Progress® MarkLogic®

Unit 4

Search

Basic Search

Advanced Word Search

Ranged Search

Relevance

Search API

TDE, SQL, and Optic API

Exercise 1: Basic Search

This exercise focuses on constructing basic queries and how to combine them. This uses Server-side JavaScript (SJS) but there is also a XQuery equivalent available. Feel free to adjust the presented workspace to have a better familiarity with the used functions.

1. In Query Console, http://localhost:8000/qconsole, delete the previous workspace and import the workspace located at:

   /home/cent/Desktop/fundamentals/solutions/unit_04/exercise_01/Basic Search - SJS.xml”

   - There may be some key differences with “Basic Search - XQY.xml”. This can be attributed to the fact that XQuery is more natural in dealing XML while being just as capable in handling JSON content.

2. The instructor will go through each tab.

Exercise 2: Advanced Word Search

This exercise focuses on some advanced concepts for word search constructing basic queries and how to combine them using Server-side JavaScript (SJS) or a XQuery. Feel free to adjust the presented workspace to have a better familiarity with the used functions.

1. In Query Console, http://localhost:8000/qconsole, delete the previous workspace and import the workspace located at:

   "/home/cent/Desktop/fundamentals/solutions/unit_04/exercise_02/Advanced Word Search - SJS.xml"

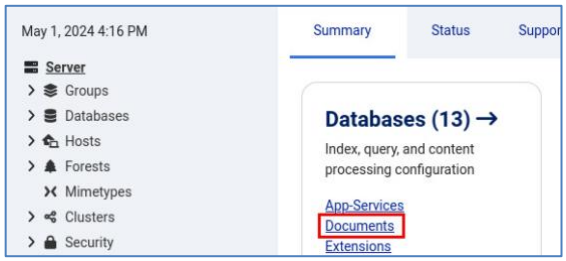
   - There may be some key differences with "Advanced Word Search - XQY.xml". This can be attributed to the fact that XQuery is more natural in dealing XML while being just as capable in handling JSON content.

2. The instructor will go through each tab.

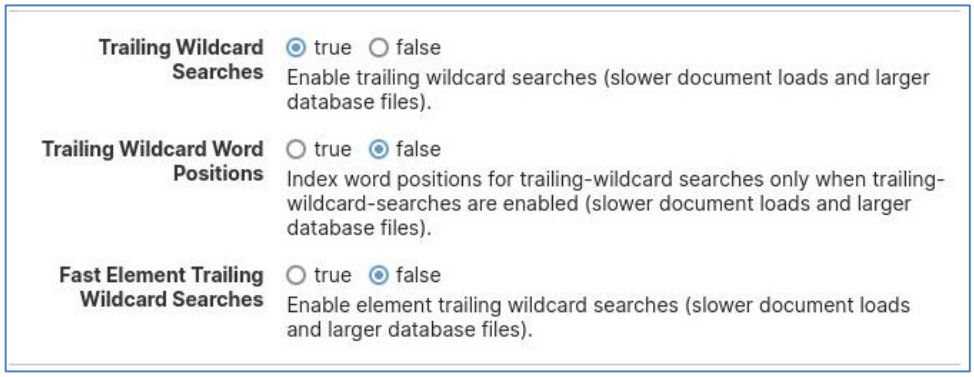
Enabling trailing wildcard search:

On the "Filtered vs Unfiltered" tab, notice the resulting count is different. This highlights how filtered searches are targeted towards accuracy. This does not mean that unfiltered search cannot be accurate.

3. In the Admin UI (http://localhost:8001), select the `Documents` database:

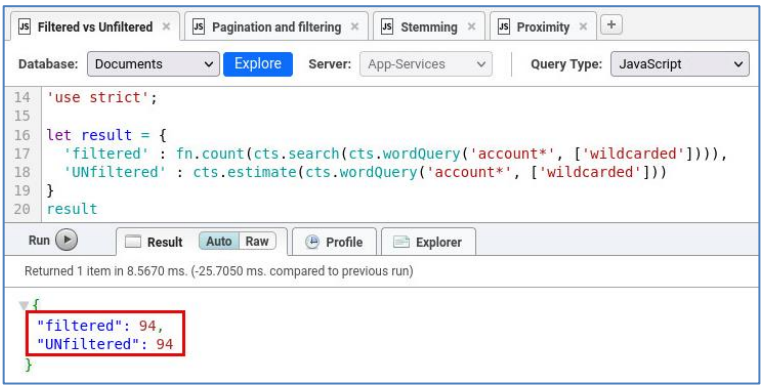


4. Scroll down to "Trailing Wildcard Searches", set it to "true" and click "ok":



   Note: MarkLogic will immediately execute a re-indexing of the documents.

