RESEARCH OF WORD SEARCH ALGORITHMS BASED ON RELATIONAL DATABASE

Oleksij Volkov, Simona Ramanauskaitė
Šiauliai university, Faculty of Technology

Introduction

Word games are a simple and fun way to spend time. It helps to improve a person's vocabulary, expand erudition, train memory and intelligence, and develop logic and associative thinking. There exists different types of word games which require different game logic, and different word-finding algorithms. The optimization of word search algorithms is required to design a fully functioning and efficient system.

The aim of this work is to research the efficiency of different kinds of word search algorithms, using an SQL-based relational database. Results of this research could provide valuable information for the design of an efficient helping system for solving word games. String Search Algorithms.

To find a suitable word-for-word game requires review of dictionary, and to find one. This requires certain resources to compare all words from the dictionary to the word search pattern. There are different algorithms for string matching problem. As R. Rivest [1] states, to search for a pattern of length m in a text of length n (where $n > m$) the search time is $0(n)$ in the worst case (for fixed m). Moreover, in the worst case, at least $n - m + 1$ character must be inspected. However the complexity of string search can vary depending on the algorithm used.

The most known unoptimized string search algorithm is naive or brute force algorithm. It tries to match any substring of length m in the text with the pattern [2], therefore n-m+1different combination of one word must be checked to find all pattern matches. This algorithm matches all pattern letters for all word combinations. Meanwhile Knuth, Morris, and Pratt's algorithm [3] optimizes it and does not examine all the patterns. This algorithm comparing two strings compares it until the first unmatched letters; therefore, the complexity of this algorithm is a little bit smaller and the search time is more dependable on the search pattern and comparable string.

To reduce the complexity of the search algorithm Boyer and Moore [4] proposed to search the pattern from right to left. If no mismatch occurs, then the pattern has been found. Otherwise, the pattern matching is moved to the left by an amount of letters which is calculated using heuristic algorithm (adjusting to the difference of the matching pattern).

The complexity research of these string search algorithms depend on the length of the pattern was carried out by Ricardo A. Baeza-Yates [5] (see. Fig. 1). It showed the decrease of the Boyer and Moore algorithm, compared to the naive and Knuth, Morris, Pratt algorithm. However all these classic algorithms were proposed in 1977; meanwhile, new information technologies allow different solutions to the problem.

Word Search in SQL-Based Relational Databases

Relational database model is an organization of data into collections of two-dimensional tables called "relations" [6]. This type of database management systems at present is the most common in different kinds of information systems [7, 8]. The popularity of relational database management systems is increased by support of SQL, which allowed clear syntax to get required data from relational databases.

The Structured Query Language (SQL) is a standardized language used to retrieve and update data stored in relational tables (or databases) [9]. SQL has different type of queries and parameters. However, for string search algorithms the LIKE clause is usually used. It has two wildcards: the percent sign (%) represents zero, one, or multiple characters; the underscore (_) represents a single number or character [10].

The symbols can be used in combinations to compose a desired pattern of search string.

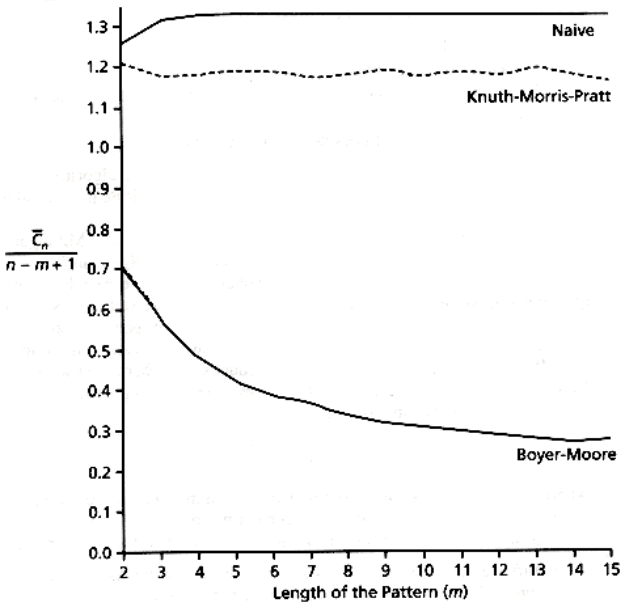

Fig. 1. Comparison of complexity of string search algorithms [5]

Possible Database Designs for Word Search Based on Letter Patterns

The most important element of word search systems is the design of a system database: the database has to store the word dictionary which usually stores a large number of data; a word search system usually searches for matching words according to a known pattern; therefore, many connections to the database can be used.

