Retrieving Non-Relational Data:

INTRODUCTION:

For Information to be conveniently retrieved through query processing, it is necessary that the data is cleansed of unnecessary material. In most use cases, the data is not cleansed of this unnecessary/extraneous data and not conveniently tagged for retrieval. To extract information from this content you will need to rely on some levels of text mining, text extraction, or possibly full-up natural language processing (NLP) techniques. The typical ways to extract data are:

- Extracting Entities: Finding entities such as 'author, title, references etc.' in the input and using them to match with the existing data.
- Clustering Content: Sorting data into specific categories such as 'science, technology, analytics etc.' and retrieve this data from the information understood through input.
- Relationship Extraction: To use graph based databases to cluster similar data and use it for retrieval.

To process the input language, the machine needs to convert the given data into meaningful lexical items such as words, phrases and syntactic markers. This can be done in several methods:
Structure Extraction, Lemmatization, Tokenization, Decompounding, Entity Extraction etc. While these methods are not used to retrieve data, they help the machine in understanding what the user wants.

MySQL supports text searching by using LIKE operator and REGULAR expressions. The REGULAR expression is a special string that describes a search pattern. It is a powerful that can be used to identify strings of text based on patterns. However, when the text data is too large, using these methods has some limitations such as performance, relevance factoring and inability to search flexibly. Because of such limitations, MySQL has introduced full-text searches. In Full-text searching, MySQL creates an index of words of the document and searches on these indices. The size of these indices are moderate, therefore, it does not take much space to store the indices. The Full-text searches can only be done in MySQL versions having 'MyISAM' and 'InnoDB' storage engines.

In MySQL, natural language full-text searches are used to retrieve text data using a part of the data. MySQL looks for rows or documents that are relevant to the free-text natural human language query, for example, "Recipe of Grilled Sandwich". MySQL uses relevance as factor in determining text data similar to input. Relevance is a function that gives a floating-point number. When the relevance returns '0', it means that there is no similarity between the given input and the text data. MySQL computes relevance based on various factors such as number of words in the document, number of unique words, total number of words in the collection of files, and number of documents containing a unique word. To use relevance in MySQL, we utilize the functions MATCH() and AGAINST(). The MATCH() function specifies the column where you want to search while AGAINST() function determines the search expression to be used. The AGAINST() function uses 'IN NATURAL LANGUAGE MODE' by default without explicit mentioning. We can also instruct MySQL to be case-sensitive while searching for text data. While MySQL does support full-text searching, it has limitations.

- The minimum length of search term defined in MySQL engine is 4. Any input less than 4 words will not return a result in MySQL.
- Stop words are ignored by MySQL. A list of these words can be found in the MySQL source code path "storage/myisam/ft_static.c".

Before performing a Full-text search on the database, you must index its data. MySQL recreates a Full-text indices on the data whenever it changes. This type of indexing can be performed by using 'FULLTEXT (column1, column2, ...)' syntax. Indexing can be done on columns of data types CHAR, VARCHAR or TEXT in 'MyISAM' or 'InnoDB' table types. To help users search text data with too-short words, MySQL uses query expansion. Query Expansion widens the search result of Full-text searches based on automatic relevance feedback. To use query expansion, we use the search modifier "WITH QUERY EXPANSION". The MySQL engine uses a set of steps when query expansion is used:

- MySQL search engine looks for all rows that match the search query.
- It checks all rows in the search result and finds all relevant words.
- It performs a search again based on the relevant words.

You can use query expansion when the search results are to small. Query Expansion offers the user to get more information that is relevant from the original information provided by the users.

Example:

```
SELECT column1, column2
FROM table_name
WHERE MATCH (column1, column2)
        AGAINST ('keyword', WITH QUERY EXPANSION);
```

Apart from Natural Language Full-text searching, MySQL supports Boolean Full-text searches. This allows you to perform full-text search based on complex queries in Boolean mode along with Boolean operators. To perform full-text search in Boolean mode, we use the "IN BOOLEAN MODE" modifier in the AGAINST () expression. MySQL full-text Boolean search is recommended only for experienced users, since it is complex to operate.

CONCLUSION:

Information retrieval based on input requires the machine to process the given data as well as the data present in the database. This can be done by various tools and methods provided by Natural Language Processing. MySQL provides LIKE operator and REGULAR expressions to find data similar to the given input. Full-text search can be applied onto MySQL databases that run on 'MyISAM' and 'InnoDB' search engines. Using MATCH () and AGAINST() functions. Query Expansion can also be used to retrieve data based on relevance factorization. Hence, Data can be retrieved based on input in MySQL using these methods.

REFERENCES:

1. Fellbaum, Christiane (2005). WordNet and wordnets. In: Brown, Keith et al. (eds.), Encyclopedia of Language and Linguistics, Second Edition, Oxford: Elsevier, 665-670

2. George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.
3. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.
4. Google/CloudSQL
5. Google/Cloud Bigtable