5. Back in Query Console, re-run the “Filtered vs Unfiltered” tab and note that the “filtered” and “UNfiltered” counts are the same.



Enabling word positions

In Query Console, run the code on the “Proximity” tab and notice that the resulting counts are different. In the following steps you will enable Term Lists using ml-gradle instead of the Admin UI.

6. In VSC or a Terminal window, navigate to your project folder:

```
$ cd /home/cent/Desktop/star-wars
```

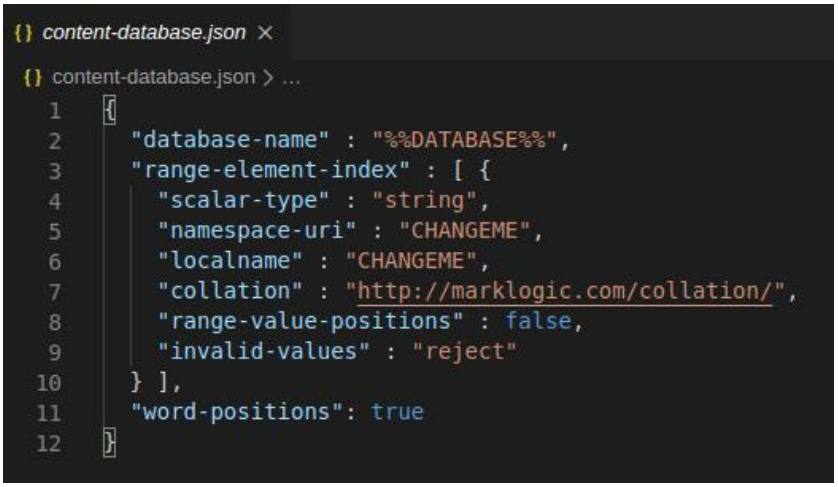
7. Copy the prepared query configuration.

```
$ cp -r /home/cent/Desktop/fundamentals/solutions/unit_04/exercise_02/star-wars/* .
```

- The folder structure:

```
star-wars
└───src
    └───main
        └───ml-config
            └───databases
                    content-database.json
```

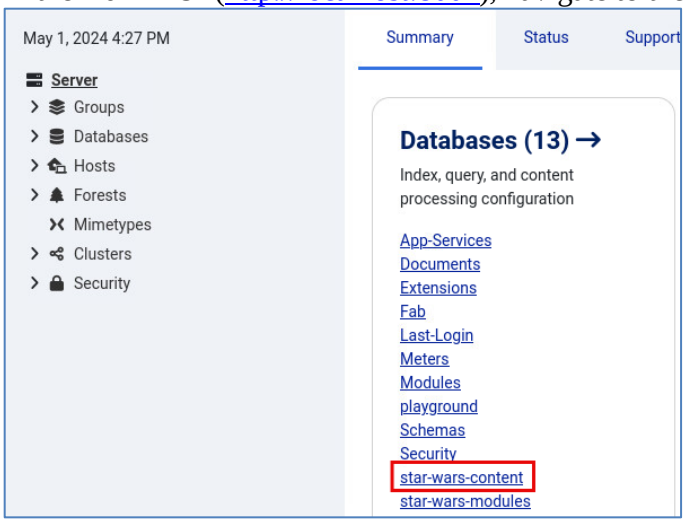
View the “content-database.json” and notice the new entry of **"word-positions": true** at the bottom.



