Journal of Computer Science 2 (5): 460-465, 2006
ISSN 1549-3636

A Transformation-oriented Methodology to Knowledge-based Conceptual Data Warehouse Design

[1]Opim Salim Sitompul and [2]Shahrul Azman Noah
[1]Department of Computer Science, Faculty of Mathematics and Natural Sciences
University of Sumatera Utara, Medan-20155, Indonesia
[2]Department of Information Science, Faculty of Information Science and Technology
National University of Malaysia, 43600 UKM Bangi, Malaysia

Abstract: Applications of artificial intelligence (AI) technology in the form of knowledge-based systems within the context of database design have been extensively researched particularly to provide support within the conceptual design phase. However, a similar approach to the task of data warehouse design has yet to be seriously initiated. In this paper, we proposed a design methodology for conceptual data warehouse design called the transformation-oriented methodology, which transforms an Entity-Relationship (ER) model into a multidimensional model based on a series of transformation and analysis rules. The transformation-oriented methodology translates the ER model into a specification language model and transformed it into an initial problem domain model. A set of synthesis and diagnosis rules will then gradually transform the problem domain model into the multidimensional model. A prototype KB tool called the DWDesigner has been developed to implement the aforementioned methodology. The multidimensional model produces by the DWDesigner as output is presented in a graphical form for better visualization. Testing has been conducted to a number of design problems, such as university, business and hospital domains and consistent results have been achieved.

Key words: Conceptual design, data warehouse, knowledge-based CASE tool

INTRODUCTION

Data warehouse is an increasingly popular data repository system for enterprises. A data warehouse design is commonly supported by a conceptual data model called multidimensional model by which users could view data from different dimensions necessary for analysis purposes. In multidimensional model, data are represented in terms of facts and dimensions where each fact is associated to multiple dimensions. In this manner, facts are the focus of interest by which they are analyzed through the quantifying context stored in measures and the qualifying context determined through dimension levels[1]. Categorizing data along dimensions is a mean to organize them into hierarchical levels so that data can be viewed from their finer to coarser granularities[2].

The multidimensional model as a conceptual view plays an important role in data warehouse design. The model can be considered as a mediator between system analysts and users as they work together in formulating the data warehouse requirements. At this conceptual level, both the analysts and users could propose their ideas in terms that they understood, avoiding technical and theoretical jargons. In addition, the conceptual design is the basic building block for subsequent stages of data warehouse design. It is considered as the most important stage for the successful of the overall design where modeling errors could be detected early and the schema could be extended easily[1, 3].

While it has universally agreed that the implementation of data warehouse rest on the multidimensional model, little agreement has been said on how to carry out its conceptual design. The most popular opinion would be of using an existing ER model whereby the model is progressively translated and extended to include the dimensional functionality that is necessary in data warehousing. Although a number of methods supporting the aforementioned approach have been proposed[1,3-7], the capacity of these methods to be successfully implemented in the form of computer aided software engineering (CASE) largely remains a question.

Designing and implementing a data warehouse is a highly complex engineering task that asks for methodological support[8]. However, it is well-known among software designers that devising a design methodology is almost useless, unless it is supported with a CASE tool that could assist the designer in specifying and implementing the warehouse design[8, 9]. By using CASE tool, the designer will obtain several advantages in terms of productivity and quality of the data warehouse design produced.

Corresponding Author: Opim Salim Sitompul, Department of Computer Science, Faculty of Mathematics and Natural Sciences, University of Sumatera Utara, Medan-20155, Indonesia

Applications of artificial intelligence (AI) technology in the form of knowledge-based systems within the context of database design have been extensively researched particularly to provide support within the conceptual design phase. However, a similar approach to the task of data warehouse design has yet to be seriously initiated. In this paper, we proposed a design methodology for conceptual data warehouse design called the transformation-oriented methodology, which transforms an Entity-Relationship (ER) model into a multidimensional model based on a series of transformation and analysis rules.

Conceptual data warehouse design: Conceptual data warehouse design is a process to develop a data warehouse model that is represented in the form of multidimensional model. Research works on conceptual data warehouse design has started to receive more attention from the database community since the late 1990s with the aim to develop a conceptual schema, which is understandable by both users and system analysts as well as to provide a basis for the subsequent stages of the design process. One major approach taken by the database research community to the construction of this model is based on the ER model, which could be either extended or transformed into the multidimensional model. Research works on this area is then advancing to the development of automated conceptual designs that leads to the development of case tools for data warehouse design.

