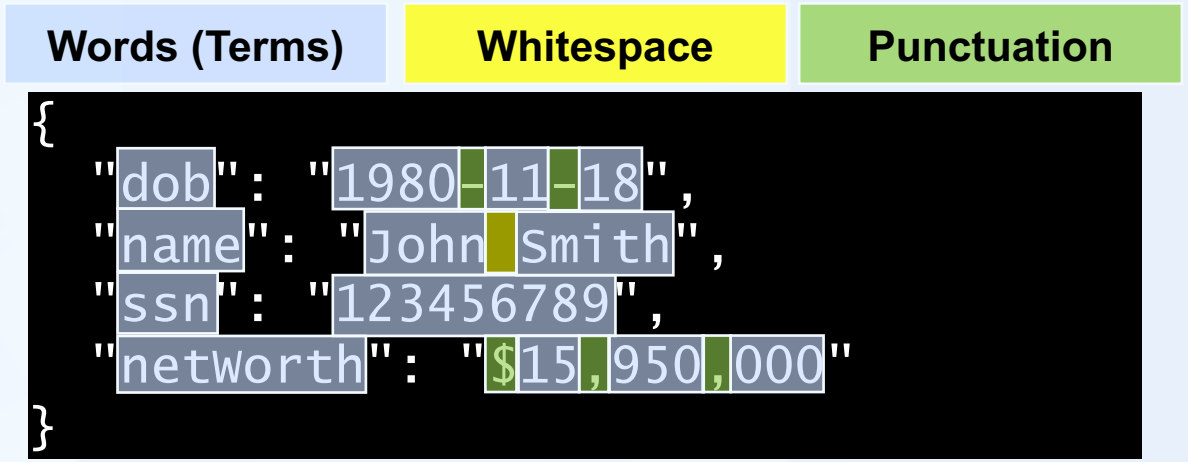
Search

Key Concepts in this Unit

- Term lists and how queries are resolved.
- Stemming, Snippeting and grammar.
- How indexes help search speed and accuracy.
- How they can be configured.
- Searching documents, triples, and rows.
- Introduction on the Optic API.

Concept: Terms/Words

- When documents are loaded into a MarkLogic database it indexes:
  - The “terms” for element/property names and element attribute names.
  - All ”terms” in a document, as shown on the right.
- Whitespace and punctuation is ignored.



Scenario

Given the following documents, how does this tracking work?

Document #1

<description>

Jack ran to the store.

</description>

Document #2

<description>

Jill runs to the store.

</description>

Document #3

<description>

Jack drives to the market.

</description>

Document #4

<description>

Jill, running up the hill.

</description>

Concept: Universal Index

- Built and updated when all documents are loaded into the MarkLogic database.
- Keeps track of which documents contain:
  - A given word.
  - A given property/element name.
- An “inverted” index because “terms” are used as “keys” (left-most column) instead of document/fragment ids.
- Loaded into memory as needed and saved to disk.

TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
ran	1			
runs		2		
running				4
drives			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

Concept: AND Query

- Makes use of list intersections.
- Query: Which documents contain the words “jill” and “hill”?

TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
ran	1			
runs		2		
running				4
drives			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

Concept: OR Query

- Makes use of list union.
- Query: Which documents contain the words “jill” or “hill”?

TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
ran	1			
runs		2		
running				4
drives			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

Concept: NOT Query

- Makes use of list subtraction.
- Query: Which documents contain the word “jack” but not “store”?

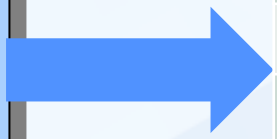
TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
ran	1			
runs		2		
running				4
drives			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

Concepts: Structure + Element Values

- Document Structure is indexed, which yields fast resolution of Xpath.

Document #5

```
<bookstore>
  <book>
    <title>
      Moby Dick
    </title>
    <author>
      H. Melville
    </author>
  </book>
</bookstore>
```



TERM	DOCUMENT SET			
bookstore/book	5			
book/title	5			
book/author	5			
<title>:Moby Dick	5			
<author>:H. Melville	5			

Progress MarkLogic

Lab: Basic Search

Concept: Filtering

Question: How does a person determine which documents contain the phrase “to the store”?

Document #1

<description>

Jack ran to the store.

</description>

Document #2

<description>

Jill runs to the store.

</description>

Document #3

<description>

Jack drives to the market.

