Journal of Computer Science 7 (7): 1052-1059, 2011
ISSN 1549-3636

Multilingual Database Management System: A Performance Evaluation

Nurul Husna Mohd Saad and Hamidah Ibrahim
Department of Computer Science, Faculty of Computer Science and Information Technology,
University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

Abstract: Problem statement: The use of English as well as Arabic language is increasingly evident in the aspects of international business and finance. Therefore, this study explored the management of multilingual data in multilingual system in order to be able to cater two or more different speakers of Internet users. **Approach:** The proposed method is divided into two ends: The front-end that consisted of the Client and the Translator components and the back-end where the Management Module and the Database are located. In this method, a single encoded table is needed to store information and corresponding dictionaries are needed to store the multilingual data. The proposed method is based on the framework proposed in previous work with some modification to suit with the characteristics of the chosen languages on the case study. **Results:** Experimental evaluation had been done in storage requirement and mathematical analysis had been used to show the time of each of database operations for both of the traditional and the proposed method. **Conclusion/Recommendations:** The proposed method had been found to be consistently performed in the developed multilingual system.

Key words: Multilingual data, database performance, database management system, encoded representation, data dictionary

INTRODUCTION

The world in which we live that was once unconnected has now became globalized in every sense of words. Though, the most apparent effect that can be seen is from the aspect of language barriers. The driving force for this phenomenon has been the introduction of technology such as the Internet, fax machines, satellite TV, IP telephony, and mobile phones. And now with the era of computing at its peak, almost every single thing-from information access to commerce- has been computerized. But as we mentioned before, globalization has caused difficulties among countries with different language to communicate and in this case making information sharing impossible.

In this study, we are going to focus on two most popular Internet languages which are English and Arabic. As we all know, the de facto language for international business and finance is English. Arabic language, however, is currently gaining popularity not only in Arabic speaking countries but also for Curds, Persians and Aurdo-speaking Indians (Jannoud, 2007). In order to cater the non-native English speaking users, a multilingual system should be able to produce the information in the native language of the Internet users. To have this done, a database management system that can handle multilingual data efficiently is needed. Nevertheless, to translate English to Arabic and vice versa is not an easy task for a number of reasons. Our main worry is because Arabic sentences are usually long and contain only few punctuation marks. Due to the complexity of the Arabic syntax, sometimes Arabic sentences are syntactically ambiguous and require much effort when trying to resolve such ambiguities automatically (Sherif and Kondrak, 2007). Shirko *et al*. (2010); Shaalan *et al*. (2004) and Mohammed and Aziz (2011) have stressed the need for an efficient machine translation (Arabic-to-English and English-to-Arabic, respectively) in language processing due to the vast number of ways to express the same sentence in either languages. Although they have developed an effective machine translation, they are nowhere close to covering the issue of multilingual data management in database environment. Another problem deals with the occurrence of foreign words in Arabic text as transliteration, where it involves not only just proper names but also technical terms (Karimi *et al*., 2006). Through these observations, we have realized the significance of this issue and have developed a

Corresponding Author: Nurul Husna Mohd Saad, Department of Computer Science,
Faculty of Computer Science and Information Technology, University Putra Malaysia,
43400 UPM, Serdang, Selangor, Malaysia

multilingual database management system to ensure the availability of information in the native language of the Internet users.

In a multilingual database management system, user of any language speakers can search and retrieve data regardless of the language of those data. One might say that multilingual database is quite the same as real-time database since both databases deal in a multi-user environment. However, a real difference between multilingual database and real-time database transaction processing is their approach in concurrency control. In real-time database, concurrency control mechanism is important to ensure the consistency of the database while allowing a set of transactions to execute concurrently (Ali, 2006). Concurrency control in multilingual database on the other hand is quite the same as in conventional database, only that they have to ensure integrity between the languages involved. In this work, we concentrated mostly on design and implementation of the multilingual database management system, but did not concentrate on implementing the component of translator efficiently. A multilingual system has been developed that focuses on English and Arabic languages based on the framework proposed in (Hoque and Arefin, 2009) but with some modifications.