The simplest design of a word search database is represented in Fig. 2.

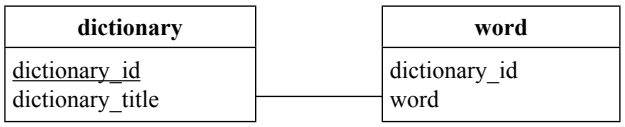

Fig. 2. Database design No. 1

This architecture is very simple and allows having different dictionaries in the same system. Using this kind of database design, two types of SQL queries can be used to get matching words from the database:

- To select all matching words from the database according to provided pattern PTRN: SELECT word.word FROM word WHERE word.word LIKE 'PTRN'.
- To select all matching words from a specific dictionary *DICT* in the database according to provided pattern *PTRN*: SELECT word.word FROM word WHERE word.dictionary_id = *DICT* AND word.word LIKE '*PTRN*'.

The pattern can be composed of any symbols from the alphabet and percent sign (%) as well as underscore (_) wildcards to represent any unimportant parts of the word. However this architecture does not allow for a search according to the length of the word. Therefore this database design can be improved by adding an additional field to the table *word* (see Fig. 3).

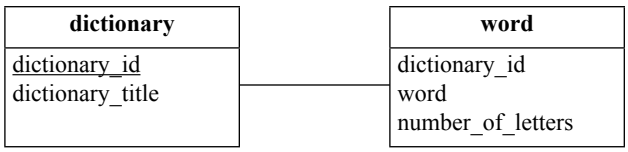

Fig. 3. Database design No. 2

This database design improvement allows for two additional queries to specify the search:

- To select all matching words from the specific dictionary *DICT* in the database according to a provided pattern *PTRN* where the length of the word is exactly *N*: SELECT word.word FROM word WHERE word.dictionary_id = *DICT* AND word.word LIKE '*PTRN*' AND word.number_of_letters = *N*.
- To select all matching words from the specific dictionary *DICT* in the database according to a provided pattern *PTRN* where the length of the word is more than *N1* and less than *N2*: SELECT word.word FROM word WHERE word.dictionary_id = *DICT* AND word.word LIKE '*PTRN*' AND (word.number_of_letters > *N1* AND word.number_of_letters < *N2*).

There can be different combinations of word length limitations; however, these two are the mostly used.

The SQL query to get all words with exact length can also be realized with first database design. For this purpose, thee underscore (_) wildcard should be used to identify all unknown letters and their position in the word. However, to specify the range of world letters would not be possible with the first database design.

To design a system with a wide variety of string search algorithms, a scrabble-type search has to be supported. This type of search usually has no pattern, but just a list of possible letters which have to be used to get a word. To do this with the previously described database design would be difficult: all possible combinations of known letters have to be generated; a large number of possible letter combinations can be generated for long set of letters; and SQL is not capable of generating all possible word combinations from a known set of letters, etc.

To solve this problem, a new improvement to the database design has to be made – the database should describe all words by splitting them into letters and associating them to the corresponding word (see. Fig 4).

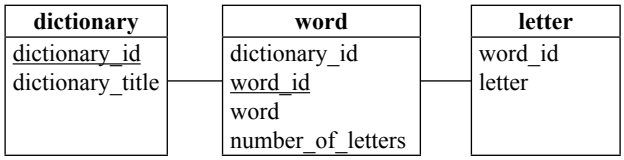

Fig. 4. Database design No. 3

This architecture would allow for finding all words for a scrabble-type word search:

- To select all matching words from a specific dictionary *DICT* in the database according to a provided set of letters {*L1*, *L2*, …, *LN*} where at least *M* letters of the set would be used: SELECT word.word FROM letter, word WHERE word.word_id = letter.word_id AND (letter.letter = '*L1*' OR letter.letter = '*L2*' OR … OR letter.letter = '*LN*') AND word.dictionary_id = *DICT* group by letter.word_id HAVING count(letter.word_id) > *M*.

This database design would provide all the functionality the system needs. However, it can be extended to search for words by pattern using the table letter, instead of the word. This requires adding the position of letter in the word only (see. Fig. 5).