</description>

Document #4

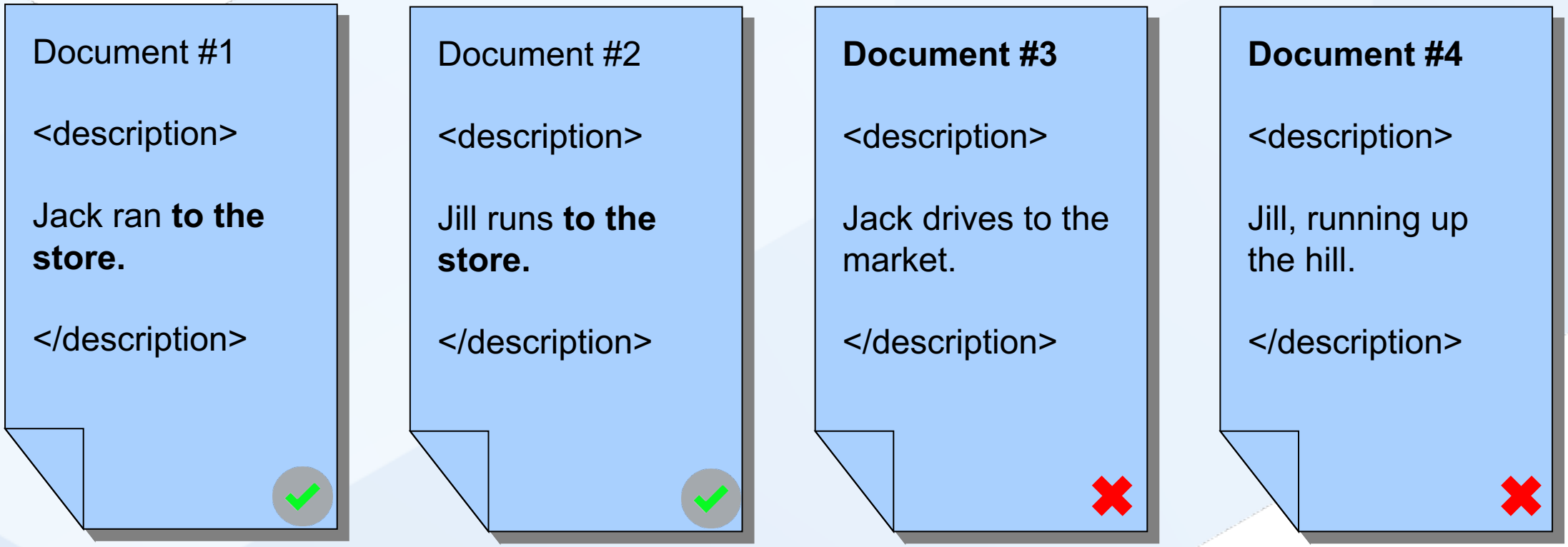
<description>

Jill, running up the hill.

</description>

Concept: Filtering

Answer: A person needs to read each document looking for the phrase "to the store" (Filtering).



Concept: Unfiltered vs Filtered Search

- Unfiltered Search:
  - Resolves queries using **configured indexes only.**
  - Fast but potentially inaccurate.
    - Especially with phrases, wildcards, and proximity searches.
  - Accuracy can be improved when enough indexes are enabled.
- Filtered Search:
  - Resolves queries by using indexes to identify candidate documents and then opens and reads each document to validate.
  - Accurate but can be slower than Unfiltered search – even with pagination.
- Goal: make unfiltered searches accurate.
  - Compare results of `xdmp.estimate` (unfiltered) against `fn.count(cts.search)` (filtered).

Concept: Stemming disabled

- Without stemming, words are tracked based on actual words that appear on the documents.
- Search for “ran” matches “ran”, but not “runs” or “running”.

TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
ran	1			
runs		2		
running				4
drives			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

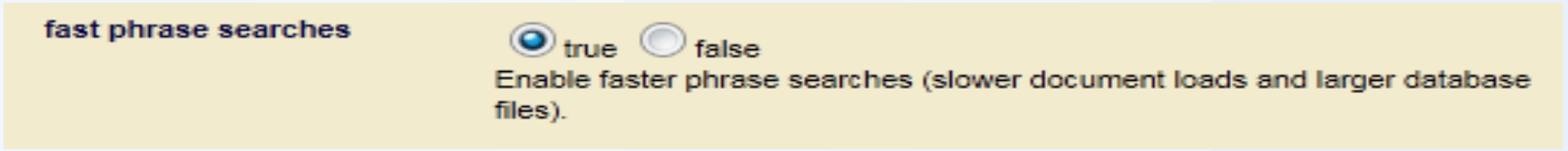
Concept: Stemming Enabled

- The “root word” (stem) is indexed instead of the actual word.
- ”run” is the root word for “ran”, “runs” and “running”.
- Stemming also depends on the configured database language:
  - For English, “run” is the root word for “ran”, “runs” and “running”, but no applicable root word for German, French or Spanish.
  - Non-english requires an additional license.

TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
run	**1**	**2**		**4**
drive			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

Concepts: Phrases

- Administration Tool → Databases → *YourDatabase* → Configure
- Phrase “to the store” is in Documents 1 and 2.



Document #1

<description>

Jack ran **to the store**.

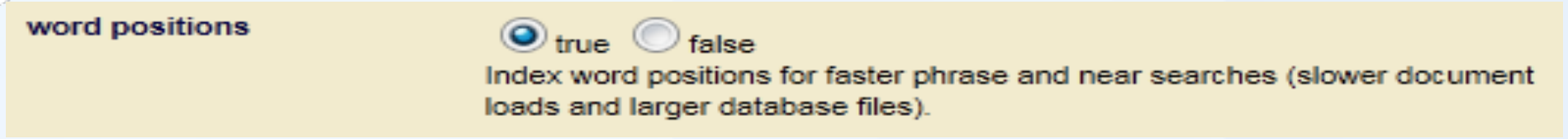
</description>

TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
run	1	2		4
to	1	2	3	
the	1	2	3	4
store	1	2		
jack run	1			
run to	1			
to the	1	2	3	
the store	1	2		

Concepts: Proximity

- Administration Tool → Databases → *YourDatabase* → Configure

- "to the store" found in Documents 1 and 2 and the words are in the same sequence.