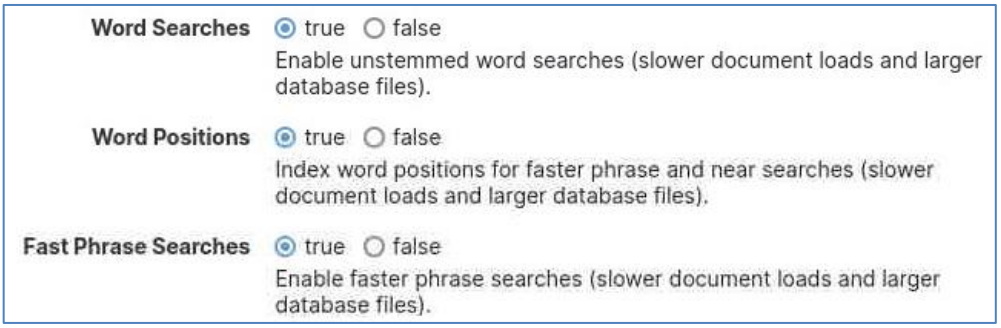
8. Deploy your configuration.

```
$ ./gradlew mlUpdateIndexes
```

In the Admin UI (http://localhost:8001), navigate to the `star-wars-content` database.



9. Scroll down to "Word Positions" and confirm that it is now set to "true".



For a complete list of the property names you could supply, use the Database Management API and access http://localhost:8002/manage/LATEST/databases/star-wars-content/properties?format=json

Challenge

To test your familiarity with index configuration, adjust your `top-songs` project to enable the following term lists:

- Element word positions
- Element value positions
- Attribute value positions
- Three character searches
- Trailing wild card searches

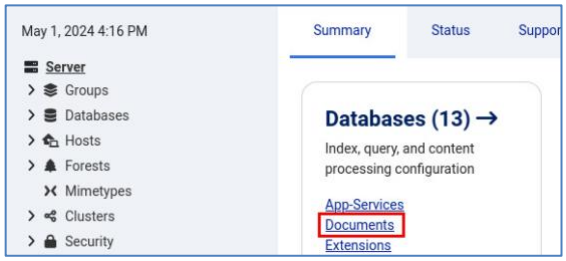
Exercise 3: Ranged Search

Before this exercise, we have been mostly focused on whether there are documents that contain a certain value or a particular word. This exercise is focused on answering questions like:

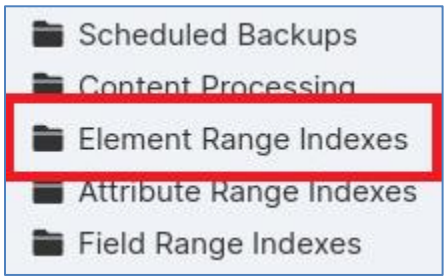
- Who was hired after the 2001?
- Who is taller than 1.5m?
- Who is located near the Times Square in New York?

We start by configuring some `Ranged Indexes`.

1. In the Admin UI (http://localhost:8001), navigate to `Documents` database:



2. Select "Element Range Indexes" on the left-hand side:



3. Click on the "Add" tab and enter the following:

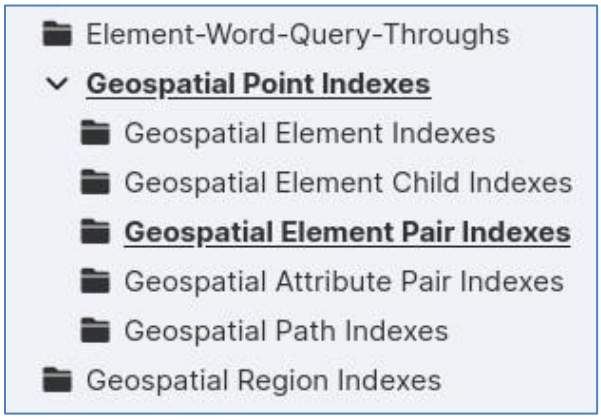
Field	Value
Scalar type	string
Local name	job_title,last_name,first_name

4. Click "Ok" button.
5. Click on "Add" tab to add another and enter the following:

Field	Value
Scalar type	int
Local name	office_number

6. Click "Ok" button.

7. Next, scroll down to select “Geospatial Point Indexes” and then “Geospatial Element Pair Indexes” as shown below:



8. Click on the “Add” tab and enter the following:

Field	Value
parent localname	instance
latitude localname	latitude
longitude localname	longitude

- This matches our updated structure of employee documents that applies the envelope pattern.
- This also highlights that the parent `localname` cannot be empty.

9. Click on “Ok”.

That section above is how one will configure indexes using the Admin UI. This is performed only for familiarity, but is generally not recommended for production. The following steps are considered best practice as it allows the configuration to be encoded and can be pushed to a code versioning repository.

