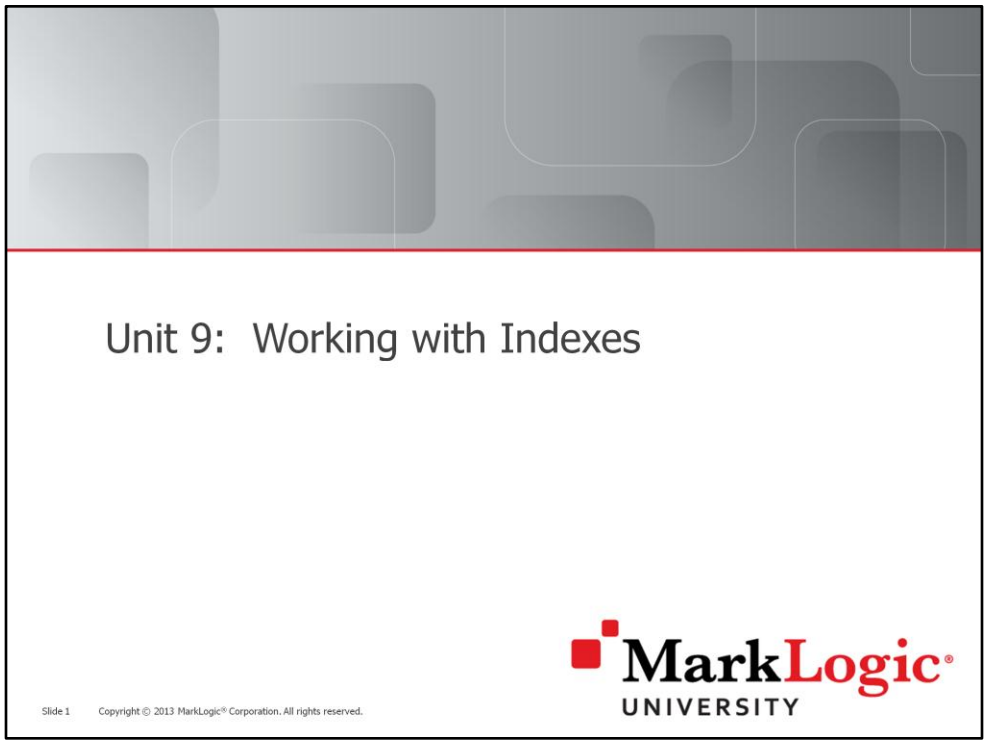
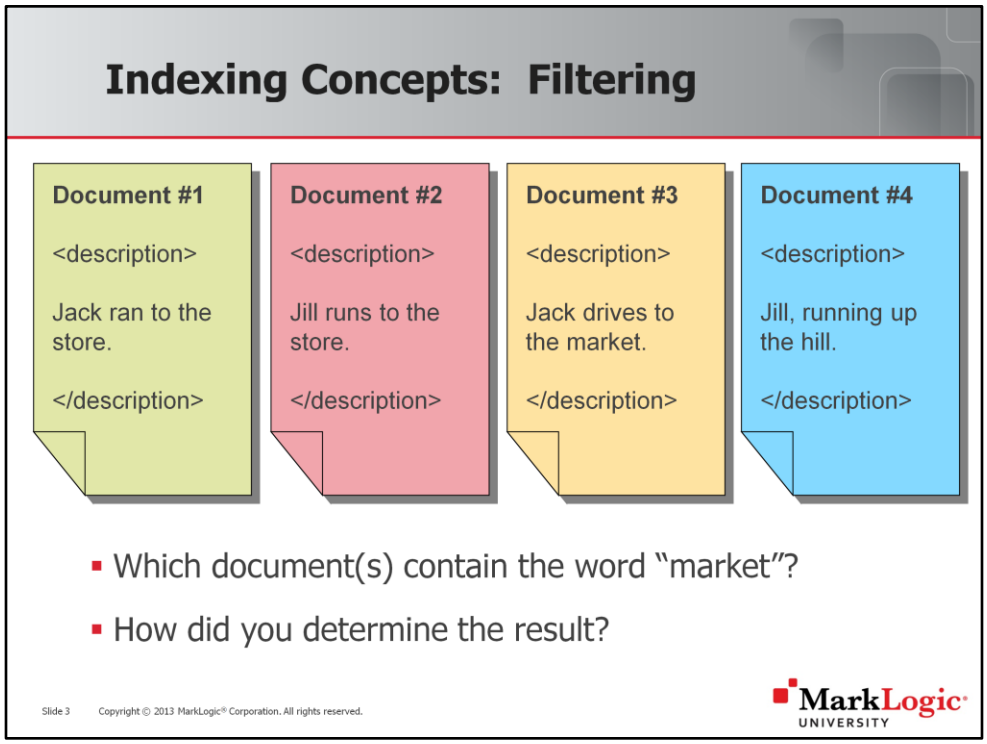