Document #1

<description>

Jack ran **to the store**.

</description>

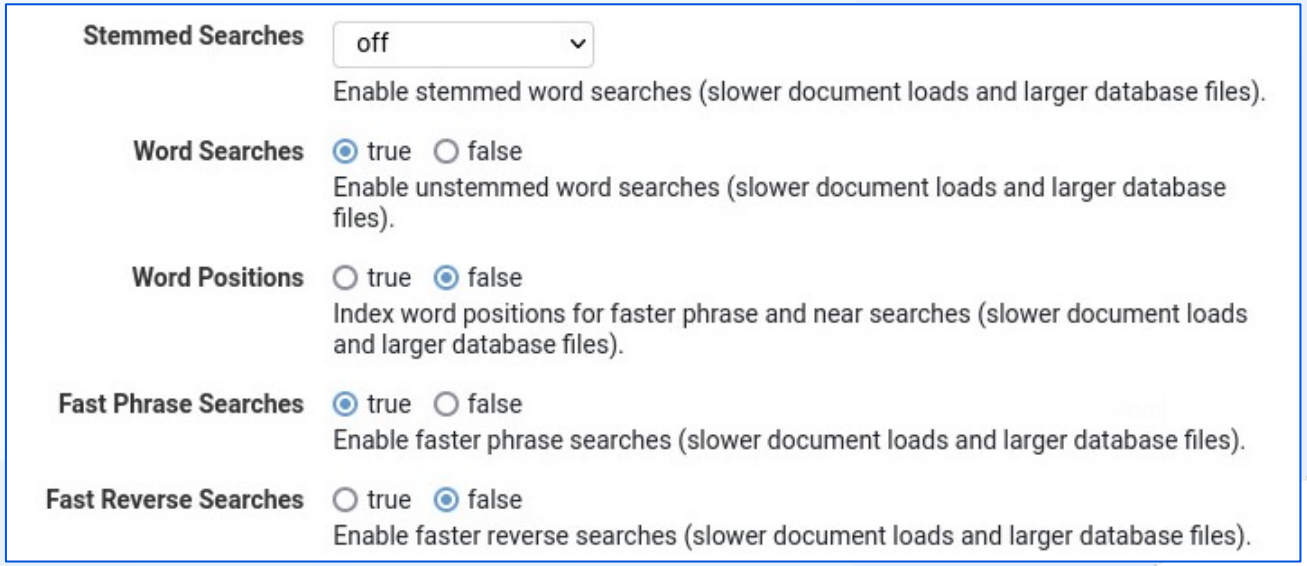
TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1:1		3:1	
run	1:2	2:2		4:2
to	**1:3**	**2:3**	3:3	
the	**1:4**	**2:4**	3:4	4:4
store	**1:5**	**2:5**		

Indexing Concepts: Hashing

- Reduces text to a smaller integer representation.
- Reduces size on disk and is used for all term list keys.
- Sizing:
  - Document + indexes can be smaller than source file.
    - Loaded document is compressed in MarkLogic.
  - More indexes enabled means the overall index size could be larger than source data.
  - How to determine for your project?
    - Configure desired indexes.
    - Load a representative sample of data.

Concept: Universal Index

- Configured for each database (content, module, etc.)
- Each feature that is enabled translates to a separate term list.
- **WARNING:** More indexes translates to:
  - Slower document insert.
  - Higher disk space requirements.
- Unlikely that all will be enabled.