MATERIALS AND METHODS

In this study, we implemented the important parts of the system architecture in (Hoque and Arefin, 2009) and applied it with our own algorithm. In this Multilingual Database Management System (hereinafter called MDBMS), its system architecture is divided into two ends: The front-end and the back-end. The component for Client and Translator is situated in the front-end whereas the back-end comprises of the Management Module and the Database. The overall system architecture for MDBMS is shown in Fig. 1.

A Clinic System which has been developed specifically for this research to show the implementation of the MDBMS is placed in the Client component as seen in Fig. 1. In this component, users can provide input in various languages (for the sole of this research the languages have been limited to English and Arabic only) and view them in another language. In order for the Client to be able to display and treat the information in Arabic language correctly, a special Unicode character set (UTF-8) is needed to be implemented into the Client to manipulate them (Nandasara *et al*., 2008). The input provided by the users in specific language needs to be translated first before it can be inserted accordingly into the database. This is where the Translator component comes in handy. The Translator component is needed to translate the information into the target language with the help of a translator. For this research, Google Translate and Google Transliteration APIs have been used in the Translator component. As we have mentioned before, the Google Transliteration API is needed as we have to transliterate certain English words into Arabic words (e.g., proper names and technical names) and vice versa phonetically. However, transliteration and translation should not be confused their definitions, where translation involves a change in language while preserving their meaning. With transliteration, it is the sound of the words that are converted from one alphabet to the other. The accuracy of the translated and transliterated words is not our main focus here.

Management module: This module consists of a group of components. These components are Query Input/Response, Search Dictionaries, Dictionary-to-ET Mapping and ET-to-Dictionary Mapping. We have developed a new algorithm for each of these components while still retaining the definition of their functions which can be found in (Hoque and Arefin, 2009). The Management Module is responsible in performing the mapping and querying of languages.

Database: The idea of every database is to store information such as images, texts, and even media files. Just like everything else, the same goes for information storing where it will require spaces for them to be stored in. In this system architecture the Database is used as storage for the encoded tables and data dictionaries. Since the Client component needs to be manipulated using the UTF-8 character encoding in order to handle the Arabic words, the same goes for the Database, therefore the collation of the Database too needs to be set to utf8_unicode_ci.

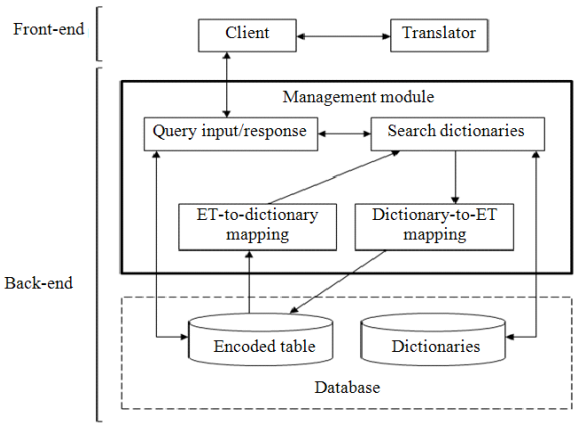

Fig. 1: System architecture for MDBMS

SSN	Name	Street	City	State	Zip	Gender	DOB	Age	Phone	Weight	Height
521457248	Burke	Gatlinburg	Louisiana	Austria	39901	Male	02-06-1989	53	1 36 828 9900-2780	52	128
192491319	Bethany	Fort Worth	New Hampshire	Mayotte	20672	Male	08-11-2009	40	1 21 420 5216-3048	112	197
235812716	Germane	Franklin	Iowa	Christmas Island	41908	Male	19-04-2003	68	1 75 756 4080-5255	81	216

