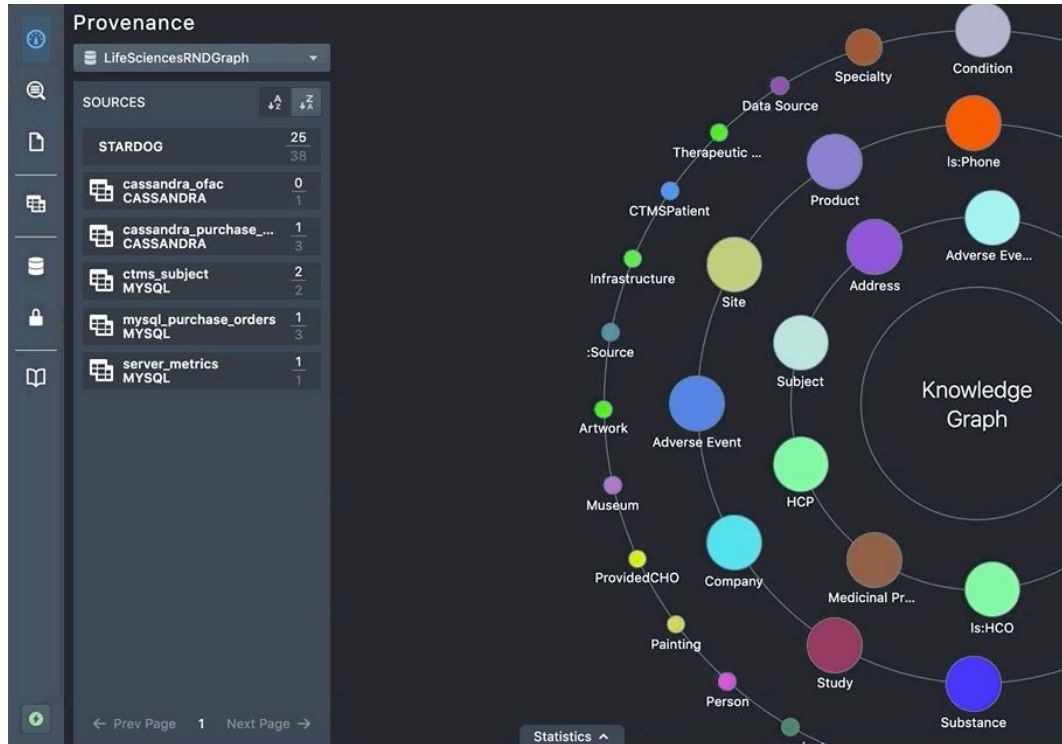
Provenance



Provenance

- Provenance allows you to explore where your data comes from - which databases, virtual graphs, etc. are the source for different classes and properties