10. In VSC or a Terminal window, navigate to the star-wars project folder:

```
$ cd /home/cent/Desktop/star-wars
```

11. Copy the prepared query configuration:

```
$ cp -r /home/cent/Desktop/fundamentals/solutions/unit_04/exercise_03/star-wars/* .
```

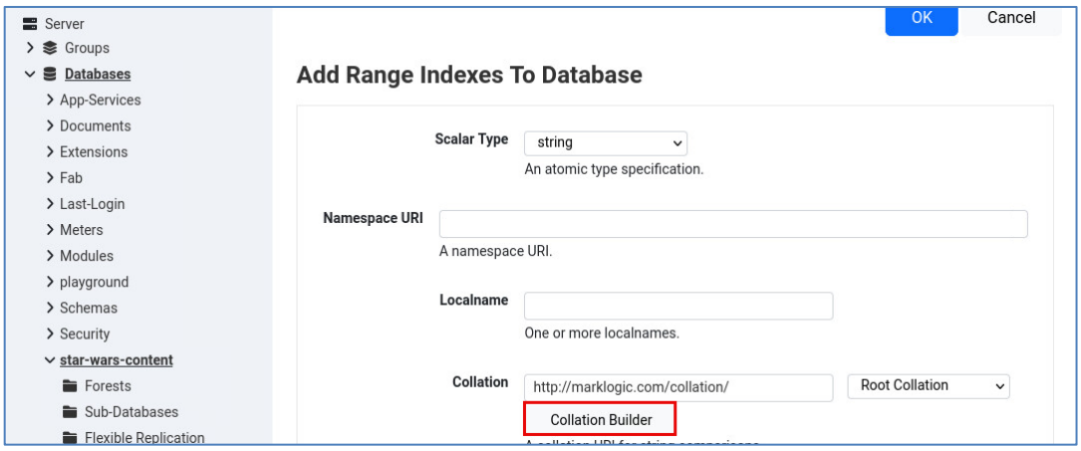
- The folder structure:

```
star-wars
└───src
    └───main
        └───ml-config
            └───databases
                    content-database.json
```

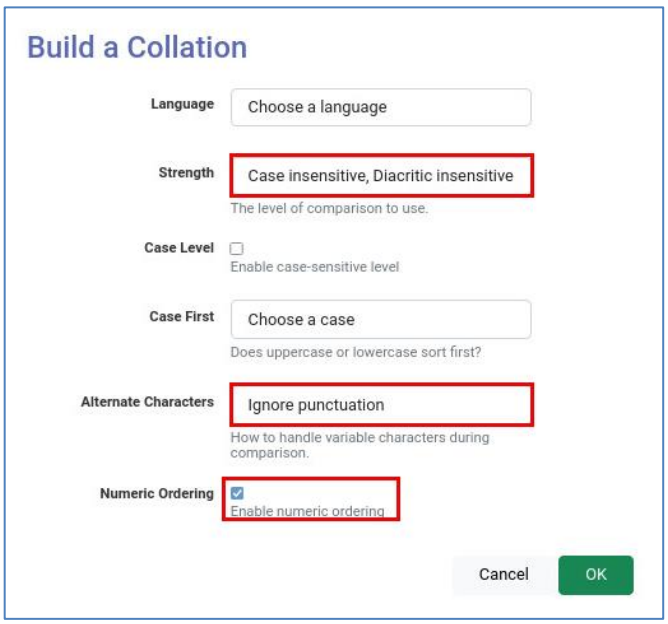
- Notice how it now includes several entries for "range-element-index"

12. Use the Admin UI's "collation builder" to generate that custom collation string on the `star-wars-content` database.

    - On the `star-wars-content` database, select >> Element Range Index >> Add tab, next select `string` for Scalar Type to expose the "Collation Builder" button:



    - In the "Build a Collation" set the properties as follows:



    - Click on "Ok".
        a. It's not required to fill every single input.

13. Return to VSC and deploy your configuration:

```
$ ./gradlew mlUpdateIndexes
```

14. In the Admin UI (http://localhost:8001), navigate to configuration page for star-wars-content database: Configure >> Databases >> star-wars-content >> Element Range Indexes:

   - Scroll down to view the Element Range Indexes that were enabled by the additions to the content-database.json file in a previous step.
   - You can check the database's status page to confirm that re-indexing has completed.

