**Non-Traditional Data Survey**

***By Zihan Li and Chunan Lin***

In our database, the non-traditional data we chose is the description of each film and the data type is TEXT. We have already import all the data into our database. Considering it as a non-traditional data, our plan is that when you do a query related to the description, you can enter sentences as part of the description and then you can search the movie along with the entire description.

In our example, we would like to handle non-traditional data like this:

When the user tries to get a description of what he knows might be the name of the movie or the actor's name or might be a part of the description. There will be a description search operation in the menu. If the user chooses the option, he will then have to input what he knows about the description like the name of the movie or the name of one of the actors or may be a part of the description or even some non-related words then hit enter all descriptions that have what the user entered will be shown, the one matches most of the words will be shown first, the one matches the least of the words will be shown last, those cannot match the words won't be shown.

To achieve this, we plan to use TEXT type to store all the descriptions.

We have learned something about handling BLOB and TEXT type of file. BLOB values are treated as binary strings (byte strings). They have the binary character set and collation, and comparison and sorting are based on the numeric values of the bytes in column values, and it has four types: TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. TEXT values are treated as non-binary strings (character strings). They have a character set other than binary, and values are sorted and compared based on the collation of the character set. It also has four types: TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT. In case it seems that we can consider BLOB as varbinary and consider TEXT as varchar, but there are differences between them. TEXT and BLOB cannot have default values, and for indexes on BLOB and TEXT columns, you must specify an index prefix length. Each BLOB or TEXT value is represented internally by a separately allocated object. This is in contrast to all other data types, for which storage is allocated once per column when the table is opened.[1]

To do so we need to do things like fuzzy query, we have already done a simpler one in our query that we can search, we used the "like" function in SQL.

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| Like operator | Description |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

[2]

With this, we can find the data with the word we entered.

But what we are going to do to handle these descriptions will be much more complicated. The user will not just type in a single word, but mostly like a sentence. So if we want to search as google does, we need to separate every single word in the sentence, and search all of them separately and put all results of one search in one column and do a calculation of how many times does a result shows then arrange them from the most frequent to the least, so the result users will see first is the one matches the most.

Further: The users who search the database can be from all over the world so they might use dialect words, specialist words, national spelling or even misspelling some words, so in general our fuzzy search needs to contain the following function:

Dialect differences (things like crayfish, crawfish)

National spelling differences (things like yoghourt, yogurt and yoghurt)

National word differences (like pants and trousers)

Multiple valid spellings (like encyclopaedia and encyclopedia)

Misspelling (like definitely often rendered as definitly, definately)

Mistyping (like computer as computer or computer.)

To realize these functions we will need a reference word list. The word list needs to contain all kinds of words with different spellings and some common misspellings. This word list will be referenced, by a foreign key constraint, from a large narrow table that records where the various entities that are associated with reference word. So that relevant matches, maybe a link, a phrase, a description or an image, can be returned to the user. [3]

Bibliography:

[1] referring : "MySQL Reference Manual"

[2] referring : "W3schools.com"

[3] referring: "Doing Fuzzy Searches in SQL Server"