Progress MarkLogic

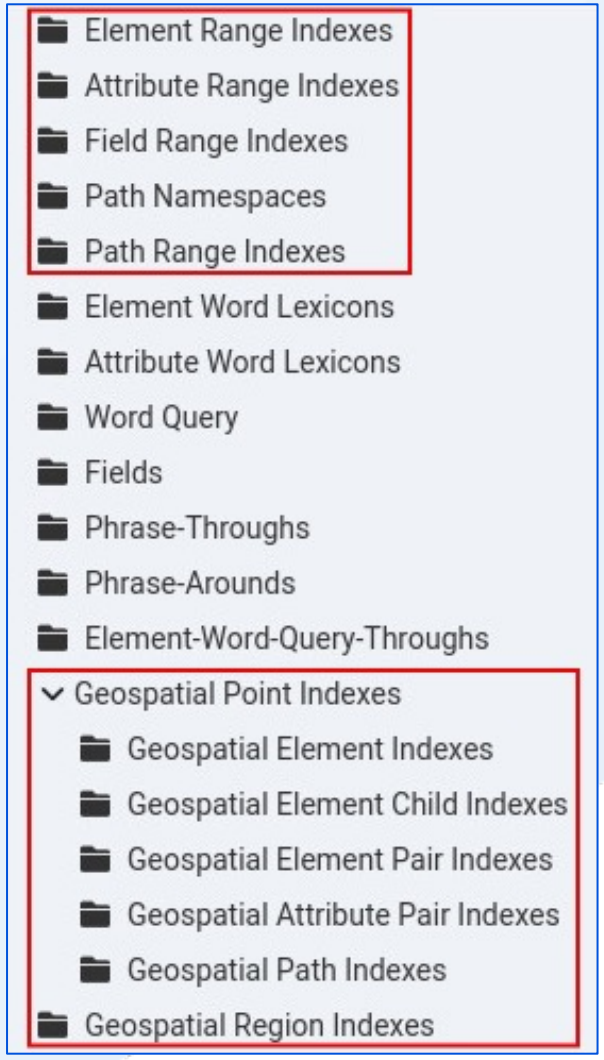
Lab: Advanced Word Search

Limitations of Term Lists

- For XML documents, everything is a string.
  - "1.0“ (string) != 1 (integer)
- Query is limited to “Yes/No”.
  - Does the document contain the word “market”?
- Can**not** answer “range” type of questions.
  - Does the document have property “age” with value greater than 18?

Range Index

- Configured for each database.
- Data type aware
  - Find documents whose “price” is greater than 500.
  - Find documents whose “timestamp” is within a year.
  - Find documents whose “geolocation” is within 20 miles of The Eiffel Tower.
- Can be used to enforce rules on documents.
  - "price" should be a decimal.
  - "timestamp" should be a long integer.
  - "geolocation" should be a decimal pair.



Range Index and Collations

- Collations only apply to strings and there are two types:
  - Default:

RANGE(artist, default collation)	
Madonna	C
the Beatles	A
the beatles	B

  - Custom: Configured to ignore whitespace, punctuation and/or case differences.

RANGE(artist, punctuation, whitespace & case insensitive collation)	
Madonna	C
the Beatles	A, B

A

```
<top-song>
 <artist>the Beatles</artist>
 <title>Yesterday</title>
 <date>1965-10-30</date>
</top-song>
```

B

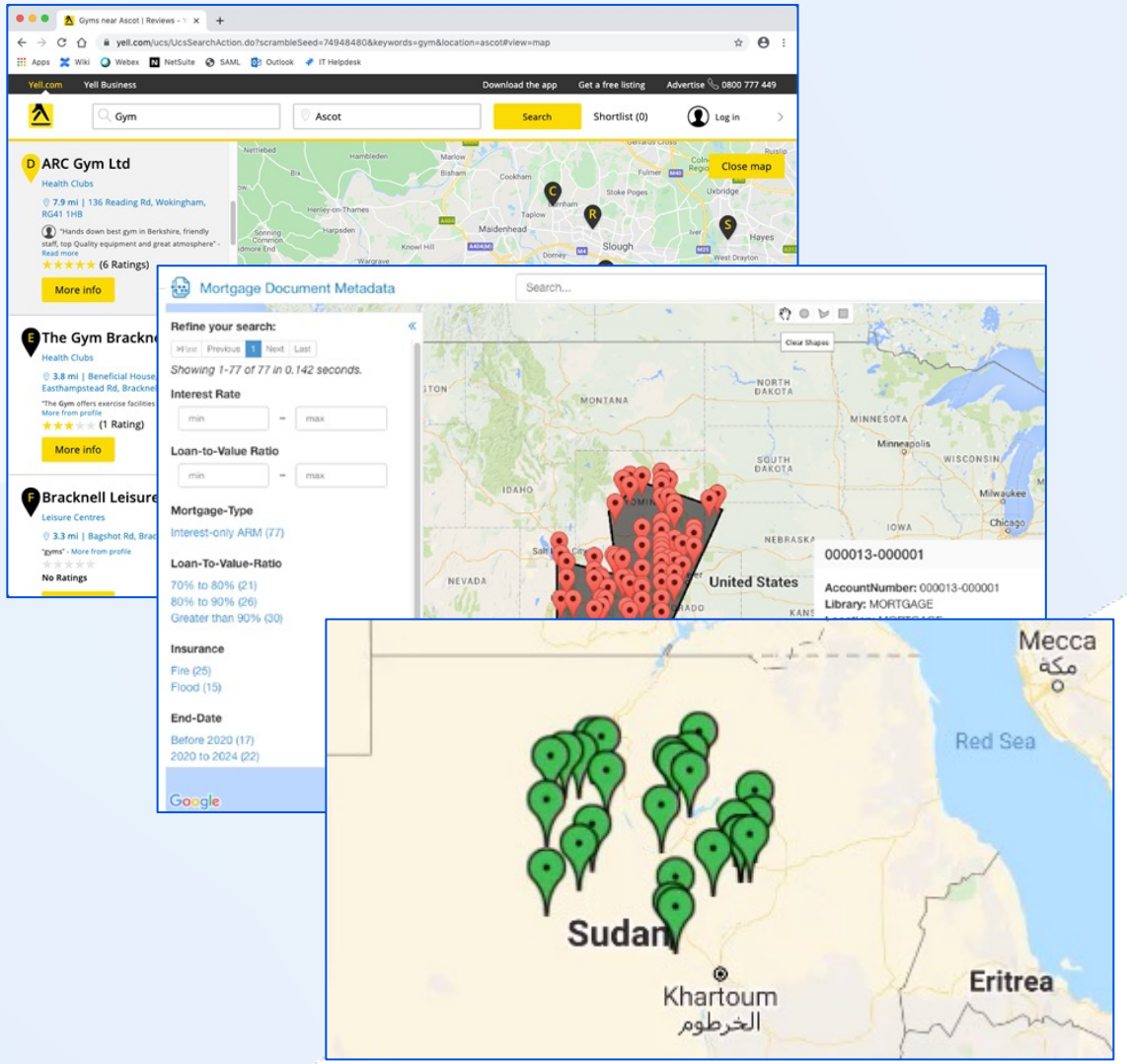
```
<top-song>
<artist>the  beatles</artist>
 <title>Help!</title>
 <date>1965-09-18</date>
</top-song>
```

