JIRAE, Vol. 4, No. 1, April 2019, 21-25
e-ISSN 2407-7259

DOI: 10.9744/jirae.4.1.21-25

Magento 2 and Elasticsearch Integration

Andy Sukanto Kan[1,a], Peter Boots[2,b], Bart Delvaux[3,c]
[1] Department of Informatics, Petra Christian University, Surabaya, Indonesia & Information and Communication Technology, Fontys University of Applied Sciences, Eindhoven, The Netherlands
[2] Fontys University of Applied Sciences, Eindhoven, The Netherlands
[3] ISAAC, Eindhoven, The Netherlands
[a]andysukanto.ask@gmail.com, [b]p.boots@fontys.nl, [c]bart.delvaux@isaac.nl

Abstract. The purpose of this research is to extend the default integration of Elasticsearch in Magento 2.3 to search for other content which is category and content pages. The current search engine in Magento uses the default MySQL search engine, with the new update of Magento 2.3, Elasticsearch is introduced as an optional search engine. Elasticsearch provides a better and optimized full text search, however this new integration can only search the catalog products, therefore a way of extending the integration of Elasticsearch in Magento is searched for. The features that wanted to be implemented is Category search using Elasticsearch in Magento 2.

Keywords: Elasticsearch, Magento, Full text search, PHP, Indexing.

1. Introduction

An optimal search experience is very important for an E-commerce website to provide their consumer with. Good search experience leads to more conversion, which means more profit. One way of doing that is providing full-text search, which Elasticsearch is able to provide. Compared to other search methods such as Boolean search and standard database search, using Boolean search requires a lot of filtering that the user needs to input, this can turn out to be a hassle for some users, while standard database search doesn't have advanced features such as stemming that can improve search result by finding the root form of a word, weighting which is used for keyword prioritize keyword, synonyms which is useful for query expansion, stop words that can improve search result by filtering common words out and much more. With full-text search, a website can now search into documents just like a human. That's why ISAAC is trying to integrate Elasticsearch into their E-commerce solutions. ISAAC usually uses Magento, a PHP based E-commerce platform. The current search engine used in Magento is the default MySQL search engine, which is a standard database search method. With the update of Magento 2.3, Magento now comes with the integration of Elasticsearch as their search engine. The Elasticsearch integration with Magento 2 is specifically made for searching the product catalog of Magento. The main goal of this project is to extend the integration to be able to search other items which is Category of Magento 2 and do index operation such as search and reindexing.

2. Research and Implementation

2.1 Elasticsearch

This project will focus on using Elasticsearch to search and store categories and content pages of Magento 2. Content pages refers to pages that belong to Magento's CMS.

Elasticsearch is a group of one or more Elastic nodes which contains shards. Figure 1 shows the cluster configuration in this project. As seen on figure 1, there are 2 nodes in the cluster where node 1 is the master node. Inside each node there are indices and shards, the numbers indicate the amount of shard in a node, while the colors indicate their roles. The dark blue colored squares represent the primary shards, while light blue colored squares represent the replica shard. Category and CMS indices here represent the indices for the feature being implemented in this research, which is Category and CMS pages search.

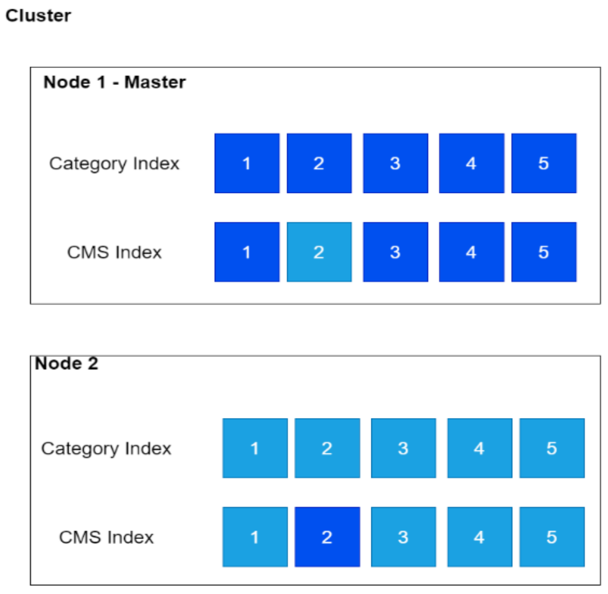


Figure 1. Cluster configuration

As seen in figure 1, this project will use two nodes which is the Master and Data nodes in the same cluster, this way a replica can be put in the data nodes. This ensures high availability in case a shard/node fails. Two nodes were used in this project since it's the minimum number of nodes needed to allocate replica shards; the minimum amount is chosen as the configuration because the number of sample data used in this project is relatively small. The number of nodes varies depending on the size of data; website with larger data to store will require a higher number of nodes for better scalability.