Now that our configuration changes have been applied. Let's return to Query Console to see how we make use of these newly configured range indexes.

15. In Query Console, http://localhost:8000/qconsole, delete the previous workspace and import the workspace located at:

    "/home/cent/Desktop/fundamentals/solutions/unit_04/exercise_03/Ranged Search - SJS.xml"

    - There may be some key differences with "Ranged Search - XQY.xml". This can be attributed to the fact that XQuery is more natural in dealing XML while being just as capable in handling JSON content.
16. The instructor will go through each tab.

Challenge

To test your familiarity with range index configuration, adjust your `top-songs` project to enable the following range indexes:

- genre
- week
- released attribute (See challenge on unit 3, exercise 4)
- length attribute (See challenge on unit 3, exercise 4)

Exercise 4: Relevance

This exercise focuses on how “Score” is used to sort your search results by default.

1. In Query Console, http://localhost:8000/qconsole, delete the previous workspace and import the workspace located at:

   “/home/cent/Desktop/fundamentals/solutions/unit_04/exercise_04/Relevance - SJS.xml”

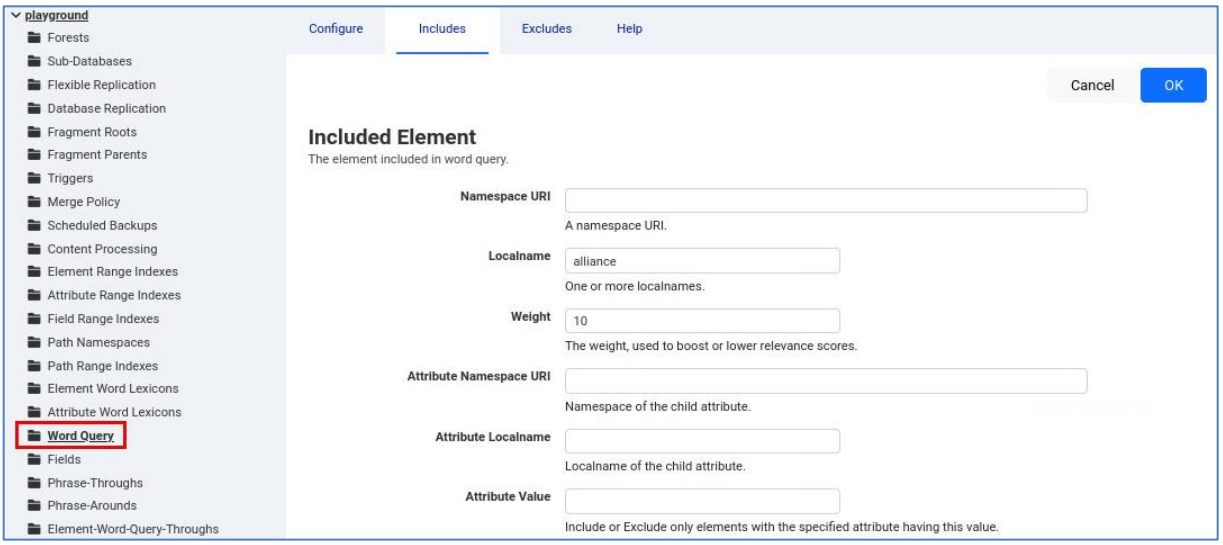
   - There may be some key differences with “Relevance - XQY.xml”. This can be attributed to the fact that XQuery is more natural in dealing XML while being just as capable in handling JSON content.

2. The instructor will go through each tab.

Enabling Word Query weights:

On the tab “Word query weights”, you will notice that the resulting sort is different from the tab description comments. This highlights how words of each property are treated the same by default. The following steps allows us to give priority to certain properties or elements.

3. In the Admin UI (http://localhost:8001), navigate to configuration page for the Documents database: Server >> Databases >> playground >> Word Query



   - Click on the “Includes” tab, add the following values, and click OK:

Field	Value
localname	alliance
weight	10

   - Go back to Query Console and continue with the lab exercises.

Challenge

To test your familiarity with index configuration, adjust your `top-songs` project to configure word query weights such that:

Local name	Weight
title	4
artist	4
album	3
descr	2

And on the “Excludes” tab, add `lengths` and `weeks`.