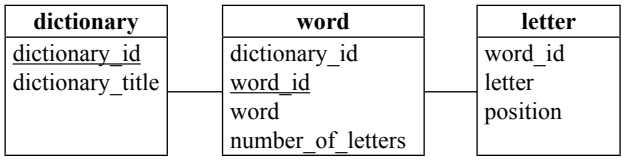

Fig. 5. Database design No. 4

The position of a letter in a word allows us to search by specifying which letter in the set and the letter should be placed in the word. Therefore, the query for searching a word of specific length can be changed to it:

- To select all matching words from a specific dictionary *DICT* in the database according to a provided set of letters {*L1*, *L2*, …, *LN*} the length of the word is exactly *M*: SELECT word.word FROM letter, word WHERE word.number_of_letters = *M* AND word.

word_id = letter.word_id AND ((letter.letter = '*L1*' and letter.position=1) OR (letter.letter = '*L2*' AND letter.position=2) OR … OR (letter.letter = '*LN*' AND letter.position=*N*)) AND word.dictionary_id = *DICT* group by letter.word_id HAVING count(letter.word_id) > *N*.

However, it is unknown if this query would be more efficient than in the previous architecture.

Research of Relational Database-Based Word Search Algorithms

To define the best database architecture from the proposed ones, a research of its efficiency has to be carried out. For this purpose four different database architectures (described earlier in this paper) will be created in the SQLite database management system. To examine the efficiency of these database architectures, experiments with different number of words in the database have to be done. Therefore, experiments will be executed with 10, 100, 1000 and 5000 words in the database. According to the database architecture, all provided SQL queries should be tested using different patterns or sets of letters.

The purpose of the research would be to evaluate the query execution time to obtain a suggested list of words according to the chosen database architecture and the number of words in the database and search pattern or set of letters.

Testing will be done with words from the English dictionary. With each design of the database two types of requests (complex and simple) will be performed; the average query time will be compared with other designs of databases.

Research of Database Design No. 1

To test the performance of the first database design, 4 SQL queries were used to get all the words which start with letter 'a' and differ in conditions:

1. SELECT word.word FROM word WHERE word.word LIKE 'a%'
2. SELECT word.word FROM word WHERE word.word LIKE 'a_%'
3. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a%'
4. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a_%'

Results, obtained from the test are presented in 1 table.

As the results of this test showed, the execution time is dependable on the number of words in the database. However, the type of SQL query does not have a great influence on speed (as only one dictionary exists in the database), but only the number of found elements differs because of the conditions which were used.

Research of Database Design No. 2

The second database design is tested as well as the first one; however, a limitation to the word size in used:

1. SELECT word.word FROM word WHERE word.word LIKE 'a%' AND word.number_of_letters = 7
2. SELECT word.word FROM word WHERE word.word LIKE 'a_%' AND (word.number_of_letters > 3 AND word.number_of_letters < 8)
3. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a%' AND word.number_of_letters = 7
4. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a_%' AND (word.number_of_letters > 3 AND word.number_of_letters < 8)

The result of the second database designs testing is provided in table 2 where can be noticed similar tendencies. However, comparing results of the first and second research, we can notice the decrees of the query execution time. It shows that word length storage can optimize the SQL query execution time for a word search when the desired length of the word is known.

Table 1. ***Results of testing with the first database design***

Site of the dictionary	Query execution time, ms				Number of returned rows by query			
	1'st query	2'nd query	3'rd query	4'th query	1'st query	2'nd query	3'rd query	4'th query
10 words	1	1	1	1	10	9	10	9
100 words	2	2	2	2	100	99	100	99
1000 words	6	6	6	6	408	407	408	407
5000 words	8	8	8	8	408	407	408	407

Table 2. ***Results of testing with the second database design***

Site of the dictionary	Query execution time, ms				Number of returned rows by query			
	1'st query	2'nd query	3'rd query	4'th query	1'st query	2'nd query	3'rd query	4'th query
10 words	1	1	1	1	3	5	3	5
100 words	1	1	1	1	11	38	11	38
1000 words	1	3	1	3	51	186	51	186
5000 words	5	6	5	6	51	186	51	186

Research of Database Design No. 3

The third design has more differences in comparison to the first one, as it has an additional table to store letters of the words. Therefore, more complex SQL queries to join multiple tables have to be used; however, similar queries to the first and second design should work either:

1. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a%' AND word.number_of_letters = 7
2. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a_%' AND (word.number_of_letters > 3 AND word.number_of_letters < 8)
3. SELECT word.word FROM letter, word WHERE word.word_id = letter.word_id AND (letter.letter = 'a' OR letter.letter = 'b') AND word.dictionary_id = 1 group by letter.word_id HAVING count(letter.word_id) > 1
4. SELECT word.word FROM letter, word WHERE word.word_id = letter.word_id AND (letter.letter = 'a' OR letter.letter = 'b' OR letter.letter = 'r') AND word.dictionary_id = 1 group by letter.word_id HAVING count(letter.word_id) > 3

As results of this test show, (see 3 table) the execution time of complex queries is bigger comparing to queries which do not use the table 'letter'. Another interesting thing which was noticed by comparing the 3rd and 4th queries is the execution time when there are 1000 words in the database – the 4th query is faster when there are up to 1000 words, while the 3rd query is faster when the number of words in the database is 5000. This leads to fact that the speed of these queries is dependent on the word in the database.