C

```
<top-song>
 <artist>Madonna</artist>
 <title>Take a Bow</title>
 <date>1995-04-08</date>
</top-song>
```

Geospatial Indexes

- Speed up geospatial search with built-in support for points, lines, circles and polygons.
- Allows for combination of geospatial and word search.
- Latitude and longitude are supported
- Elevation not supported.

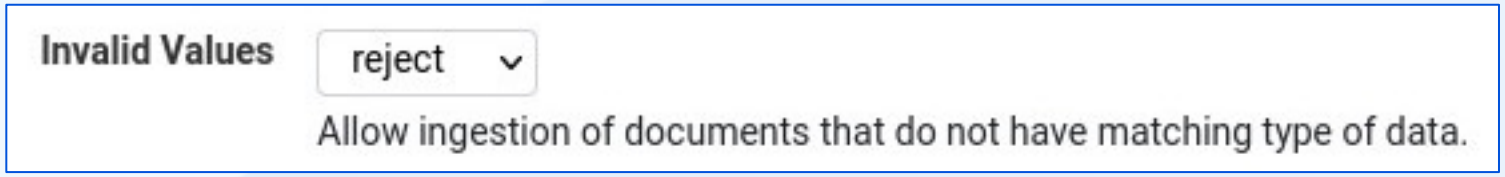


Range Indexes – Other Uses

- Fast sorting of candidates/results.
  - The list of candidates can be sorted using indexed values before document retrieval.
- Value extraction and Faceting.
  - Retrieve a list of values for that property / element / attribute along with the number of documents that contain those values.
- Tiered storage.
  - Based on an indexed value, conditional routing of documents to a specific forest or host with a high-performance media storage type, for example, Solid State Drive (SSD).

Concerns

- Ranged indexes must remain loaded in memory (not get swapped out).
  - Results in faster query resolution and sorting.
  - Has higher RAM requirements (not part of the list cache).
- More indexes (ranged or not) translates to:
  - Slower document insert.
  - Higher disk space requirements.
- All applicable range indexes must be satisfied, or the entire insert/update fails.



Progress MarkLogic

Lab: Ranged Search

Search Results and Relevance Order

- When performing a full text query, the results are returned in Relevance Order.
- Relevance is determined by a mathematical construct.
- It starts with simple concepts:
  - Documents with more matches are more relevant.
  - Shorter Documents with the same number of matches are more relevant than longer documents.
- There are a number of mathematical formulas that determine Relevance Order.

Relevance Order

DOCUMENT 1

```
<doc>

  101 Dalmatians is
  a fun story about
  dogs escaping
  from Cruella
  Deville."

</doc>
```

DOCUMENT 2

```
<doc>

  Fun, Fun, Fun is
  a song by
  the Beach Boys.

</doc>
```

DOCUMENT 3

```
<doc>

  Dogs are fun.

</doc>
```

DOCUMENT 4

```
<doc>

  Raining cats and
  dogs.

</doc>
```

Query: Find documents that contain the word “fun”.

Which documents match?
Which documents are the ***best*** matches?

Relevance Scoring

- The Relevance Order is determined by the Relevance Score formula:

  - Score – A calculated value for each item returned in the search query result set.
  - TF – Term Frequency
  - IDF – Inverse Document Frequency

Relevance Order

- Term Frequency:
  - How often a term occurs within a specific fragment (document) of the result set.
  - Normalized relative to total words in document: ***term density***.
  - Example: Search for “dog”

Document #1
- “dog” occurs 10 times
- 100 total words in document
- Density 10%

Document #2
- “dog” occurs 100 times
- 10,000 total words in doc
- Density 1%

Relevance Order

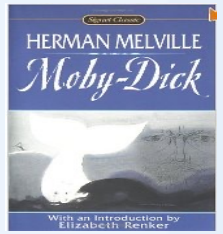
- Inverse Document Frequency
  - IDF = (1/DF)
  - Example: Search for “cat” OR “dog”
    - 100 documents contain “cat”. IDF = 1/100 = .01 (lower)
    - 2 documents contain “dog”. IDF = 1/2 = .5 (higher)
    - Assume a cat and dog document both have the same LOG(TF) value.
    - Which document would receive the highest overall score?