Learning Objectives

- Describe the indexing concepts of filtering, Universal Index, Inverted Index, Term List, Stemming and Hashing
- Implement additional indexes to enhance query performance
- Describe `xdmp:estimate` and `fn:count`
- Create element and attribute range indexes
- Create a lexicon and collation
- Create a word query
- Automate index deployment with the Admin API

MarkLogic UNIVERSITY



Imagine you have these 4 documents in your database. If were asked to tell someone which documents contained a particular word, how would you go about answering that question efficiently?

One approach would be to read each document and look for the desired word. This would be easy with this document set, but would get difficult when more documents are present, and if the documents contained more text. Opening each document and reading it to determine a match is called "filtering" in MarkLogic. Our goal as developers is to configure indexes and develop queries to reduce or eliminate filtering, as it is costly.

Another approach to delivering a fast response to the question would be to create a lookup table, or index. MarkLogic calls this a "term list", and it is a very fast way to resolve a query.

Indexing Concepts: Inverted Index

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

- Which document(s) contain the word "market"?
- How did you determine the result?
- This is called an *Inverted Index*
- Each entry is called a *Term List*

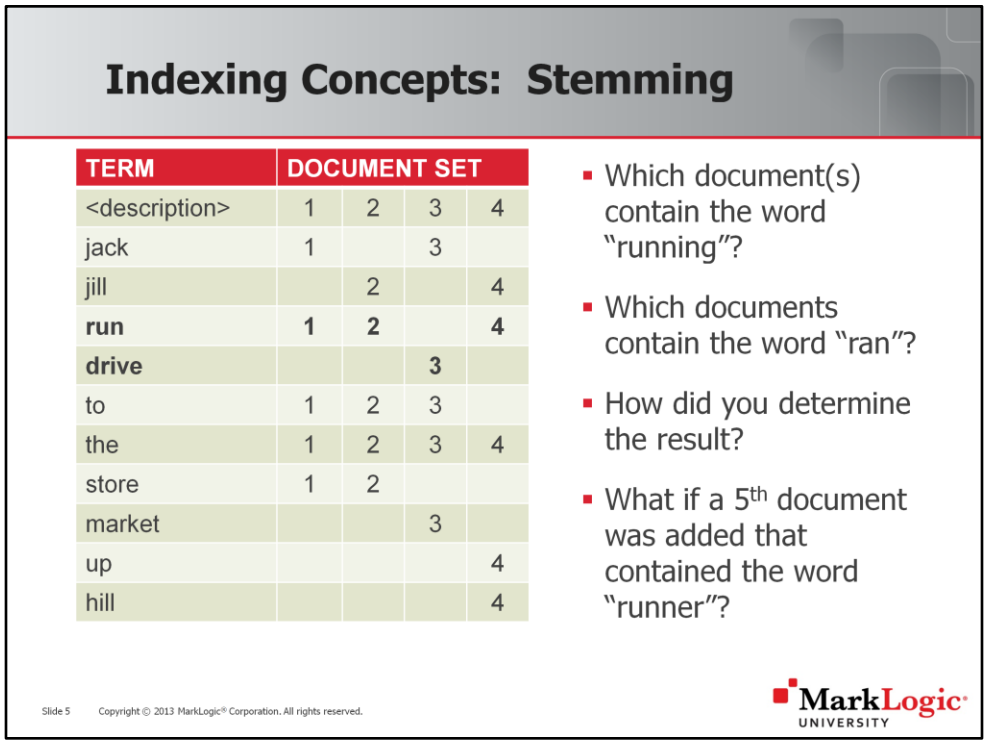
Slide 4

MarkLogic UNIVERSITY

What if the document contents were presented to you in another way? You could now more easily and quickly determing the results. This logic of organization is called an "inverted index" or "term list" and is a good way to think of the structure of the MarkLogic index.