Compare the results of your search after configuration change.

Exercise 5: Search API

Building a google-like service/app can be expedited by using the pre-built Search API. Some of these APIs can be used as part of your MarkLogic application or can be used by your client applications using the REST API.

Start by loading prebuilt configurations that we will be using later.

1. In VSC or a Terminal window, navigate to the star-wars project folder:

```
$ cd /home/cent/Desktop/star-wars
```

2. Copy the prepared query configuration:

```
$ cp -r /home/cent/Desktop/fundamentals/solutions/unit_04/exercise_05/star-wars/* .
```

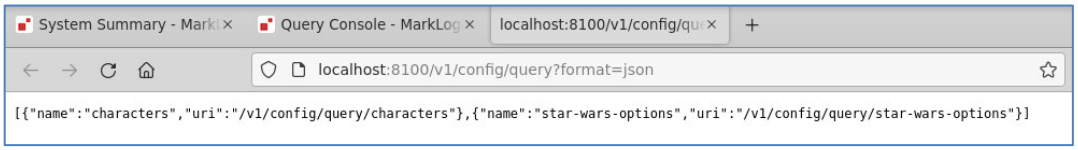
   - The folder structure:

```
star-wars
└───src
    └───main
        └───ml-modules
            └───options
                    characters.json
```

3. Deploy your configuration:

```
$ ./gradlew mlLoadModules
```

4. Using your browser, load this link (http://localhost:8100/v1/config/query?format=json) to confirm successful deployment. If prompted, enter *admin/admin* for a username/password.



This query option is specific to the Client REST API's search endpoint and the Search API's option input. Check out the documentation for more information about how you could further customize behavior.

5. In Query Console, http://localhost:8000/qconsole, delete the previous workspace and import the workspace located at:

   "/home/cent/Desktop/fundamentals/solutions/unit_04/exercise_05/Search API - SJS.xml"

   - There may be some key differences with "Search API - XQY.xml".
   - Note that JSearch is JavaScript specific, while the Search API is recommended for XQuery.

6. The instructor will go through each tab.

Challenge

To test your familiarity with Client REST API Search, adjust your `top-songs` project to enable the following facet search on the previously configured range indexes:

- genre
- week
- released attribute
- length attribute

Exercise 6: TDE, SQL, and Optic API

MarkLogic supports the creation of SQL views using Template Driven Extraction (TDE). These views can be exposed as typical relational tables using ODBC App Servers. They can also be accessed using the Client REST API or even as part of custom endpoints/services using the Optic API.

Start by loading prebuilt configurations that will be used later.

1. In VSC or a Terminal window, navigate to the star-wars project folder:

```
$ cd /home/cent/Desktop/star-wars
```

2. Copy the prepared query configuration:

```
$ cp -r /home/cent/Desktop/fundamentals/solutions/unit_04/exercise_06/star-wars/* .
```

   - The folder structure:

```
star-wars
│   gradle.properties
│
└───src
    └───main
        ├───ml-config
        │   ├───databases
        │   │       content-database.json
        │   │       schema-database.json
        │   │
        │   └───servers
        │           odbc.json
        │
        └───ml-schemas
            └───tde
                    characters.json
```

   - `odbc.json` configures an app server that your BI Tools could interact with.
   - `content-database.json` now makes use of the newly added `schema-database.json` to store your TDE templates. It is considered best practice for each content database to have its own schema database.

3. Deploy your configuration:

```
$ ./gradlew mlDeploy
```

4. Return to Query Console, http://localhost:8000/qconsole, delete the previous workspace and import the workspace located at:

   "/home/cent/Desktop/fundamentals/solutions/unit_04/exercise_06/TDE-SQL-Optic - SJS.xml"

   - There may be some key differences with "TDE-SQL-Optic - XQY.xml". Note that JSearch is JavaScript specific, while the Search API is recommended for XQuery.

5. The instructor will go through each tab.

Challenge

To test your familiarity with TDE configuration, adjust your `top-songs` project to create a view for top songs that will include the following columns:

- Title
- Artist
- Genre (as one concatenated entry in case of multiple values)
- Week
- Released (in date format)
- Length (in duration format)
- Album
- Label
- Writer
- Producer