Impacting Relevance: Quality

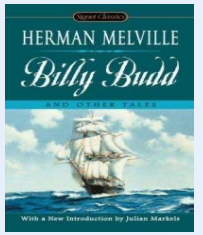
Score = LOG (TF) X IDF

\+ Query Weight X Document Quality

- Quality is:
  - A factor to increase a documents Relevance score
  - Set upon ingestion (or modified later)
  - Default is 0



Q = 3



Q = 0

Search:
author = Herman Melville

1. Moby Dick
2. Billy Budd

Weighting Example

- **QW** = Query Weight, **DQ** = Document Quality
- Search = cat OR dog, QW=1

DOCUMENT 1
WEIGHT = 10

```
<title>
  North American
  Field Guide to
  Mammals.
</title>
<data>
  Domesticated cats
  and dogs arrived
  in current day
  USA in…
</data>
```

DOCUMENT 1
WEIGHT = 0 (default)

```
<title>
  Bob's Blog About
  Life with His Dog
  Rufus
</title>
<data>
  We saw a cat on
  our walk today.
</data>
```

```
score=log(TF)*IDF + (QW*DQ)
```

```
scoreD1= ## + (1*10)
```

```
scoreD2= ## + (1*0)
```

Impacting Relevance: Word Query

- Why does the Coldplay song appear first?

Impacting Relevance: Word Query

- Why does the Coldplay song appear first?

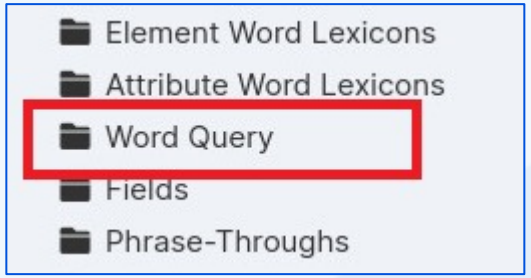
Included Elements

Localname(s)	Namespace	Attribute	Attribute Namespace	Value	Weight	
artist	http://marklogic.com/MLU/top-songs				4	[delete]
title	http://marklogic.com/MLU/top-songs				4	[delete]
descr	http://marklogic.com/MLU/top-songs				0.75	[delete]

Excluded Elements

Localname(s)	Namespace	Attribute	Attribute Namespace	Value	
format	http://marklogic.com/MLU/top-songs				[delete]
length	http://marklogic.com/MLU/top-songs				[delete]

- Configured at the Database-level



Relevance Ratings

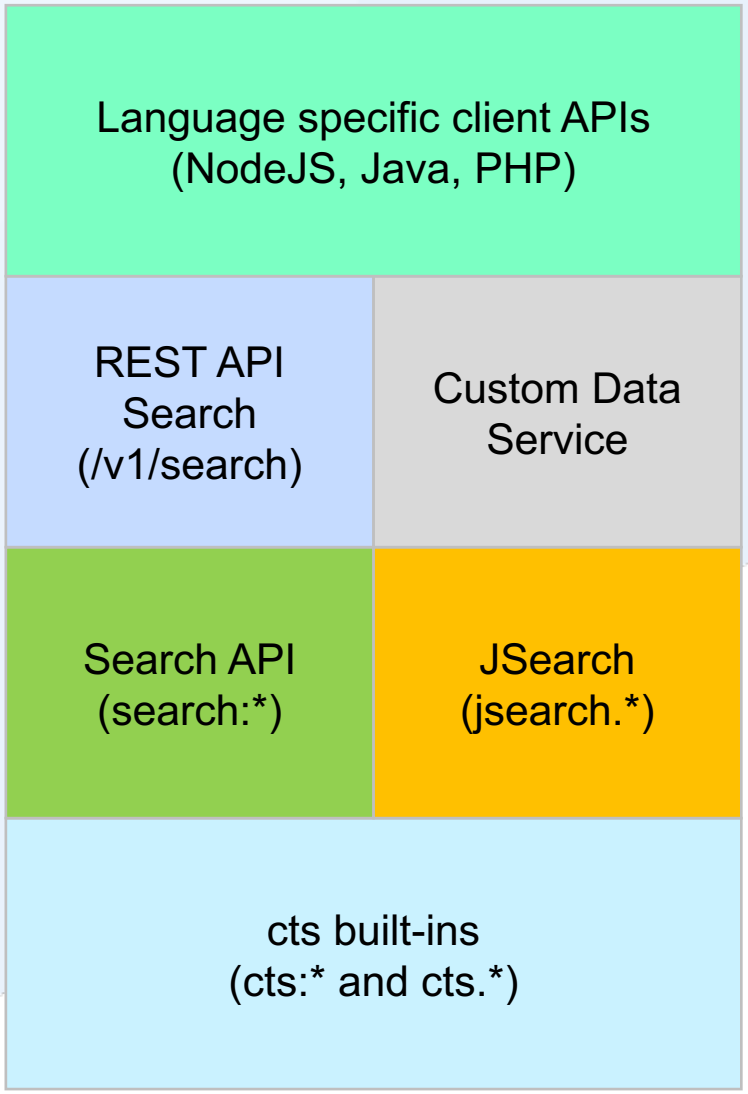
- Score
  - log(TF)*IDF + (QW*DQ)
- Confidence
  - Score affected by average document frequency
  - Bounded between 0 to 1.0
- Fitness
  - Score affected by maximum document frequency
  - Bounded between 0 to 1.0
- More information is available when using `cts:relevance-info`
  - The formulas are simplification of a more complex formula implemented by the product.