Example: Provenance





Demo





Geospatial Queries



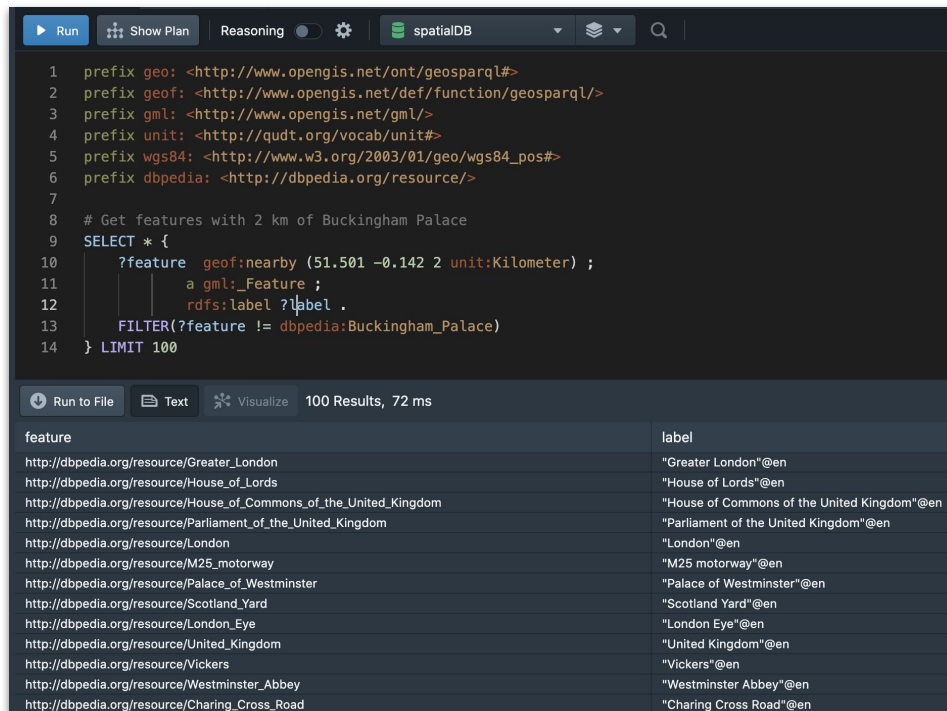
Geospatial Queries

- You can enable geospatial on any Stardog database with geospatial data (e.g., longitude, latitude, etc.) encoded using WGS84 or the OGC's GeoSPARQL vocabulary
- Stardog provides support for the following operators (defined by <http://www.opengis.net/def/function/geosparql/>):
 - geof:relate
 - geof:distance
 - geof:within
 - geof:nearby
 - geof:area
- Datatypes are defined by the QUDT ontology and the OGC units vocabulary <http://www.opengis.net/def/uom/OGC/1.0/>



Example: Geospatial Queries

- Get the points within 2 km of Buckingham Palace (an example of the geo:nearby function)



```
1 prefix geo: <http://www.opengis.net/ont/geosparql#>
2 prefix geof: <http://www.opengis.net/def/function/geosparql/>
3 prefix gml: <http://www.opengis.net/gml/>
4 prefix unit: <http://qudt.org/vocab/unit#>
5 prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#>
6 prefix dbpedia: <http://dbpedia.org/resource/>
7
8 # Get features with 2 km of Buckingham Palace
9 SELECT * {
10   ?feature geof:nearby (51.501 -0.142 2 unit:Kilometer) ;
11   | a gml:_Feature ;
12   | rdfs:label ?label .
13   FILTER(?feature != dbpedia:Buckingham_Palace)
14 } LIMIT 100
```

Run to File | Text | Visualize | 100 Results, 72 ms

feature	label
http://dbpedia.org/resource/Greater_London	"Greater London"@en
http://dbpedia.org/resource/House_of_Lords	"House of Lords"@en
http://dbpedia.org/resource/House_of_Commons_of_the_United_Kingdom	"House of Commons of the United Kingdom"@en
http://dbpedia.org/resource/Parliament_of_the_United_Kingdom	"Parliament of the United Kingdom"@en
http://dbpedia.org/resource/London	"London"@en
http://dbpedia.org/resource/M25_motorway	"M25 motorway"@en
http://dbpedia.org/resource/Palace_of_Westminster	"Palace of Westminster"@en
http://dbpedia.org/resource/Scotland_Yard	"Scotland Yard"@en
http://dbpedia.org/resource/London_Eye	"London Eye"@en
http://dbpedia.org/resource/United_Kingdom	"United Kingdom"@en
http://dbpedia.org/resource/Vickers	"Vickers"@en
http://dbpedia.org/resource/Westminster_Abbey	"Westminster Abbey"@en
http://dbpedia.org/resource/Charing_Cross_Road	"Charing Cross Road"@en



Demo





Stored Query Service (SQS)