An inverted index can be defined as a list of words, with references to the documents that contain them.

Notice the following:
There is no punctuation stored in the term list, even though the document contains commas and periods.
"Jack" and "Jill" are stored as lowercase.
Notice the redundancy in storing the words "ran", "runs" and "running". In the next slide we see how MarkLogic solves this problem with the concept of stemming.



Stemming helps MarkLogic manage the index more efficiently by reducing term list size. Instead of storing all variations of a word, MarkLogic reduces words down to their root, allowing for smaller term lists and faster performance.

Notes:

- Words from different languages are treated differently, and will not stem to the same root word entry from another language.
- Note: Nouns will not stem to verbs and vice versa. For example, the word "runner" will not stem to "run".
- If you are curious about what a word will stem to, use the cts:stem function.

cts:stem

```
xquery version "1.0-ml";

let $stems := ("ran", "runner")

return cts:stem($stems)
```

```
run
runner
```

Indexing Concepts: AND Query

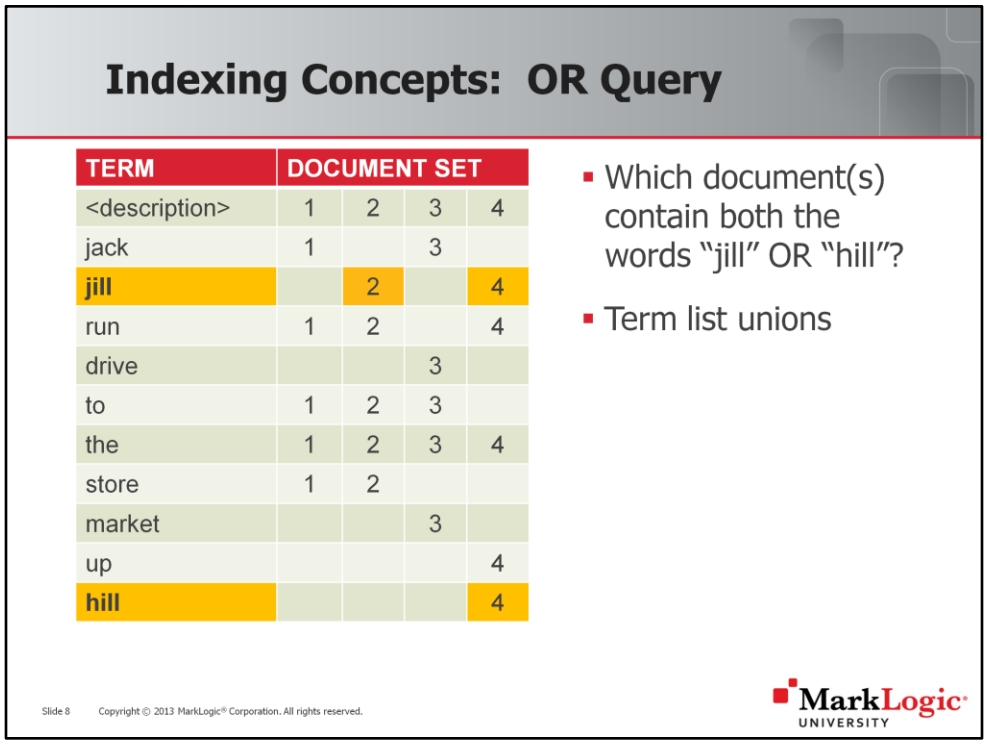
TERM	DOCUMENT SET			
<description>	1	2	3	4
jack	1		3	
jill		2		4
run	1	2		4
drive			3	
to	1	2	3	
the	1	2	3	4
store	1	2		
market			3	
up				4
hill				4

- Which document(s) contain both the words "jill" AND "hill"?
- Term list intersections