Research of Database Design No. 4

For the experiment with the fourth database design, different types of SQL queries were tested as well:

1. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a%' AND word.number_of_letters = 7
2. SELECT word.word FROM word WHERE word.dictionary_id = 1 AND word.word LIKE 'a_%' AND (word.number_of_letters > 3 AND word.number_of_letters < 8)
3. SELECT word.word FROM letter, word WHERE word.number_of_letters = 7 AND word.word_id = letter.word_id AND ((letter.letter = 'a' and letter.position=1) OR (letter.letter = 'b' AND letter.position=2)) AND word.dictionary_id = 1 group by letter.word_id HAVING count(letter.word_id) > 1
4. SELECT word.word FROM letter, word WHERE word.number_of_letters = 7 AND word.word_id = letter.word_id AND ((letter.letter = 'a' and letter.position=1) OR (letter.letter = 'b' AND letter.position=2) OR (letter.letter = 'i' AND letter.position=3)) AND word.dictionary_id = 1 group by letter.word_id HAVING count(letter.word_id) > 2

This database design allows us to search for words by a very specific letter pattern as can be seen from the tested queries. Meanwhile, the execution time of these queries (see 4 table) is noticeably bigger. This leads to the fact that this kind of search can be useful in certain situations; however, it should not be used for a simple word search as it can decrease the performance of the system.

Summary of the Research

As can be seen from research, the time to perform simple queries does not depend on the structure of the database, but only on how much data there is inside. The ascent of time to perform more complex queries can be seen with 5000 words, which generally gives more than 40,000 rows in the database.

The average performance of any of the requests depending on the design can be seen in the chart below.

The second type gives the best performance, because it is possible to more accurately describe the request, but these features are not enough. The fourth type requires too much time to process their full opportunities. The most

Table 3. ***Results of testing with third database design***

Site of the dictionary	Query execution time, ms				Number of returned rows by query			
	1'st query	2'nd query	3'rd query	4'th query	1'st query	2'nd query	3'rd query	4'th query
10 words	1	1	1	1	3	5	9	2
100 words	1	1	2	1	11	38	55	6
1000 words	2	2	8	6	51	186	354	52
5000 words	4	5	22	26	51	186	660	111

Table 4. ***Results of testing with the fourth database design***

Site of the dictionary	Query execution time, ms				Number of returned rows by query			
	1'st query	2'nd query	3'rd query	4'th query	1'st query	2'nd query	3'rd query	4'th query
10 words	1	1	1	1	3	5	3	1
100 words	1	1	4	5	11	38	4	1
1000 words	1	2	42	65	51	186	4	1
5000 words	5	6	350	520	51	186	4	1

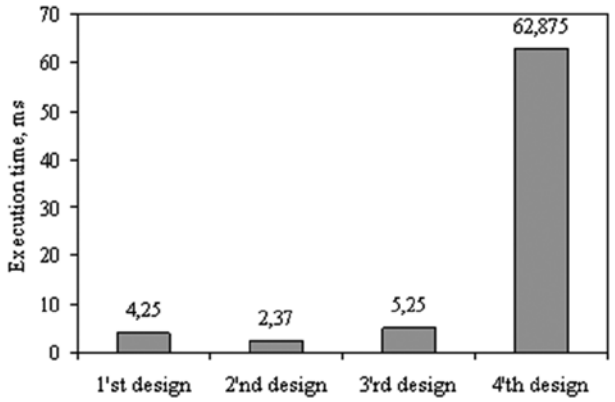

Fig. 5. Comparison of execution time in proposed database designs

balanced result is provided by the third type, because of quite exact queries that do not require too much time to perform.

No relationship between query execution time and number of returned results was noticed. The query execution time is more dependent on the number of records which must be examined. Therefore, an effective way to reduce the word search execution time is word length limitation, which reduces the number of examined records in the database as well as query execution time.