Fig. 2: Storage of patient records in English

SSN	Name	Street	City	State	Zip	Gender	DOB	Age	Phone	Weight	Height
٥٢١٤٥٧٢٤٨	برق	اتلينبورج	لصيانة	أسترا	٣٩٩٠١	ذكر	١٩٨٩-٠٦-٠٢	٥٣	٢٧٨٠-٩٩٠٠ ٨٢٨ ٣٦ ١	٥٢	١٢٨
١٩٢٤٩١٣١٩	بتاني	فرط ورث	نو هامبشير	مايوت	٢٠٦٧٢	ذكر	٢٠٠٩-١١-٠٨	٤٠	٣٠٤٨-٥٢١٦ ٤٢٠ ٢١ ١	١١٢	١٩٧
٢٣٥٨١٢٧١٦	جرمان	فرانكلين	إوا	هريستماس يسلند	٤١٩٠٨	ذكر	٢٠٠٣-٠٤-١٩	٦٨	٥٢٥٥-٤٠٨٠ ٧٥٦ ٧٥ ١	٨١	٢١٦

Fig. 3: Storage of patient records in Arabic

SSN	Name	Street	City	State	Zip	Gender	DOB	Age	Phone	Weight	Height
521457248	1	1	1	1	39901	1	02-06-1989	53	1 36 828 9900-2780	52	128
192491319	2	2	2	2	20672	1	08-11-2009	40	1 21 420 5216-3048	112	197
235812716	3	3	3	3	41908	1	19-04-2003	68	1 75 756 4080-5255	81	216

Fig. 4: Encoded table storing multilingual information

Code	English	Arabic
1	Burke	برق
2	Bethany	بتاني
3	Germane	جرمان

Fig. 5: Dictionary pname for name attribute

Code	English	Arabic
1	Gatlinburg	اتلينبورج
2	Fort Worth	فرط ورث
3	Franklin	فرانكلين

Fig. 6: Dictionary pstreet for street attribute

To show the differences between existing Database Management System (hereinafter called DBMS) and the MDBMS, we have used the Clinic System to store information in both English and the native language Arabic using both the traditional and the MDBMS approaches. Consider the storage of Patient relation in English and Arabic languages that are shown in figures 2 and 3. These figures show the traditional approach of the DBMS for storing relations for each language. Hence, in this case data redundancy is relational to the number of languages support. The MDBMS uses only single encoded table to represent the multilingual information regardless of the number of languages support. The encoded representation of the relations Patient in English and Arabic (Fig. 2 and 3, respectively) is shown in Fig. 4. This idea of encoded representation is adopted from (Hoque and Arefin, 2009) where data are stored in information theoretic way in encoded form with minimum redundancy.

Those two relations are encoded into a single representation with respect to the type of their attributes. For attributes with numeric and alphanumeric fields, their values are represented directly into the encoded table without having to translate any of them at all. On the other hand, the values for attributes with text field have to be translated (or transliterate, depending on the word itself) and placed in the dictionaries. This would auto-generate a code that would represents the values in the encoded table. For example, Age, Weight and Height in Fig. 2 and 3 are the attributes with numeric fields while SSN, Zip, DOB and Phone are attributes with alphanumeric fields and their representation in the encoded table are shown in Fig. 4. The attributes for text fields such as Name, Street, City, State and Gender are encoded into the encoded table based on the corresponding code generated in the dictionaries. These dictionaries are shown in Fig. 5-9, respectively. These dictionaries are created by storing values that do not already exist in them so as to prevent data redundancy. Note that these rules that are implemented in this framework are the same as the rules applied in framework (Hoque and Arefin, 2009). The only difference being the algorithms used for each of the database operation.

Code	English	Arabic
1	Louisiana	لصيانة
2	New Hampshire	نو هاميشير
3	Iowa	إوا

Fig. 7: Dictionary pcity for city attribute

Code	English	Arabic
1	Austria	أسترا
2	Mayotte	مايوت
3	Christmas Island	هريستماس يسلند

Fig. 8: Dictionary pstate for state attribute

Code	English	Arabic
1	Male	ذكر

Fig. 9: Dictionary pgender for gender attribute

In Fig. 5, it shows the dictionary for the attribute Name where there are three name instances in English and their corresponding values in Arabic in the following columns. Another column is set to store the code (it is auto-generated each time a new value is inserted) that will represent these values in the encoded table. For example, let's consider the column Name in the encoded table (Fig. 4) and the dictionary Pname (Fig. 5). In the column Name of the encoded table, codes 1, 2 and 3 represent Burke or برق, Bethany or بثاني and Germane or جرمان, respectively. A data item that is stored in different languages in this dictionary is represented in the encoded table by the equivalent code and thus making the storage in the encoded table independent of the number of languages support. The same can be said for dictionaries Pstreet, Pcity, Pstate and Pgender which are shown in Fig. 6-9, respectively.