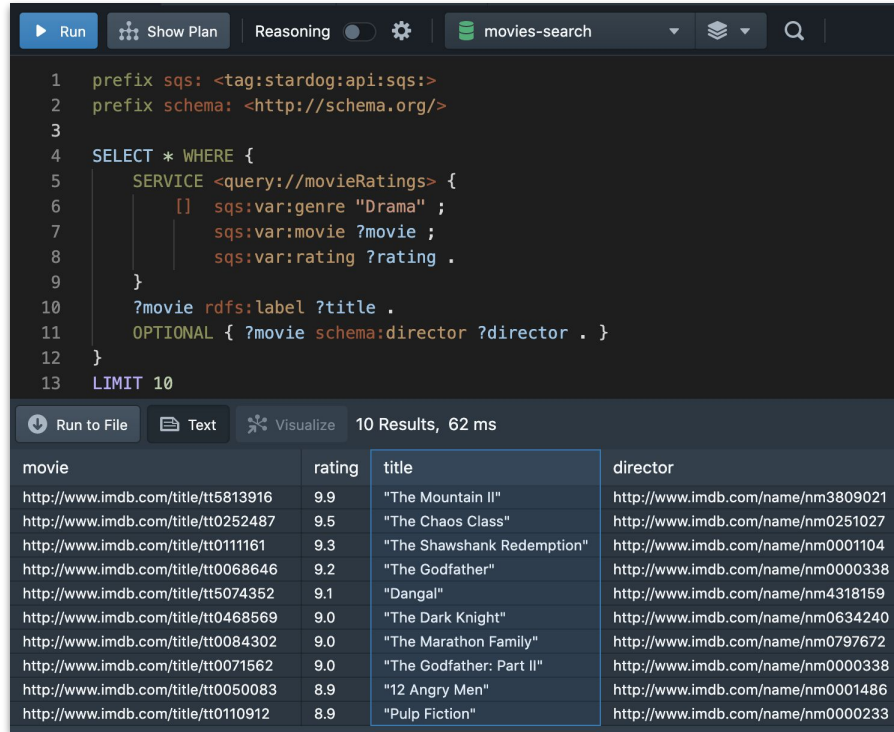


Stored Query Service (SQS)

- Stored queries can be invoked via the SERVICE keyword
- You can choose which stored query variables can be used in the calling query's scope
- You can alias the stored query variable names to another name
- You can statically bind stored query variables
- In addition, you can use the service to call path queries
 - Stardog provides special functions to perform operations on the returned paths. These include length, nodes, any and all



Example: Stored Query Service (SQS)



The screenshot shows a SPARQL query editor interface. The top bar includes buttons for 'Run', 'Show Plan', 'Reasoning', and a search bar with the text 'movies-search'. The query is as follows:

```
1 prefix sqs: <tag:stardog:api:sqs:>
2 prefix schema: <http://schema.org/>
3
4 SELECT * WHERE {
5   SERVICE <query://movieRatings> {
6     [] sqs:var:genre "Drama" ;
7     [] sqs:var:movie ?movie ;
8     [] sqs:var:rating ?rating .
9   }
10  ?movie rdfs:label ?title .
11  OPTIONAL { ?movie schema:director ?director . }
12 }
13 LIMIT 10
```

Below the query, the interface shows '10 Results, 62 ms'. The results are displayed in a table with columns: movie, rating, title, and director.

movie	rating	title	director
http://www.imdb.com/title/tt5813916	9.9	"The Mountain II"	http://www.imdb.com/name/nm3809021
http://www.imdb.com/title/tt0252487	9.5	"The Chaos Class"	http://www.imdb.com/name/nm0251027
http://www.imdb.com/title/tt0111161	9.3	"The Shawshank Redemption"	http://www.imdb.com/name/nm0001104
http://www.imdb.com/title/tt0068646	9.2	"The Godfather"	http://www.imdb.com/name/nm0000338
http://www.imdb.com/title/tt5074352	9.1	"Dangal"	http://www.imdb.com/name/nm4318159
http://www.imdb.com/title/tt0468569	9.0	"The Dark Knight"	http://www.imdb.com/name/nm0634240
http://www.imdb.com/title/tt0084302	9.0	"The Marathon Family"	http://www.imdb.com/name/nm0797672
http://www.imdb.com/title/tt0071562	9.0	"The Godfather: Part II"	http://www.imdb.com/name/nm0000338
http://www.imdb.com/title/tt0050083	8.9	"12 Angry Men"	http://www.imdb.com/name/nm0001486
http://www.imdb.com/title/tt0110912	8.9	"Pulp Fiction"	http://www.imdb.com/name/nm0000233