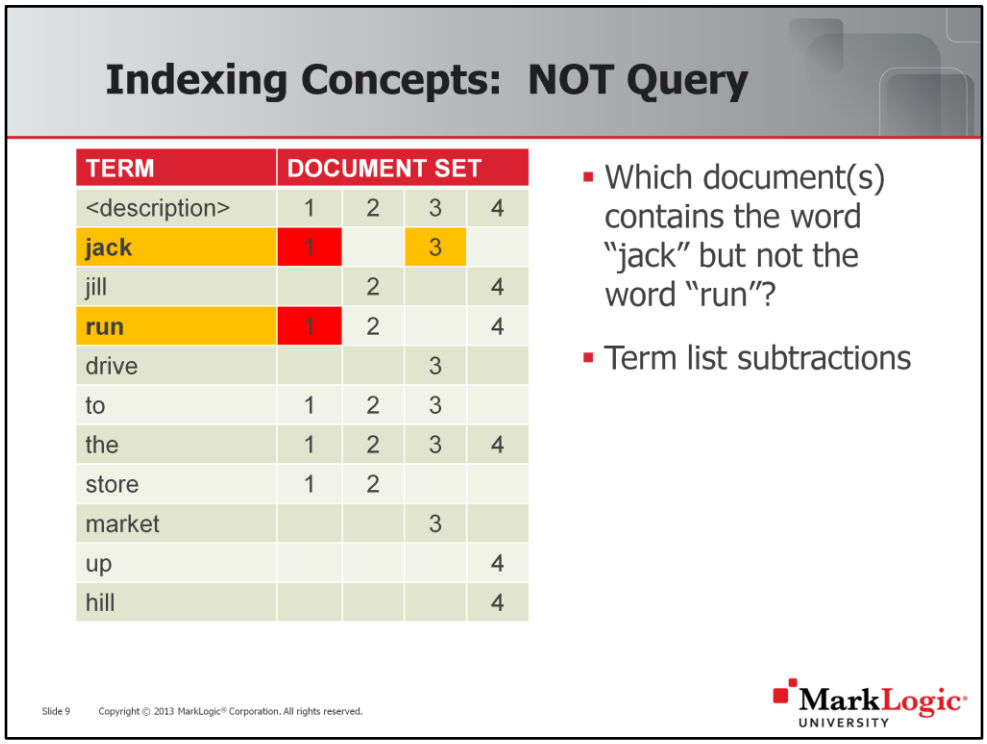
Slide 7

MarkLogic UNIVERSITY

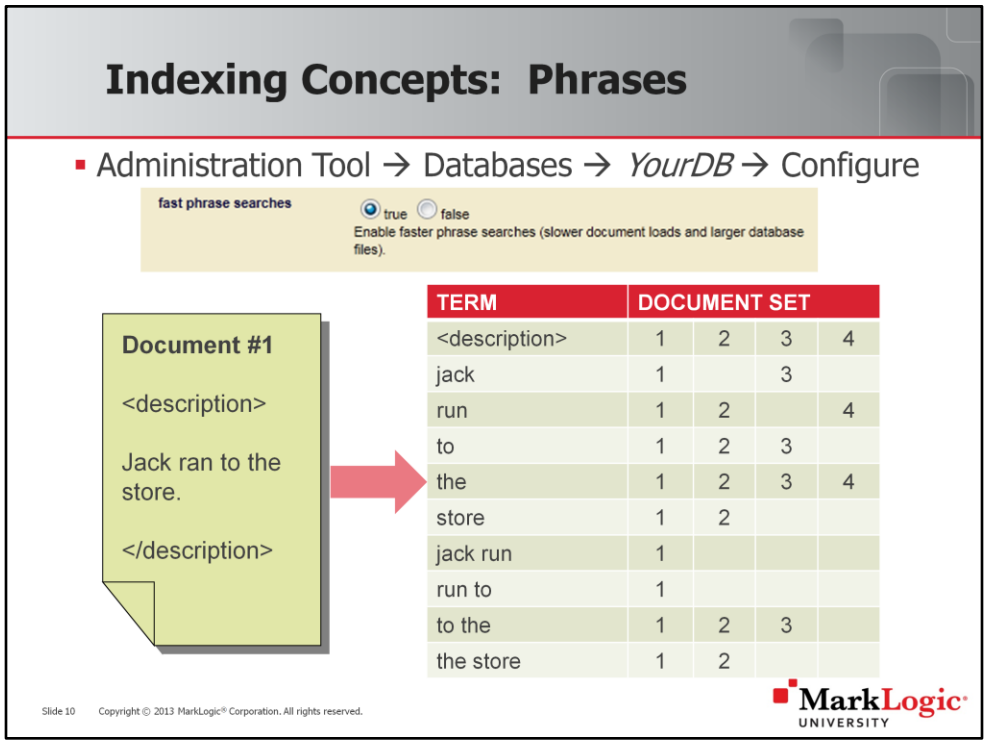
Using the index it makes it very easy to determine documents that contain word pairs by doing an intersection.