Progress MarkLogic

Lab: Relevance

Document Search Libraries

- REST API Search
  - Supported using built-in custom URL rewriter.
  - Uses configuration files to adjust behavior.
- Custom Data Service
  - No rewriter required.
- Search API
  - Meant for XQuery developers.
- JSearch
  - Meant for SJS/MJS developers.
- Both are built on top of native cts library

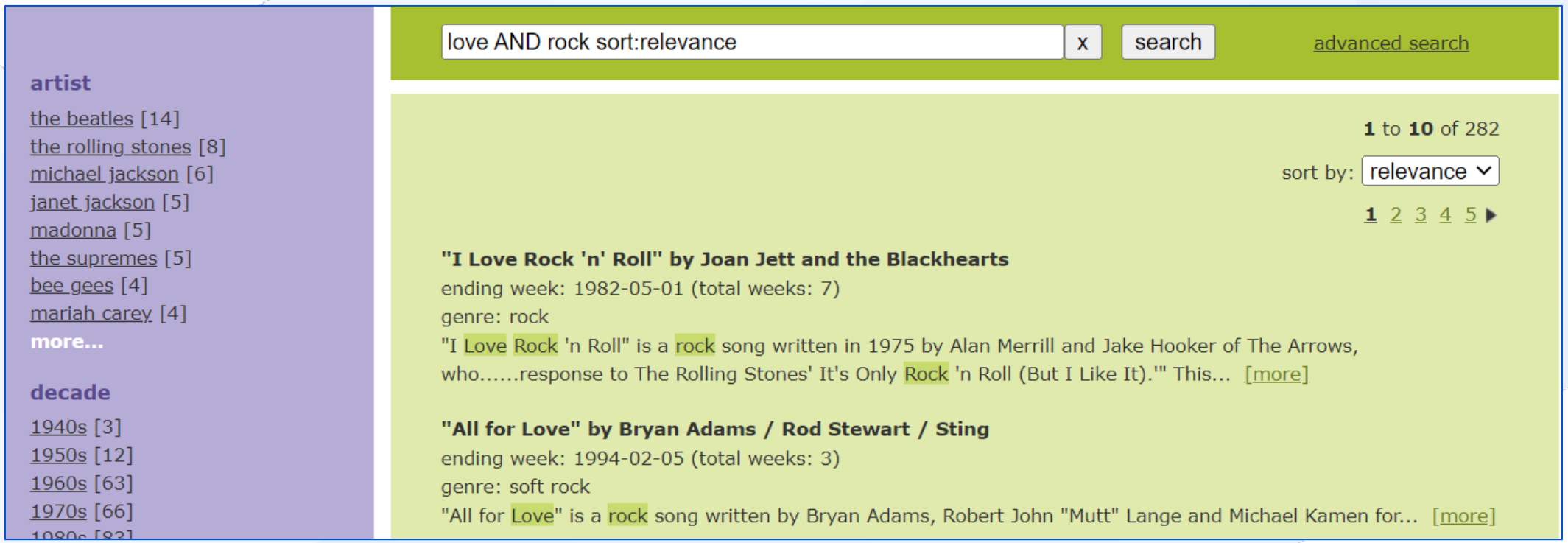


Document Search Libraries

- Document Search libraries enable quick implementation of “google-like” services.

Document Search Libraries

- Snippets, Facets, Pagination, etc.



Search API: Grammar

Type	Example	Description
Word Search	new city	Match the words `new` and the word `city` regardless of sequence on a single property or element
Phrase Search	"new york"	Match the phrase `new york` on a single property or element
AND	new AND city	AND is a reserved word Behaves the same way as `Word Search`
OR	morrow OR city	OR is a reserved word Match the words `morrow` or the word `city`
Negation	new -york	Match the words `new` while ensuring `york` is not present anywhere in the document
Grouping	(jersey OR york) AND new	the words `jersey` or `york` must be present with the word `new` on the same property or element regardless of sequence.

Search API: Grammar

Type	Example	Description
Facet value	sourceName:"coastal employees"	Indexed property/element/attribute must have the value of “coastal employees”
Facet range	baseSalary GT 5000	Indexed property/element/attribute must have a numeric value greater than 5000

- `sourceName` and `baseSalary` has to match an existing <constraint> entry in your <search:options>.
- Case, whitespace and diacritic sensitivity is now controlled by the configured collation.

cts.parse: Grammar

Type	Example	Description
Facet word	sourceName:"coastal"	Indexed property/element/attribute must have the word of “coastal”
Facet value	sourceName="coastal employees"	Indexed property/element/attribute must have the value of “coastal employees”