Before storing documents in Elasticsearch indices, index needs to be first configured. Index configurations include Index name, number of shards, mapping and analyzer. These are the configurations that needs to be configured during index creation. Other classes used to simplify the task are:

- IndexNameResolver to handle the index name with during creation of a new index to prevent downtime, this will be further discussed in point 2.5.
- Field Mapper to retrieve the attributes of the categories/content pages from Magento's database and map it according to their data types, this will be further discussed in point 2.3.
- Builder class is used to configure the settings of the analyzer in Elasticsearch, the settings of the analyzer will be further discussed in point 2.4.

2.2 Indexing

Magento 2 has 2 indexing types:

- Full Reindex rebuilds all the indexing-related database table. This can be done any time using the command line.
- The logic workflow for partial indexing is displayed in the diagram below:

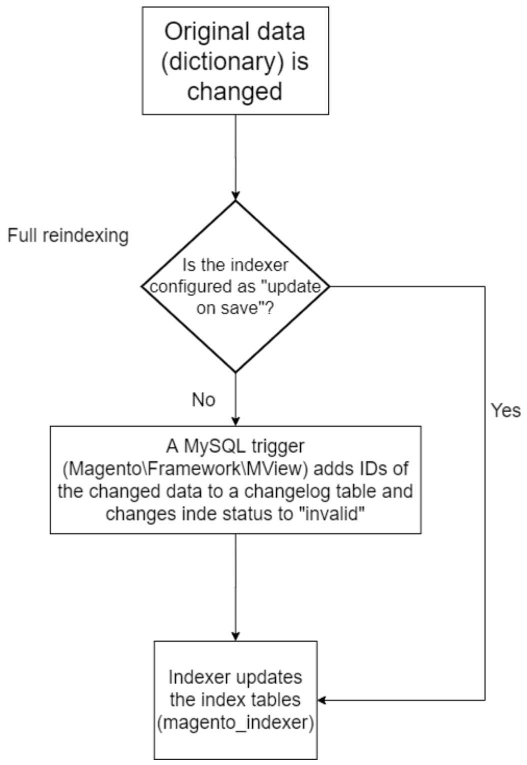

Original data (dictionary) is changed
Full reindexing
Is the indexer configured as "update on save"?
No
Yes
A MySQL trigger (Magento\Framework\MView) adds IDs of the changed data to a changelog table and changes inde status to "invalid"
Indexer updates the index tables (magento_indexer)

Figure 2. Partial reindexing workflow

Partial Reindex rebuilds the database only when there is a change on the database, the change is monitored by an observer. There are 2 indexer modes that follows this principle, Update on Save and Update by Schedule.

- Update by Schedule will create a changelog table to record every change made to the described table, and will run the indexing job according to cron schedule
- Update by save will do a reindexing every time an item is changed in the subscribed table.

Magento allows the developer to create their own custom indexer, this is achievable by implementing interfaces provided by Magento. The default Magento indexer will create a special table in the MySQL database to store the index information. While in this project, the index table will be stored in the Elasticsearch index because Elasticsearch will be used as the search engine instead of MySQL.

There are four functions that needs to be implemented in the custom indexer. These functions will be called according to the Indexing types and the enabled mode.

- Full reindexing:
  - Execute Full: This function runs when the full reindexing command is called from the command line.
- Partial reindexing:
  - ExecuteRow: This function runs when there is a CUD (Create-Update-Delete) operation on a single item of the observed items, and when the Update by Save is enabled.
  - ExecuteList: This function runs when there is a CUD operation on a list of items of the observed items, and when the Update by save is enabled.
  - Execute: This function will run according to the cron schedule, and when the update by schedule is enabled.

To be able to do partial reindexing an observer class is created, this is done by implementing Magento's observer interface. The function of the observer is to observe specific event, when the event is triggered, it will detect the event and runs. This can be done by specifying which event that wants to be tracked. In this project, an observer is created to observe the event that tracks the changes of category and content pages to do partial reindexing. Partial reindexing will only run when these two events are triggered.

In this project, the ExecuteFull method will create a whole new mapping of the index, and then load the documents. Therefore, the best practice to run full reindexing is when some changes need to be made to the mapping, for example a new stemmer, different language. The other methods will all run according to the indexer mode and will not update the index mapping. It will only delete the item changed and will load new list of items to the index.