Demo





Named Graph Aliases



Named Graph Aliases

- Aliases can point to any named graph - local or virtual
- Queries written using aliases do not need to change even though the alias is updated
- Can be used in Read and update queries (via the FROM/FROM NAMED and USING/USING NAMED keywords)
- Aliases honor Named Graph Security
- They *cannot* be used with GRAPH keywords, CONSTRUCT, INSERT, or UPDATE templates or ADD/DROP/CLEAR/COPY/MOVE queries. Nor can you use an alias to point to another alias

Example: Named Graph Aliases

```
1 # create the alias production:movies for example:movies
2 insert data {
3   graph <tag:stardog:api:graph:aliases> {
4     <production:movies> <tag:stardog:api:graph:alias> <example:movies> .
5   }
6 }
7
```

Create an alias for a named graph and then use it queries

Simply repoint the alias at another named graph when you need to

```
1 select * from <production:movies> {
2   ?movie rdfs:label ?title .
3 }
4
5 LIMIT 100
6
```

Run to File Text Visualize 100 Results, 35 ms

movie	title
http://www.imdb.com/title/tt0010323	"The Cabinet of Dr. Caligari"
http://www.imdb.com/title/tt0012349	"The Kid"
http://www.imdb.com/title/tt0219854	"The Kid"
http://www.imdb.com/title/tt0013442	"Nosferatu"
http://www.imdb.com/title/tt0014429	"Safety Last!"
http://www.imdb.com/title/tt0015324	"Sherlock Jr."
http://www.imdb.com/title/tt0015648	"Battleship Potemkin"
http://www.imdb.com/title/tt0015864	"The Gold Rush"
http://www.imdb.com/title/tt0016847	"Faust"
http://www.imdb.com/title/tt0017136	"Metropolis"
http://www.imdb.com/title/tt0293416	"Metropolis"
http://www.imdb.com/title/tt0017925	"The General"
http://www.imdb.com/title/tt0499556	"War"
http://www.imdb.com/title/tt0010323	"The Cabinet of Dr. Caligari"





Demo





Edge Properties



Edge Properties

- Edge properties allow users to attach information to RDF statements by using them as the subject of another RDF statement
- Based on RDF*/SPARQL* extensions
- Bridges the gap between RDF data models and Property graph data model

Example: Edge Properties

edgePropsDB

```
1 SELECT * {  
2   << ?emp a :Engineer >> :since ?year .  
3   << ?emp :worksAt :Stardog >> :source ?who .  
4 }
```