Database operations: In this study we proved that the MDBMS approach could support all the operations of normal databases (such as inserting, deleting and updating) and perform them efficiently. The insert and update operations should be treated with great care in order to prevent data redundancy and inconsistency in the dictionaries. When a new record is to be inserted (or updated) into the encoded table, if the data are of numeric or alphanumeric type, then they will be directly inserted into the encoded table without having to translate them.

```
ML-INSERT (R[n])
1.  for i ← 1 to n
2.      if R[i] is numeric or alphanumeric
3.          then i + 1
4.      end if
5.      else
6.          translate R[i]
7.          check corresponding dictionary
8.          if value exists
9.              then get code from dictionary
10.                 R[i] ← code
11.         end if
12.         else
13.             insert R[i] and its translated value
                into dictionary and get its code
14.             R[i] ← code
15.         end else
16.         i + 1
17.     end else
18. end for
19. insert R[n] into encoded table
```

Fig. 10: Algorithm for insert operation

```
ML-UPDATE (R[n])
1.  for j ← 1 to n
2.      if R[j] is numeric or alphanumeric
3.          then j + 1
4.      end if
5.      else
6.          if R[j] is the updated attribute
7.              then translate R[j]
8.                  check corresponding dictionary
9.                  if value exists
10.                     then get code from dictionary
11.                         R[j] ← code
12.                 end if
13.                 else
14.                     insert R[j] and its translated value
                        into dictionary and get its code
15.                     R[j] ← code
16.                 end else
17.         end if
18.         j + 1
19.     end else
20. end for
21. update R[n] into encoded table
```

Fig. 11: Algorithm for update operation

Alternatively for text data, after translating (or transliterating) them the system then will check related dictionaries for the existence of the data.

SSN	Name	Street	City	State	Zip	Gender	DOB	Age	Phone	Weight	Height
297688801	Noelle	Huntsville	South Carolina	Timor-leste	19141	Male	07-06-2008	43	1 33 183 7307-7041	77	241

Fig. 12: Record to be inserted

SSN	Name	Street	City	State	Zip	Gender	DOB	Age	Phone	Weight	Height
297688801	4	4	4	4	19141	1	07-06-2008	43	1 33 183 7307-7041	77	241

Fig. 13: Record in the encoded table

ML-DELETE (*R*)

1. delete *R* from the encoded table and not from the corresponding dictionaries

Fig. 14: Algorithm for delete operation

If the results return an empty set, then the data and their translated values will be inserted into the corresponding dictionaries, respectively. Otherwise, the system will get the key equivalent to the data searched and used it for their representation in the encoded table. Figure 10 and 11 show the algorithm used for insert and update operation, respectively. Since we have modified the framework adopted in (Hoque and Arefin, 2009), their algorithms could not be implemented with such ease in this research and hence is why the need for new algorithms for these database operations.

Consider the following example. To insert a new record (as shown in Fig. 12) the system first identifies SSN, Zip, DOB and Phone as alphanumeric attributes, Age, Weight and Height as numeric attributes and Name, Street, City, State and Gender as text attributes. These numeric and alphanumeric values are directly stored in the encoded table. Next, the dictionaries of Pname, Pstreet, Pcity, Pstate and Pgender are searched since they correspond to the attributes Name, Street, City, State and Gender in the encoded table, respectively. As dictionary Pname does not contain the name Noelle, the value Noelle is inserted into the dictionary with its translated value in Arabic language. A code (in this case, code 4) has been generated at the time of insertion, which is then used to represent Noelle in the encoded table. The same goes for dictionaries Pstreet, Pcity and Pstate since the values (Huntsville, South Carolina and Timor-leste) needed do not exist in those dictionaries. However, in dictionary Pgender, it already contains the value Male. Therefore, the information for attribute Gender is not inserted into the dictionary. The code that corresponds to Male is then grabbed (in this case, code 1) from dictionary Pgender. The record is then represented in the encoded table as shown in Fig. 13.