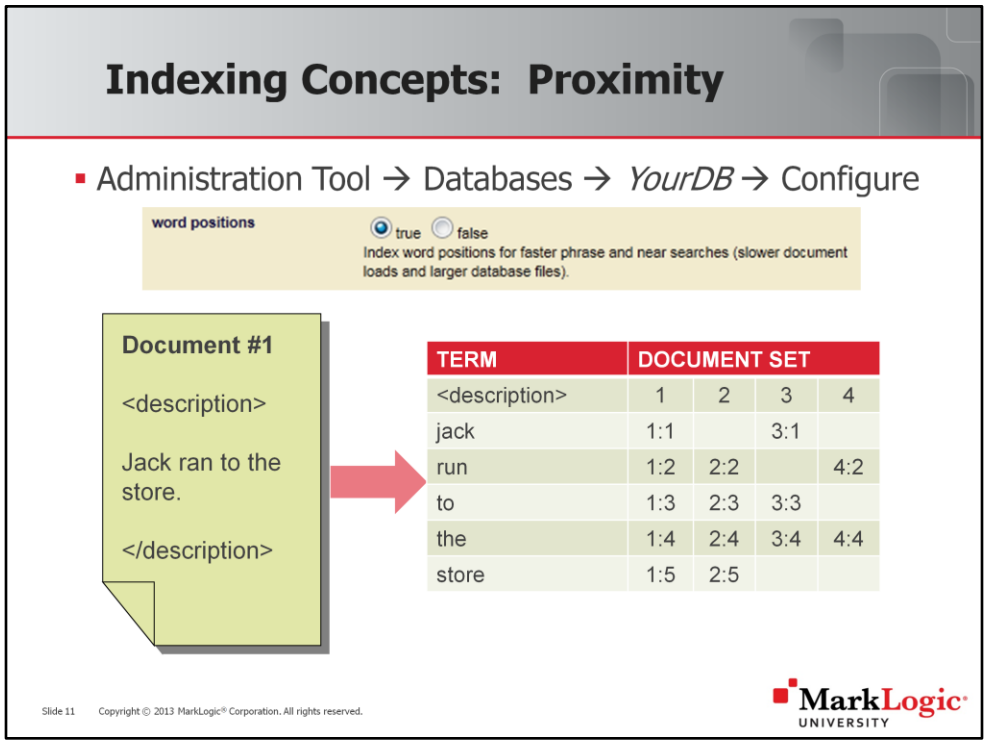
Using the index it makes it very easy to determine documents that contain word pairs by doing an intersection.



Using the index it makes it very easy to determine documents that contain one word but not another by doing a subtraction.



The “Fast Phrase Searches” option adds to word phrases to the inverted index term list making it more likely to identify phrases within a document from indexes as opposed to filtering.



The "Word Positions" option adds position information to the inverted index term list. This makes it possible to determine not only phrases where the words are **next** to each other, but also allows us to search using indexes for words that are **close** to each other.