Run to File Text Visualize 3 Results, 33 ms

emp	year	who
:Pete	:2010	:HR
:Joe	:2019	:HR
:Bob	:2015	:HR

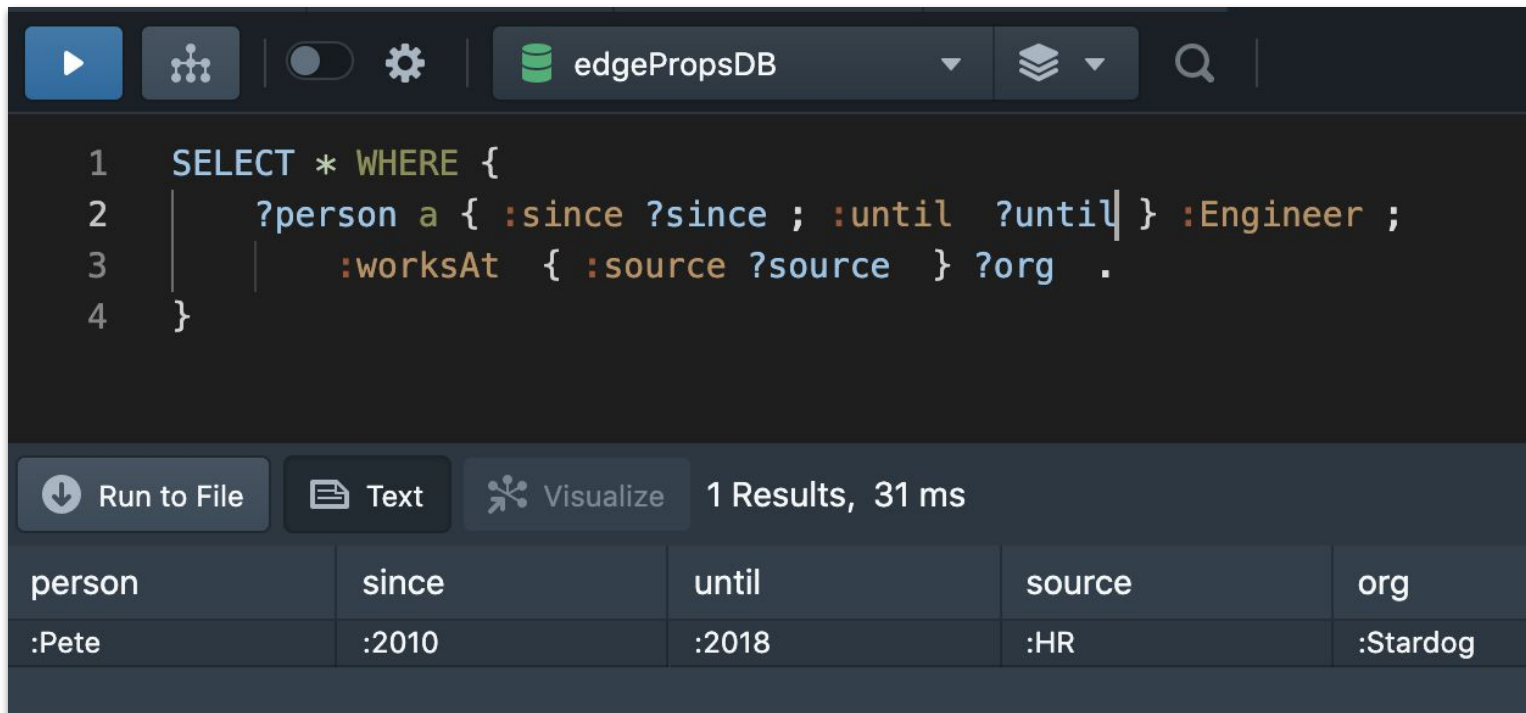
edgePropsDB

```
1 SELECT ?emp ?year {  
2   BIND(<< ?emp a :Engineer >> as ?edge)  
3   ?edge :until ?year .  
4 }
```

Run to File Text Visualize 1 Results, 42 ms

emp	year
:Pete	:2018

Example: Edge Properties



The screenshot shows a query editor interface with a dark theme. At the top, there is a toolbar with icons for play, graph, toggle, settings, a database icon labeled 'edgePropsDB', a stack icon, and a search icon. Below the toolbar, a SPARQL query is entered in a text area:

```
1 SELECT * WHERE {  
2   ?person a { :since ?since ; :until ?until } :Engineer ;  
3   :worksAt { :source ?source } ?org .  
4 }
```

Below the query area, there is a row of buttons: 'Run to File' (with a download icon), 'Text' (with a document icon), and 'Visualize' (with a graph icon). To the right of these buttons, it says '1 Results, 31 ms'. Below this row is a table with 5 columns: 'person', 'since', 'until', 'source', and 'org'. The table contains one row of data:

person	since	until	source	org
:Pete	:2010	:2018	:HR	:Stardog





Demo





Learning Objectives



Learning Objectives



Use PATHS queries to discover the many ways 2 nodes in your graph can be connected



Use full text-search with Stardog data



Discover where your data comes from using Studio's Provenance feature



Explore your spatial data using Stardog's GeoSPARQL support



Centrally manage your queries using the Stored Query Service (SQS)



Write flexible queries using Named Graph Aliases



Bridge the gap between Property Graphs and a Knowledge Graph by leveraging Edge Properties (a RDF*/SPARQL* extension)





Thank you



STARDOG ACADEMY