Several research works have been conducted to develop a methodology for designing conceptual data warehouse model based on the ER model. In general, the methodology used could be classified into two categories based on the design approaches, namely the ER extension and the ER transformation. The ER extension approach uses an ER model as input and extends it with additional constructs such that it can be mapped to the corresponding multidimensional model. Some examples of this approach are the Multidimensional Entity Relationship (ME/R)[10], the Structured Entity Relationship Model (SERM)[11] and the Event-Entity-Relationship model (EVER)[12].

The ER transformation approach also use the ER model as input but instead of extending the ER constructs, the ER model is subsequently transformed into the multidimensional model using different algorithms and techniques. The objective of this approach is to formulate a methodology for developing conceptual data warehouse design. Database research communities have initiated research works in this context since the late 1990s with the work by[4]. Subsequent works are presented in[1,3,5-7].

The transformation-oriented approach: Our methodology for the conceptual data warehouse design is based on the ER transformation approach called the transformation-oriented approach, which consists of five stages as illustrated in Fig. 1.

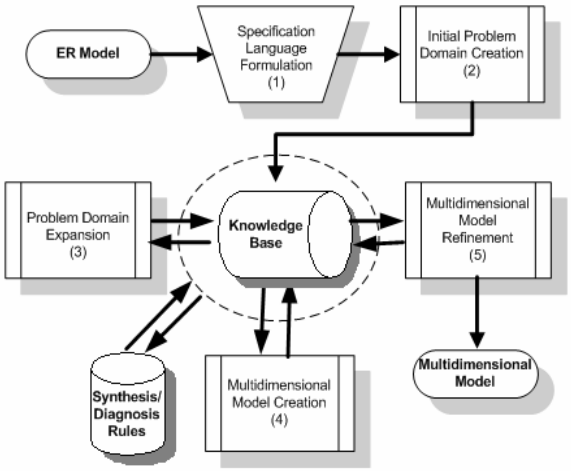

Fig. 1: The five-stage transformation-oriented approach

The specification language formulation stage is a manual process to translate the source input represented in the form of ER model into a specification language mode where each entity in the ER model is configured as a class structure with the name of the entity as the class name and its properties as the class properties. The entity properties specified in the class structure consist of attribute, identifier, subclass, aggregation and relationship. The translation of the ER model into the specification language model is guided by a set of syntax rules and the model resulted becomes the initial representation of the application domain (the problem domain model).

The initial problem domain creation stage is a stage responsible for the transformation of the specification language model created at the first stage into the initial problem domain model. The problem domain model is represented as a list of compound terms, which are ordered in property-entity-value pairs[13]. The initial problem domain will include the non-null value properties of each entity found in the specification language model. In addition, this stage is also responsible for the creation of a database in which the problem domain is stored.

The third stage is the analysis of the problem domain model in order to obtain new facts. The analysis is performed by a set of inference and translation rules using production and procedural rules[14]. Those analysis will cause some new facts are added into the database. The new added facts, however, may cause redundancies and inconsistencies within the database. Thus, in this stage some diagnostic tasks will be performed in order to prevent the database from such discrepancies. After the analysis-synthesis tasks are completed, this stage also performs an important task of classifying each entity attributes into numeric, temporal

and other categories as the basis for the creation of the multidimensional model as suggested in[7].

The fourth stage is the creation of the multidimensional model. The multidimensional constructs are created from the three categories of the attributes. Fact is created from an entity that has numeric attribute and will be called the fact entity. This fact will become a candidate fact schema, whereby the numeric attribute will become the fact attribute (measure). The dimensions of the multidimensional model are created from the temporal attribute and other attribute categories of the entity. The temporal attribute will become the temporal dimension and the other attribute will add other dimensions into the fact schema. In addition, the fact schema will also obtain dimensions from the relationship property of the fact entity. In this case, each one-to-many relationship between the fact entity and another entity will create a new dimension. Recursively, if there is a one-to-many relationship between the other entity and yet another entity, a new dimension level will be added, forming a dimension hierarchy.

The last stage is a refinement of the multidimensional model obtained from the previous stages. As those previous stages are automatic processes without any user interventions, the resulted model will only portray the basic multidimensional constructs similar to how they are established in the application domain model. Therefore, the refinement is necessary to further integrate user's requirements into the model by modifying measures, temporal dimension and dimension hierarchies.