Indexing Concepts: Structure

- Structure is indexed, including parent child relationships
- Fast resolution of XPath

Document #1

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

TERM	DOCUMENT SET			
bookstore/book	1			
book/title	1			
book/author	1			
<title>:Moby Dick	1			
<author>:H. Melville	1			

MarkLogic UNIVERSITY

Indexing Concepts: Hashing

- To reduce size on disk, hashing is used for all term list keys
- Hashing reduces text down to a smaller integer representation
- Sizing:
  - XML + indexes can be smaller than source file
    - Loaded XML is compressed in MarkLogic
  - With more indexes enabled, size may be 1.5–3 times larger than XML source

MarkLogic UNIVERSITY

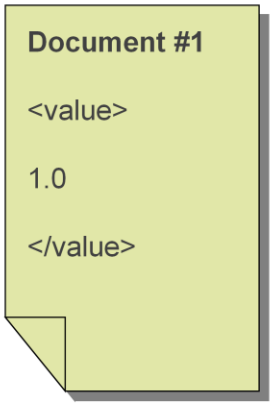
Range Indexes

- Term lists are great at Yes / No type of questions
  - Map Values→Documents
- What about:
  - Find documents where the <price> is less than $50
  - Find documents with a <date> between 1990-01-01 and 1999-12-31
- Range Indexes…
  - Map Values←→Documents
  - Values (typed), not textual matches
  - Fast Range Queries
  - Fast Sorting
  - Fast Value Extraction
  - Faceting
- Range indexes live in memory when MarkLogic starts

MarkLogic UNIVERSITY

Range Index vs. Term List Index

- Term List Query: Find documents containing "1,0"→MATCH
- Term List Query: Find documents containing "1"→MATCH
- Range Index Query: Find documents containing "10"→NO MATCH



Element / Attribute Range Indexes

- Defined on a specific element or attribute
- Defined for a specific data type, sorted in value order

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>
```

RANGE(artist)	
Madonna	C
the Beatles	A
the beatles	B

RANGE(date)	
1965-09-18	B
1965-10-30	A
1995-04-08	C

String Range Indexes & Collation

- Collations apply to String data type range indexes
- Determine what makes a unique value inside the index

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>
```

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

RANGE(artist, punctuation, whitespace & case insensitive collation)	
madonna	C
the beatles	A, B

MarkLogic UNIVERSITY

Path Range Indexes

- More control over what the range index should contain

A
```
<book>
 <title>Moby Dick</title>
 <author>Herman Melville</author>
 <chapter>
   <title>Loomings</title>
   <text>Call me Ishmael…</text>
 </chapter>
 <chapter>
   <title>The Carpet-Bag</title>
   <text>I stuffed a shirt or…</text>
 </chapter>
</book>
```

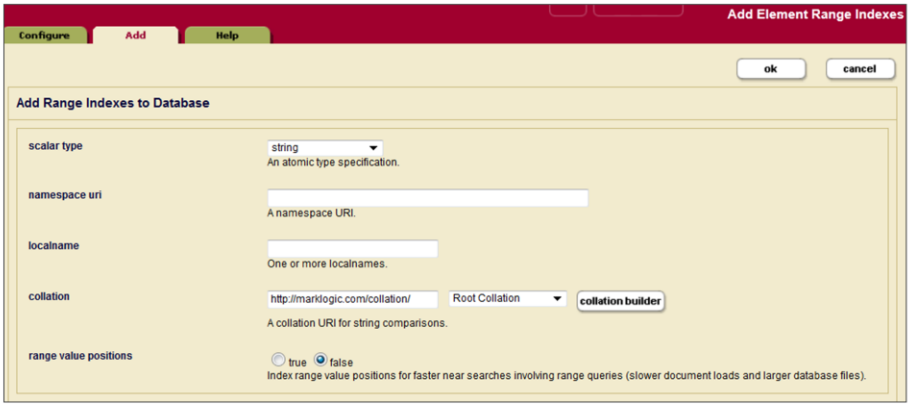
RANGE(title)	
Moby Dick	A
Loomings	A
The Carpet-Bag	A