Conclusions

1. The string search algorithm can vary in its complexity and require text analysis, while SQL-based rational database allows for a word search according to a known pattern of set of letters more easily.
2. To design certain functionality for word search algorithms, limitations can be used depending on database structure. Therefore, the design of database schema is very important to ensure all the functionality and performance of the system.
3. All proposed database designs allow for word search in the database and the results of tested SQL queries give suitable results (all words which meet the pattern of set of letters are found). Therefore, the main choice of database design should be done according to the query execution time to get a certain set of words from the database.
4. The storage of letters in different tables for all the words in the dictionary gives an opportunity to search for words which have a set of letters. However, this database design is space consuming, as all words have more space in the database, and the execution time is bigger comparing to database designs without a separate table for letters storage. Therefore, this database design should be used just for specific problem solving to ensure good system performance.

References

1. R. Rivest, 1977, On the worst-case behavior of string-searching algorithms. SIAM J on Computing, 6:669-674.
5. R. Baeza-Yates, 1989, Improved string searching. Software-Practice and Experience, 19(3):257-271.
6. D.E. Knuth, J. Morris, and V. Pratt. Fast pattern matching in strings. SIAM J on Computing, 6:323-350
7. R. Boyer and S. Moore, 1977, A fast string searching algorithm. C.ACM, 20:762-772.
8. Ricardo A. Baeza-Yates, 1989,String Searching Algorithms Revisited, Proceedings of the Workshop on Algorithms and Data Structures, p.75-96, August 17-19.
9. JeffreyD.Ullman,<http://infolab.stanford.edu/~ullman/focs/ch08.pdf>
10. Okunis R., Makarevičius M. Populiariausios duomenų bazių valdymo sistemos.
11. Paliulis E., Baronienė R., 2010, Duomenų bazių valdymo sistemų analizė. Jaunųjų mokslininkų darbai. Nr. 3 (28). P. 94—99.
12. An introduction to the SQL procedure, <http://www.ats.ucla.edu/stat/sas/library/nesug99/bt082.pdf>.
13. SQL – LIKE Clause, <http://www.tutorialspoint.com/sql/pdf/sql-like-clause.pdf>.

RESEARCH OF WORD SEARCH ALGORITHMS BASED ON RELATIONAL DATABASE

Oleksij Volkov, Simona Ramanauskaitė

Summary

Word games are a simple and fun way to spend time. It helps to improve a person's vocabulary, expand erudition, train memory and intelligence, and develop logic and associative thinking. There exist different types of word games and different types of word games require a different game logic, and different word-finding algorithms. The optimization of word search algorithms is required to designa fully-functioning and efficient system. The aim of this work is to research the efficiency of different kinds of word search algorithms, using SQL-based relational database.

In this work, classic string search algorithms are analyzed and a different architecture of relational database structure was proposed to design a functional and efficient word search system. An efficiency analysis of proposed database designs was also analyzed to obtain metrics to define which architecture would be the most suitable for th design of word search system.

Key words: SQL; word search; database.

RELIACINĖS DUOMENŲ BAZĖS STRUKTŪRA PAREMTŲ ŽODŽIŲ PAIEŠKOS ALGORITMŲ TYRIMAS

Oleksij Volkov, Simona Ramanauskaitė

Santrauka

Žaidimai žodžiais yra paprastas ir linksmas laiko praleidimo būdas, kuris padeda asmenims plėsti savo žodyną, gerinti erudiciją, lavinti atmintį ir mąstymą, vystyti loginį mąstymą. Šiuo metu egzistuoja daug skirtingų žaidimų žodžiais, kurie reikalauja ir skirtingo žaidimo algoritmo, tinkamo žodžio radimo algoritmo. Todėl kuriant informacines sistemas, padedančias surasti ieškomą žodį pagal žinomus kriterijus, yra labai svarbu tinkamai parinkti ar optimizuoti žodžių paieškos algoritmą. Šio darbo tikslas – ištirti žodžių paieškos algoritmų, naudojančių SQL paremtas reliacines duomenų bazes, našumą.

Darbe apžvelgiami klasikiniai teksto paieškos algoritmai ir analizuojamos skirtingos reliacinių duomenų bazių architektūros, leidžiančios aprašyti turimą žodžių žodyną ir atlikti žodžių atranką naudojant skirtingo tipo paiešką. Darbe tiriamas pasiūlytų duomenų bazės schemų našumas, siekiant įvertinti tinkamiausią duomenų bazės architektūrą žodžių paieškos sistemai, kurioje gali būti vykdomi skirtingo tipo žodžių paieškos algoritmai, realizuoti.

Prasminiai žodžiai: SQL; žodžių paieška; duomenų bazė.

Įteikta 2013-05-14