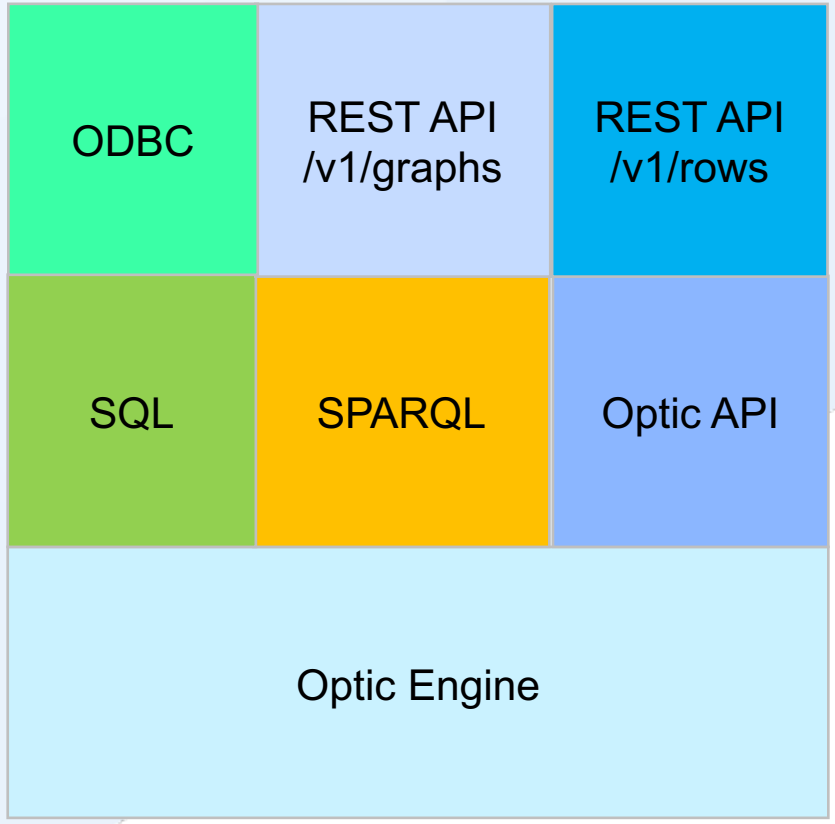
- `sourceName` and `baseSalary` has to match an existing binding.
- Case, whitespace and diacritic sensitivity is now controlled by the configured collation.
- `:` and `=` is the main differentiator between cts.parse and search.parse (Search API library)
- `:` acts as value when mapped to cts.uriReference or cts.collectionReference
- More information available at docs: https://docs.marklogic.com/cts.parse

Progress MarkLogic

Lab: Search API

Optic API and the Optic Engine

- The underlying engine supporting SQL, SPARQL and the Optic API
- The Optic API is available in XQuery, Server-side JavaScript, Java and NodeJS
- The Optic API enables search across triples, rows, indexes, and documents.
    - A more in-depth discussion is included in the Data Services class.

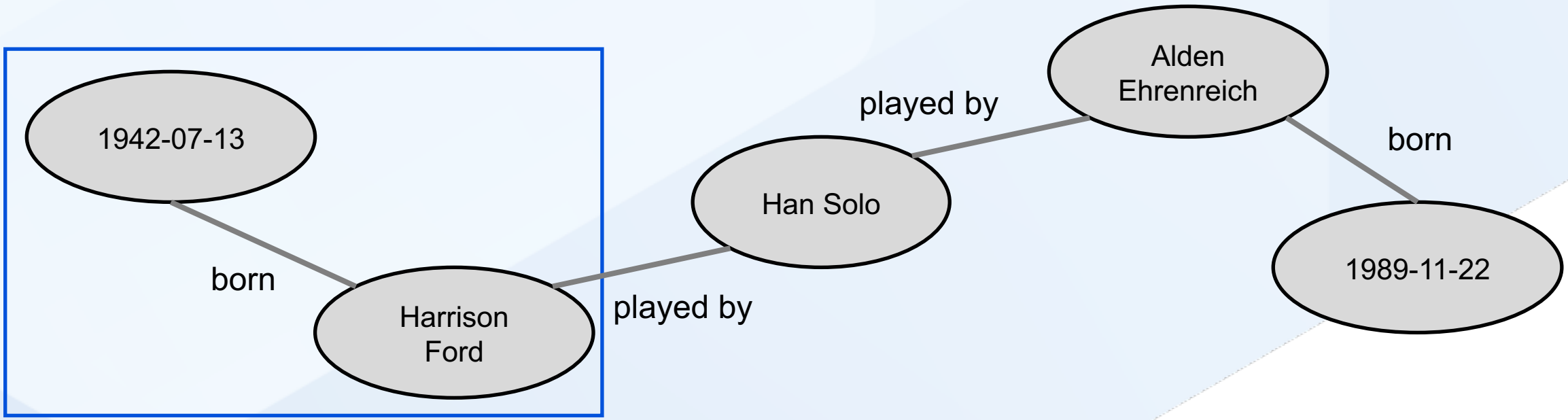


TDE and SQL

- MarkLogic supports SQL92 over views/tables configured using “templates”.
- TDE stands for “Template Driven Extraction”
  - MarkLogic “extracts” data from a document to generate “rows”.
  - Extraction happens in the background.
  - Extracted data are stored as triples.
- Like ranged indexes, templates can be used to enforce document structure and content.

Triples and SPARQL

- SPARQL stands for “SPARQL Protocol and RDF Query Language”.
- Recommended language for searching semantic data.
- Triples best record relationships between facts.



Lab: TDE, SQL, and Optic API

Recap

- Universal index keeps track of words/tokens.
- Ranged indexes are meant to answer data-type specific ranged queries.
- Built-in Search APIs speed up implementation of “google-like” services.
- Unfiltered search can be both fast and accurate when enough indexes are configured.
- Geospatial indexes support latitude and longitude only.
- MarkLogic supports SQL.
- Optic API allows for search across all document types.