RANGE(chapter/title)	
Loomings	A
The Carpet-Bag	A

RANGE(book/title)	
Moby Dick	A

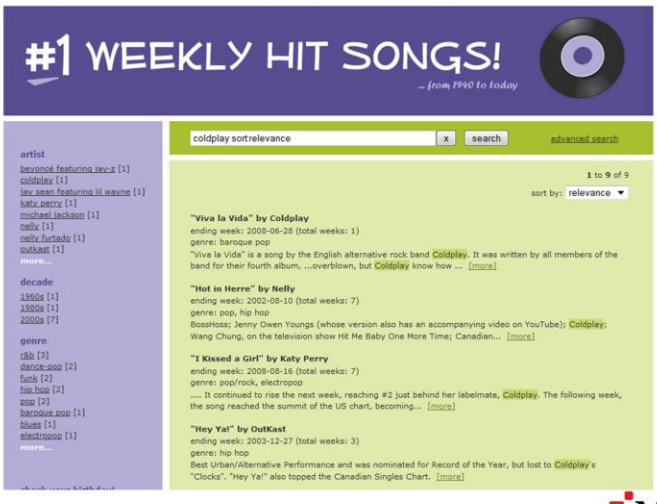
MarkLogic UNIVERSITY

Building Range Indexes



Indexing Concepts: Word Query

- Why does the Coldplay song appear first?



Indexing Concepts: Word Query

- Why does the Coldplay song appear first?
  - Word Query Defined on the Database

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]

Fields

- Query portions of a database based on elements
  - Useful if you know that you query on specific elements of the document.
  - Ex: 80% of document data is used only on display, and only 20% is queried
  - Create fields for the elements that are used in queries – the fields will create a smaller index without all the term lists of other data you are not interested in.
- Setup using the Administration Tool
- Configure → Databases → *Your DB* → Fields → Create

MarkLogic UNIVERSITY

Fields

Document #1

```
<top-song>
  <artist>
    The Beatles
  </artist>
</top-song>
```

Document #2

```
<top-song>
  <singer>
    Paul Simon
  </singer>
</top-song>
```

Document #3

```
<top-song>
  <group>
    Coldplay
  </group>
</top-song>
```

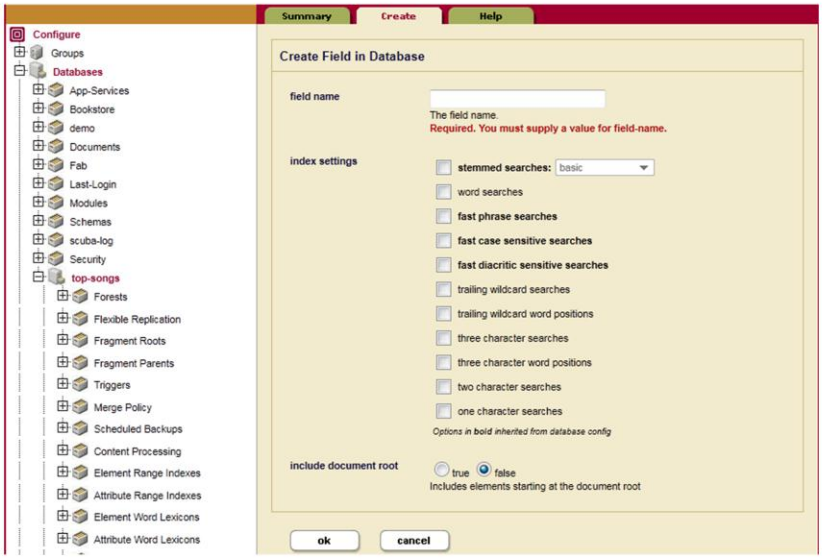
Document #4

```
<top-song>
  <band>
    Radiohead
  </band>
</top-song>
```

- Field Name: Performer
  - Include Elements:
    - <artist>|<singer>|<group>|<band>
  - Exclude Elements:
    - <writer>|<producer>
  - Specific Index Settings