The delete operation is the simplest operation of all since it involves deleting from the encoded table only. Data items in the dictionaries that correspond to a record that is going to be deleted will not be removed from their storage since those data items might be needed for different types of operations in the future. By doing so, it will reduce greatly the time for insert and update operations since less data entry into the dictionaries will be needed at this point onwards. Figure 14 shows the algorithm used in the MDBMS for delete operation.

RESULTS

Patient schema has been considered in this experiment to measure the performance of the proposed MDBMS. A data generation program, Data Generator 2.1 (http://www.generatedata.com), has been used to generate data items for the schema Patient. Ten thousand of records were randomly generated for this experiment.

Space requirement calculation: The Patient schema has been implemented with five single dictionaries and one encoded table. The dictionaries for Patient schema are used to store the data items for the attributes Name, Street, City, State and Gender. Each of these dictionaries has three fields to store information in English, Arabic and an auto-generated code (for the purpose of mapping dictionary to encoded table).

For the Clinic System, by using the MDBMS approach, database is used as storage for text data and their translated values in the dictionaries. Likewise, database is needed too to store the codes that are representing the text, numeric and alphanumeric data in the encoded table. In contrast, database for the traditional DBMS approach includes the storage of information separately in each language.

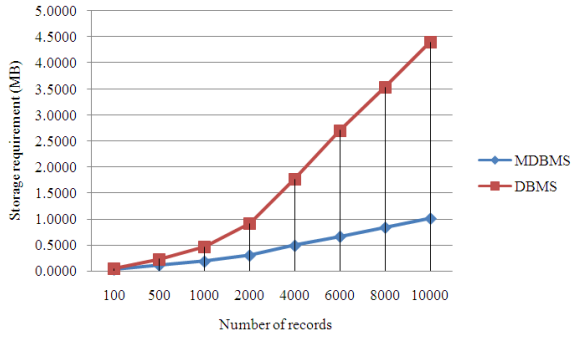

Fig. 15: Comparative storage requirement between traditional DBMS and MDBMS approach

Table 1: Description of Notation

Symbol	Description
T_{IO}	Time for insert operation
T_{DO}	Time for delete operation
T_{UO}	Time for update operation
T_T	Time to translate data
T_D	Time to delete a record
T_I	Time to insert a new record
T_U	Time to update a record
T_S	Time to search a dictionary
T_C	Time to insert a new record in a dictionary
L	Number of languages

The storage requirement for different number of records that uses the traditional DBMS approach is obtained by summing up the storage required to store information in English and Arabic whereas for the proposed approach, the storage requirement for different number of records is obtained by summing up the storage required to store data items in all the dictionaries and also the encoded table. For illustration, the summation of storage required for Fig. 2 and 3 would acquire the total storage requirement for the traditional DBMS approach whilst the summation of storage required for Fig. 4 (which is the encoded table) and Fig. 5-9 (which are the corresponding dictionaries for the encoded table) would acquire the total storage requirement for the MDBMS approach. From this experiment, a graph has been obtained as shown in Fig. 15.

Figure 15 illustrates the comparative storage between the traditional DBMS approach and the MDBMS approach. We can see that the MDBMS approach outperforms the traditional DBMS approach by about 77.08% in terms of storage requirement. These results confirm with the results obtained in (Hoque and Arefin, 2009) where with the increasing number of records, the storage requirement for the MDBMS would reduce significantly compared to the conventional DBMS, since at this point off, the dictionaries would have evolved enormously and therefore, further entry into the dictionaries is not necessary.

DISCUSSION

Query performance: The MDBMS that has been implemented has two parts for the time concern; one for searching and storing the necessary information in the dictionaries and another is for dictionary and encoded table mapping at the time of different operations. Our notations are summarized in Table 1.