The prototype knowledge-based tool: A prototype tool called the DWDesigner has been developed to implement the transformation-oriented approach. The tool was developed using a modular approach that enable the development of the tool being performed in an evolutionary way, on which current version of the prototype tool was developed based on refinement and enhancement of the previous versions. Current version of the DWDesigner is not meant as a complete implementation of a data warehouse design in which all stages of the design process are implemented. However, in implementing the conceptual stage of data warehouse design the DWDesigner has given consistent outputs.

The architecture of the DWDesigner consists of three layers, namely the user interface, the inference engine and the knowledge base as depicted in Fig. 2. The user interface facilitates interaction to users, namely the end user and the knowledge engineer. This interface provides a convenient way for the end user to perform the desired tasks by using a friendly graphical user interface from visual programming languages. The knowledge engineer, on the other hand, is the person who responsible for placing the knowledge into the system's knowledge base.

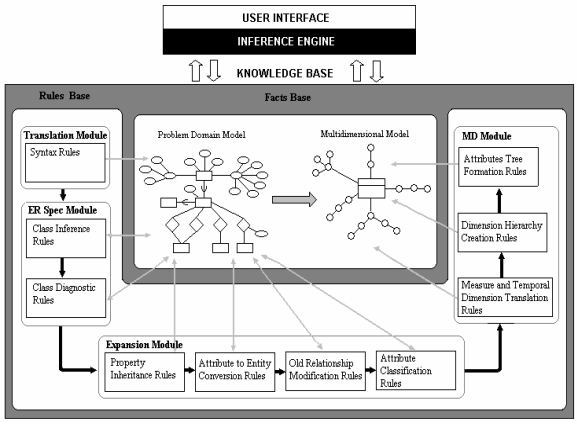

Fig. 2: Architecture of the DWDesigner

The inference engine serves as the inference and control mechanism for the overall system in order to generate the desired output. The inference mechanisms use the set of synthesis and diagnosis rules as well as the facts maintained in the knowledge base in the process of drawing a conclusion. The control mechanisms, on the other hand, responsible for the streamlining of the transformation process, such as starting the inference procedures, selection of rules to fire if there are more than one rule to trigger and how to conduct the search for solution.

The knowledge base is the lowest layer of the system's architecture, which interacts directly with the working memory of the computer system and the inference engine. Two components of the knowledge base, i.e. the rules base and the facts base, are the core of the knowledge base system and consume the major portions of Fig. 2. The facts base portion illustrates how the intermediate and the final representations of the knowledge are maintained in the working memory. The rules base portion shows how different transformation rules are distributed in a variety of modules and shows also the direct interactions between the rules in each module and the facts base.

RESULTS

We will look at the results obtained from testing the tool and describe the accuracy of the design tool in generating output in each stage of the transformation process until the multidimensional model is obtained and then shows how users could further refine the model to fulfill specific user's requirements. The ER model from the business domain taken as a sample for the input data for the DWDesigner tool is adapted from[5] as seen in Fig. 3.

To demonstrate how the tool generate the output from that input, we will see the result of each design stage by choosing the Sale entity from the ER diagram in Fig. 3 as a running example. In the first stage user should translate the ER model into the specification

language model, an example for the Sale entity is shown as follows:

```
CLASS "SALE"
ATTRIBUTE (("S-Number": Integer) ("Date": Date) ("Income": Float))
IDENTIFIER ("S-Number")
SUBCLASS NIL
AGGREGATION NIL
RELATIONSHIP (("Cust-Sale" "CUSTOMER" "NIL" "(0 1)" "(1 n)")\
              ("Sale-Stor" "STORE" "NIL" "(1 1)" "(1 n)")\
              ("Sale-Item" "ITEM" "NIL" "(1 1)" "(1 n)"))
End-Class
```

In the second stage the tool generates an entities list, which records each entity name and its properties and saves the entities list as an intermediate output in the form of a text file. A portion of the file containing the Sale entity is given in the following:

```
Entity Name: "SALE"
    Attribute(s):
        S-Number: Integer
        Date: Date
        Income: Float
    Identifier(s):
        ("S-Number")
    Subclass(es): NIL
    Aggregation(s): NIL
    Relationship(s):
        ("Sale-Item" "ITEM" "NIL" "(1 1)" "(1 n)")
        ("Sale-Stor" "STORE" "NIL" "(1 1)" "(1 n)")
        ("Cust-Sale" "CUSTOMER" "NIL" "(0 1)" "(1 n)")
```