MarkLogic UNIVERSITY

Fields



MarkLogic UNIVERSITY

Indexing Concepts: Tuning

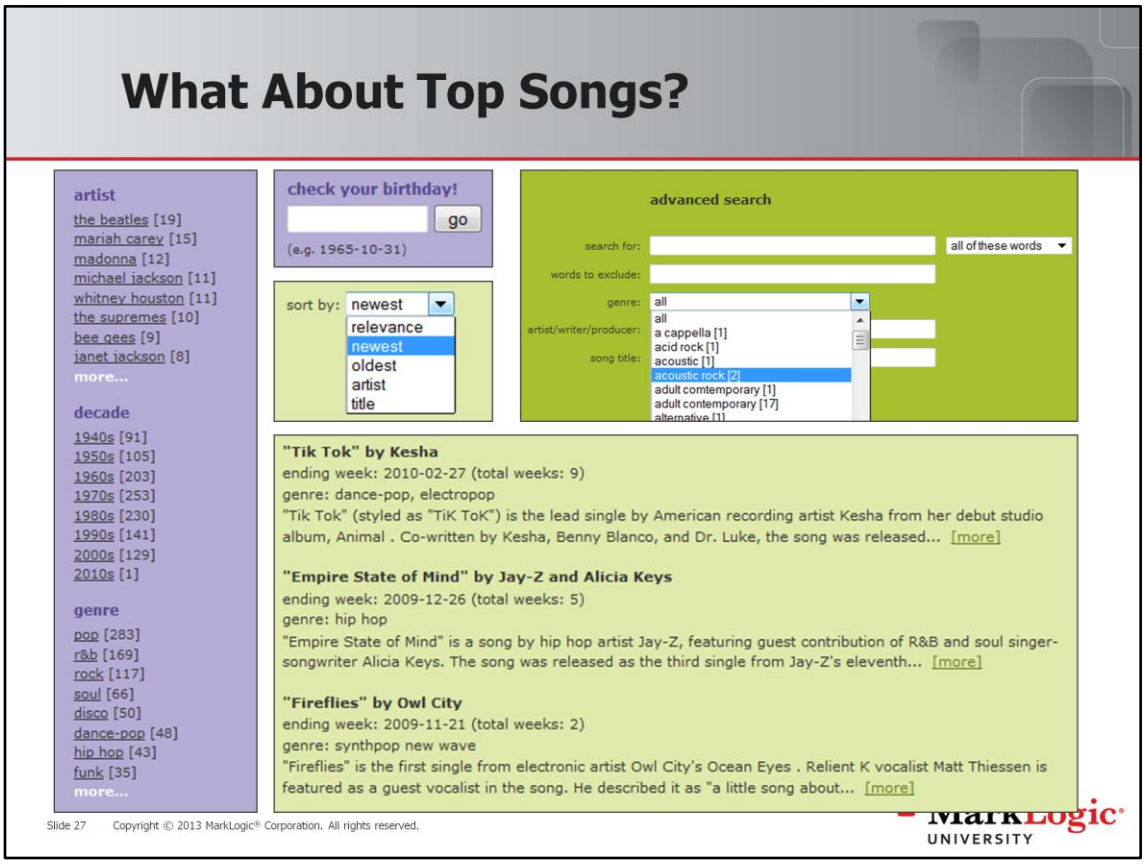
- `fn:count`
  - A 100% accurate count of your query results
  - Less efficient; requires filtering
- `xdmp:estimate`
  - May not be 100% accurate
  - Result based on indexes only
- Large gap between `fn:count` and `xdmp:estimate`?
  - Tune your query and/or indexes
  - Query Console Profile Function
  - `xdmp:query-meters()`
  - `xdmp:plan()`

MarkLogic UNIVERSITY

Indexing Concepts: Summary

- Approach to Query Resolution
  - Look at the query
  - Decide what indexes can help
  - Use indexes to narrow down the result set
    - More indexes = tighter result set
  - Filter the result set to confirm the match
- Tradeoffs
  - More indexes = more time during ingestion
  - More indexes = greater storage size on disk
  - Less indexes = more filtering = slower search
  - Range Indexes costs RAM

MarkLogic UNIVERSITY



Unit 9: Applying the Learning Objectives

- Describe the indexing concepts of filtering, Universal Index, Inverted Index, Term List, Stemming and Hashing
- Implement additional indexes to enhance query performance
  - Exercise 1
- Describe `xdmp:estimate` and `fn:count`
  - Exercise 2
- Create element and attribute range indexes
  - Exercise 3 | Exercise 4 | Exercise 5
- Create a lexicon and collation
- Create a Word Query

MarkLogic UNIVERSITY