2.3 Magento 2 and Elasticsearch Indexing

The Indexing workflows for the custom module is described in the following flowchart (Figure 3).

The workflow for indexing with the Elasticsearch module can be seen in figure 3. Before storing information in Elasticsearch indices, a mapping of variables and its field type needs to be first created, but each web shop can have different attributes adjusted to their categories, that's why the creation of mapping for the index needs to be dynamic and not only specially adjusted to one store.

Explicit mapping is implemented, field name and types are predefined in the mapping in this approach. Explicit mapping is done by creating a fieldMapper class. The purpose of this class is to retrieve the attributes of the Categories/content pages from Magento's database and map it according to their data types.

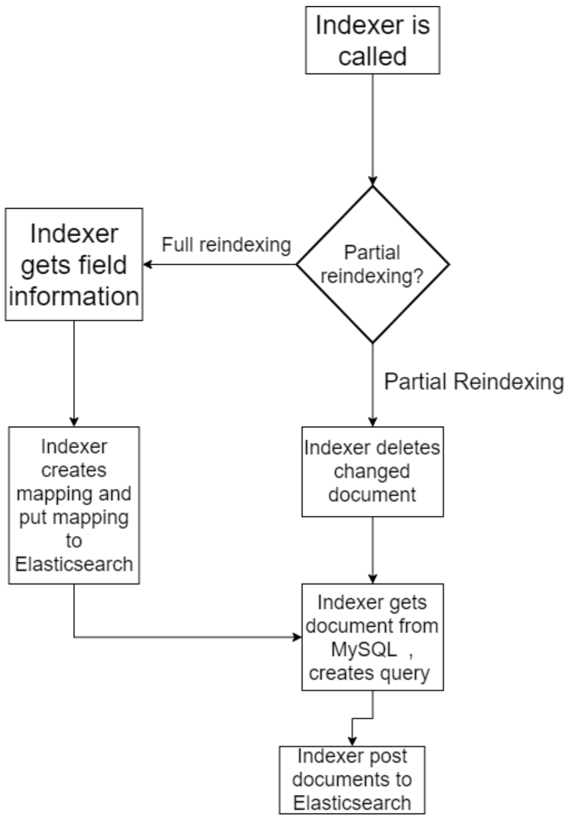

Indexer is called
Partial reindexing?
Full reindexing
Indexer gets field information
Partial Reindexing
Indexer creates mapping and put mapping to Elasticsearch
Indexer deletes changed document
Indexer gets document from MySQL , creates query
Indexer post documents to Elasticsearch

Figure 3. Indexing workflow for custom module

This way the field of the index mapping can be dynamic. For an example, a web shop that sell candies and a web shop that sell cars have different attributes for their categories. The fieldMapper class can map and determine the attributes along with their datatype without mentioning it explicitly, this makes the module compatible with all Magento2 web shops and unlikely to be obsolete.

The method to retrieve field information for categories and content pages are different, the structure of the retrieved informations is also different between the two, therefore a specific fieldMapper class is created for each category and content page.

Bulk API is chosen because it's better for index/delete operations involving a large number of documents. The Bulk API will insert the large number of documents to Elasticsearch index.

Partial reindexing runs when a CUD operation is performed at the categories/content pages table. Partial reindexing will not create a new mapping like full reindexing, instead it will capture which id the CUD operation is performed, then delete all the documents that match the specified ids. After that it will do the same insert operation for all the documents like the Full reindexing.

It is important to make sure the Elasticsearch's document id must be the same as the one on MySQL database. This way delete and update operation can be performed since the observer class will capture these ids instead of Elasticsearch's document id.

The default Magento search object has a condition where it checks if Elasticsearch is the preferred search engine, it will opt-in to use Elasticsearch when it is enabled. The default Elasticsearch module in Magento 2 works by creating an adapter to the official Elasticsearch PHP client. The adapter object is extended instead of the default Magento search object. There are some difficulties extending the default integration, because of private variables and method in the default integration, this turns out to be a problem since some function did not work, especially the main function that connects to Elasticsearch. Therefore, a custom adapter is created for Elasticsearch with a more accessible connection method to Elasticsearch.

2.4 Analyzer

Analysis in Elasticsearch is the process of converting text, like the body of an email into tokens or terms which will be used for searching. The Analysis will be performed by an analyzer. For example, the string query "The new elegant sink" will be split into the tokens "new, elegant, sink" after it has been analyzed.