Insert performance: For the existing DBMS, the insert operation is quite direct. The attributes to be inserted are translated into the target language and the attributes are then inserted into the database respective to the languages.

Let's say that R is a record to be inserted which contains N attributes (A_1, A_2, … , A_N). Hence, the insertion time for the conventional DBMS would be as follows:

$$T_{Io-C} = (\sum_{i=1}^{N} T_{Ti}) + L(T_I) \quad (1)$$

In MDBMS, first only text attributes are translated into the target language (Arabic). Since attributes of type numeric and alphanumeric are inserted directly into the encoded table in their original language (English), therefore, those attributes don't need to be translated. Next the corresponding dictionaries are searched to check the existence of the text data. Here, the insert operation is broken into two scenarios. The first scenario occurs when the values of the attributes involved in the query do not exist in the corresponding dictionaries. So, assume that from N attributes of record R, only M attributes need to be translated where M≤N. For the first scenario, let P be the attributes whose values do not exist in the corresponding dictionary where P ≤ M. The equation for the first scenario would be as follows:

$$T_P = (\sum_{i=1}^{P} T_{Si}) + L(T_{Ci}) \quad (2)$$

The second scenario occurs when the value of the attribute involved in the query exists in the corresponding dictionary. For this scenario, let Q be the attributes whose values exist in the corresponding dictionaries where Q≤ M. The equation for the second scenario would then be as follows:

$$T_Q = (\sum_{i=1}^{Q} T_{Si}) \quad (3)$$

From the Eq. 2 and 3, since P + Q = M, thus, it is safe to say that the insertion time for the MDBMS System would be as follows:

$$T_{Io-M} = (\sum_{i=1}^{M} T_{Ti}) + T_P + T_Q + T_I \quad (4)$$

These equations have clearly shown the difference of insert time between the existing (Eq. 1) and the proposed MDBMS (Eq. 4) where in the existing DBMS, all the attributes of a record have to be translated whilst in the proposed system only the attributes of type text have to be translated. Furthermore, the insertion of a record into a database for the existing DBMS has to be done numerously depending on the number of languages used whereas for the proposed MDBMS, the insertion of a record into a database has to be done only once.

Delete performance: The delete operation in both the traditional DBMS and MDBMS is quite straightforward. For the existing DBMS the delete operation has to be done for each language respectively. Hence, the delete time in this case would be as follows:

$$T_{DO\text{-}C} = L\ (T_D) \quad (5)$$

Whilst for MDBMS, the delete operation is done directly from the encoded database without the involvement of the dictionaries. For this reason, the delete time in this case would then be as follows:

$$T_{DO\text{-}M} = T_D \quad (6)$$

From these equations, it has been observed that delete time is not so much time consuming for the proposed MDBMS (equation (6)) compared to the existing DBMS (Eq. 5) since tuples are deleted directly from the database without the involvements of dictionaries.

Update performance: The update operation for both the traditional and the proposed approach is more time consuming than the other operations. For this experiment, let's assume that R is the record to be updated which contains N attributes (A_1, A_2, ... , A_N). From N attributes, only M attributes need to be updated and thus need to be translated. Hence, the update time for the traditional approach would be as follows:

$$T_{UO-C} = (\sum_{i=1}^{M} T_{Ti}) + L(T_U) \quad (7)$$

The opposite can be said for the update operation in the proposed MDBMS which is broken into three scenarios. The first scenario occurs when the values of the updated attributes are text data that do not exist in the corresponding dictionaries. So, assume that X attributes are attributes whose values do not exist in the corresponding dictionaries where X ≤ M. The equation for the first scenario would be as follows:

$$T_X = (\sum_{i=1}^{X} T_{Si} + \sum_{i=1}^{X} T_{Ci}) \quad (8)$$

The second scenario occurs when the values of the updated attributes involved are text data that have already existed in the corresponding dictionaries. From here, let's assume that Y attributes are attributes whose values have already existed in the corresponding dictionaries and where Y ≤ M. The equation for the second scenario would then be as follows:

$$T_Y = (\sum_{i=1}^{Y} T_{Si}) \quad (9)$$

For third scenario assume that from M attributes that need to be updated, Z is the number of attributes of numeric or alphanumeric type where Z ≤ M. As mentioned before, these types of attributes do not need any translation. Therefore, they are updated directly into the encoded table. Hence, no equation is needed for this scenario. Since X + Y + Z = M, thus, the update time for the proposed MDBMS would then be as follows:

$$T_{UO-M} = (\sum_{i=1}^{M-Z} T_{Ti}) + T_X + T_Y + T_U \quad (10)$$

From these equations, it is observed that the update time of the proposed MDBMS (Eq. 10) is slightly better than the update time for the existing DBMS (Eq. 7). This is because in the existing DBMS, the update operation needed to be performed in each of the databases separately to keep the consistency of information stored in different languages. But in the proposed MDBMS, the update operation only has to be performed on a single encoded table.

CONCLUSION

This study has implemented the Multilingual Database Management System approach in (Hoque and Arefin, 2009) with some modifications in its system architecture and the algorithm used for the insert, delete and update operations. The MDBMS in this study focused on English and Arabic languages, different from the MDBMS in (Hoque and Arefin, 2009). The

MDBMS approach performed consistently. The comparison to the traditional DBMS approach shows that the MDBMS approach needs less storage requirement. The MDBMS approach is found to be less time consuming in insert, delete and update operations than conventional DBMS approach. However, the MDBMS has not been developed to deal with typing errors. Let's say, a user intended to insert a name, for instance Noelle, which has already existed in the dictionary Pname. Instead, in this case the user has misspelled the name (i.e., Noeole) and during the insert operation, the MDBMS would have interpreted it as a new value since the search process in the dictionary Pname for the value Noelle would return an empty result. Therefore, the MDBMS would attempt to create a new record in the dictionary and this would obviously disrupt the consistency of the database. This little inaccuracy could be improved in future to further enhance the MDBMS.

REFERENCES

Ali, A.A., 2006. On optimistic concurrency control for real-time database systems. Am. J. Applied Sci., 3: 1706-1710. DOI: 10.3844/.2006.1706.1710

Hoque, A.S.M.L and M.S. Arefin, 2009. Multilingual data management in database environment. Malaysian J. Comput. Sci., 22: 44-63. http://mjcs.fsktm.um.edu.my/document.aspx?FileName=743.pdf

Jannoud, I.A., 2007. Automatic Arabic hand written text recognition system. Am. J. Applied Sci., 4: 857-864. DOI: 10.3844/ajassp.2007.857.864

Karimi, S., A. Turpin and F. Scholer, 2006. English to Persian Transliteration. Proceedings of Symposium on String Processing and Information Retrieval (SPIRE'06), Lecture Notes in Computer Science, Glasgow, UK, pp: 255-266. DOI: 10.1007/11880561_21

Mohammed, E.A. and M.J.A. Aziz, 2011. English to Arabic machine translation based on reordering algorithm. J. Comput. Sci., 7: 120-128. DOI: 10.3844/jcssp.2011.120.128

Nandasara, S.T., S. Kodama, C.Y. Choong, R. Caminero and A. Tarcan *et al*., 2008. An Analysis of Asian Language Web Pages. Int. J. Adv. ICT Emerging Regions, 1: 12-23. DOI: 10.4038/icter.v1i1.448

Shaalan, K., A. Rafea, A.A. Mmonem, and H. Baraka, 2004. Machine translation of English noun phrases into Arabic. Int. J. Comput. Process. Orient. Languages, 17: 121-134. DOI: 10.1142/S021942790400105X

Sherif, T. and G. Kondrak, 2007. Substring-Based Transliteration. Proceedings of the 45th Annual Meeting on Association for Computational Linguistics, Prague, Czech Republic, pp: 944-951. http://webdocs.cs.ualberta.ca/~kondrak/papers/acl07Gen.pdf

Shirko, O., N. Omar, H. Arshad, and M. Albared, 2010. Machine translation of noun phrases from arabic to English using transfer-based approach. J. Comput. Sci., 6: 350-356. DOI: 10.3844/jcssp.2010.350.356