Elasticsearch has pre-defined Analyzers that the user can use directly, or user can configure their own custom Analyzer. In this project, Custom analyzer is used because some features that are not available on the predefined analyzer such as stop words and synonyms filter. Analyzer is configured during the creation of an index. It is possible for every index to have their own unique custom analyzer.

HTML Strip Character Filter is used to strip out html entities found in the content pages input. A tokenizer receives a stream of characters and break it into individual tokens (usually individual words), and outputs a stream of tokens. For example, the letter "The new elegant sink!" will be broken into the terms [new, elegant, sink!]. These configurations are important to break descriptions found in category and the content from content pages into tokens.

To filter stop words in Elasticsearch, a filter is added in the analyzer. The list of stop words is the default Elasticsearch stop words.

Elasticsearch provide a filter that let the user configures synonyms for the Analyzer. Synonym can be applied through a file by mentioning the file path or directly written in the analyzer settings. The file approach is used for the synonyms, using file for synonyms doesn't require creation of a new index like the other approach that can slow down the index. The system needs to restart the index to load new file for synonyms, this way a new mapping doesn't need to be created everytime a new pair of synonym is created. Synonyms in search engine is very useful for query expansion, for example when user search the word for "phone" it will not only include terms that contain the word phone but the result for "smartphone, mobile" can also appear. This will provide user with better search experience by providing the expected result even if the search term is different.

To provide the user option to define their own list of synonyms, the default Magento synonym group model is extended. Administrator of the web shop can manage their synonym through the default synonyms configuration on Magento's admin page. To monitor changes in the synonym, an observer that monitor CUD operation on the synonym is created. It will create a new synonym file every time a CUD operation is performed. The file path for synonym file needs only to be mentioned once.

2.5 Zero downtime reindexing

It is not possible to modify existing fields in an Elasticsearch index. The only way of changing a mapping once an index has been created is to reindex it.

Reindexing creates a downtime of the index during the process, to handle this problem Elasticsearch has a feature called *index aliases*. An index alias is like a shortcut or symbolic link which can point to one or multiple indices. This feature allows a transparent switch between one index with another index on the running cluster. The workflow for index mapping with aliases can be found in the following figure:

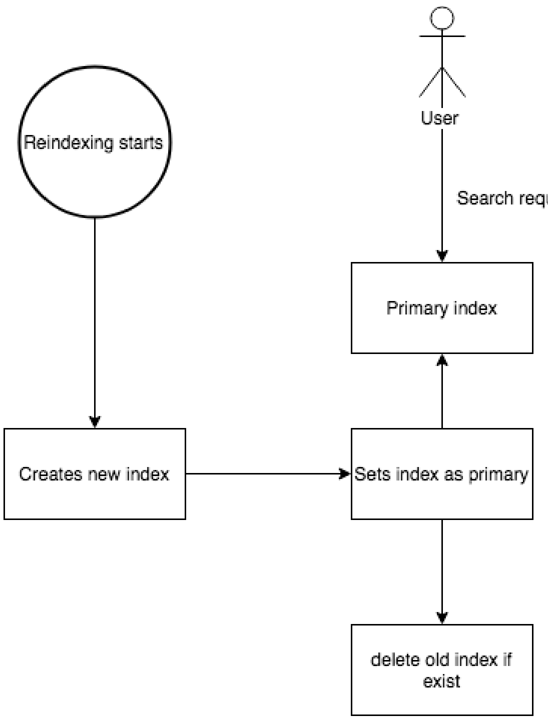


Figure 4. Zero downtime alias reindexing workflow

With this feature, a new index can be created without directly overwriting the old index. While the indexer is creating the new index, the old index can still be used for processing search request. Contact with the Index is done through the alias instead of the index name itself, after the new index is built, the alias will be set to the new index, and all the search request will now use the new index instead of the old index, the old index will then be deleted.

Indices and aliases naming are important because it needs to be unique, the indices and aliases name is updated by appending a version to the name of the index.

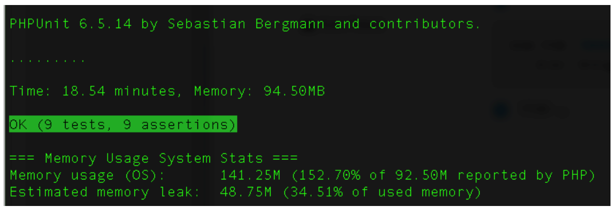
For example, the alias for the category index is *isaac_category*, therefore the index name should append the version name for each iteration. The first index should be named *isaac_category_v1*. Meanwhile the one that comes after should be *isaac_category_v2* and so on.

2.6 Integration Testing

During this project, an integration test is done to see how well the three modules in this project work together, following features are tested:

- Creation of an index in the category and content pages index
- Insertion of documents in the category and content pages index
- Deletion of documents in the category and content pages index
- Partial reindexing in the category and content pages index
- Synonyms for search in the category index.

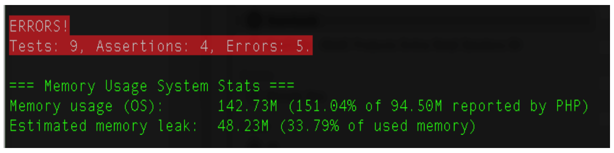
There are 9 tests in total, with 5 tests for the category module and 4 for the CMS module, all the test runs successfully in the Magento 2.3 with Elasticsearch 5.6 environment, the test results can be seen in figure 5 below.



```
PHPUnit 6.5.14 by Sebastian Bergmann and contributors.

.........

Time: 18.54 minutes, Memory: 94.50MB

OK (9 tests, 9 assertions)

=== Memory Usage System Stats ===
Memory usage (OS):        141.25M (152.70% of 92.50M reported by PHP)
Estimated memory leak:  48.75M (34.51% of used memory)
```

Figure 5. Integration Test Report on Magento 2.3 with Elasticsearch 5.6

An integration test with the updated environment, which is Magento 2.3.1 with Elasticsearch 6 is also done. There are some inconsistent failures during the testing process, sometimes every test succeeded except the synonyms test, the synonym insertion always fails with various errors, while some time the other test fails with an error that says, "No alive nodes found in your cluster". This error might be caused by the update of Elasticsearch 6, with some features deprecated and no longer supported. The results can be seen in the figure below.



```
ERRORS!
Tests: 9, Assertions: 4, Errors: 5.

=== Memory Usage System Stats ===
Memory usage (OS):        142.73M (151.04% of 94.50M reported by PHP)
Estimated memory leak:  48.23M (33.79% of used memory)
```

Figure 6. Test report on Magento 2.3.1 with Elasticsearch 6.7

3. Conclusion and Recommendation

By creating and combining several modules (base, category, CMS), custom indexer workflow, custom Elasticsearch cluster configuration, custom Elasticsearch filters. The new integration can add extra features that is previously not possible with the default integration. Features such as category and content pages search has been implemented and runs without any problem. Synonyms to improve the search experience for user has been implemented.

During the research, a newer version of Magento that supports Elasticsearch 6 has been released. After testing the module on the newer environment, the main features which are category and content pages search works perfectly. However, the synonyms is not working because of deprecated features on Elasticsearch 6. Some stability issues also occur during the testing because of cluster failures. Further research can explore on how to fix this stability issues, and fixing tokenizer issues for the synonyms.

References

1. Elastic. (n.d). Elastic Search Reference 5.6, Retrieved from https://www.elastic.co/guide/en/Elasticsearch/reference/5.6/index.html
2. Elastic. (n.d) Elastic Search Reference 6.7, Retrieved from https://www.elastic.co/guide/en/Elasticsearch/reference/6.7/index.html
3. Magento. (n.d) PHP Developer Guide, Retrieved from https://devdocs.Magento.com/guides/v2.3/extension-dev-guide/bk-extension-dev-guide.html
4. Magento. (2019, Feb 01). Index Trigger Events, Retrieved from https://docs.Magento.com/m2/ce/user_guide/system/index-management-events.html
5. Magento. (n.d). Indexing overview, Retrieved from https://devdocs.magento.com/guides/v2.3/extension-dev-guide/indexing.html
6. Pashkevich, V (2018, November 8) Comprehensive Guide to Magento 2 Indexing, Retrieved from https://amasty.com/blog/comprehensive-guide-to-magento-2-indexing/
7. Rehkopf, M (n.d) What is a Kanban board? Retrieved from https://www.atlassian.com/agile/kanban/boards
8. Santagostino, J (2018, January 20) Newline delimited JSON is awesome, Retrieved from https://medium.com/@kandros/newline-delimited-json-is-awesome-8f6259ed4b4b
9. Saylor, C. (2016, March 02). Elasticsearch Series: Rebuilding Indices with No Downtime. Retrieved from https://medium.com/zumba-tech/elasticsearch-series-rebuilding-indices-with-no-downtime-3498bebbebc
10. Teufel, S (2014) Lecture 1: Introduction and the Boolean Model [PowerPoint Slides], Retrieved from https://www.cl.cam.ac.uk/teaching/1415/InfoRtrv/lecture1.pdf