Subsequently, in the third stage the tool performed a sequence of steps of creating initial problem domain model, expanding the model by deriving more facts from subclasses and superclasses, creating new entities, inheriting new properties, generating an objects list and saving the object description into an intermediate output. A portion of this file for the Sale entity is given in the following:

```
Object name: SALE                    Relationship(s):
Attribute(s):                        (Name . Sale-Item)
Numeric Attribute:                   (Participating-obj . ITEM)
 (S-Number . Integer)                (Rel-Attribute . NIL)
 (Income . Float)                    (First-constraint . (1 1))
Date Attribute:                      (Second-constraint . (1 n))
 (Date . Date)                       (Name . Sale-Stor)
Other Attribute: NIL                 (Participating-obj . STORE)
Identifier(s): ("S-Number")          (Rel-Attribute . NIL)
Direct Subclass(es): NIL             (First-constraint . (1 1))
Indirect Subclass(es): NIL           (Second-constraint . (1 n))
Direct Superclass(es): NIL           (Name . Cust-Sale)
Indirect Superclass(es): NIL         (Participating-obj . CUSTOMER)
Aggregation(s): NIL
```

In the fourth stage the tool generates a fact list containing the candidate fact schemata of the multidimensional model from each entity and saves them as output. Finally, in the last stage user could refine the resulted fact schema of each entity. The refinement is indeed necessary because otherwise it will only produce the multidimensional model based on the entity properties available in the ER model of the design sources. For example user might want the fact

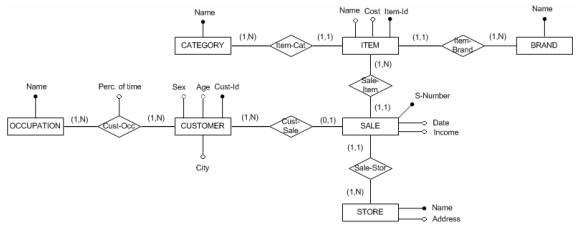

Fig. 1: ER diagram for the retail business domain

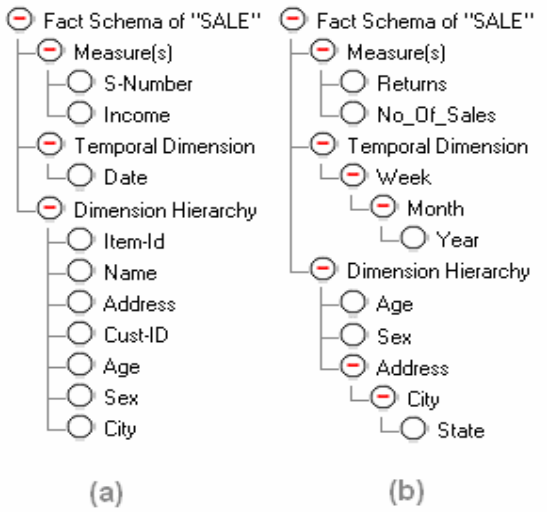

Fig. 2: Fact schema of sale (a) Before refinement (b) After refinement

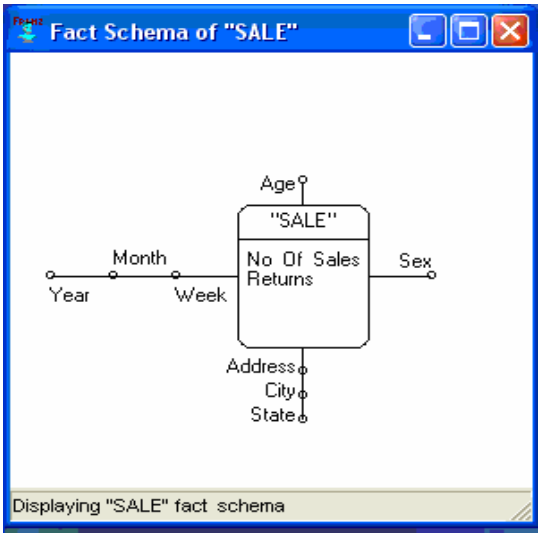

Fig. 3: Graphical output for the sale fact schema

schema is measured based on the returns of sales and the number of sales so that the two measures could be analyzed in a time interval of week, month, or year from several dimensions such as customer's age, sex and customer's address. The customer address could be further aggregated into city and state. The multidimensional model for the Sale fact schema before

Table 1: Comparison of output

Tool/ Approach	Facts	Measure	Temporal Dimension	Dimension Hierarchy
Benchmark (DFM Method)	Admission	n. of.admissions Value n. of days Score	Month Quarter Semester Year	Ward Division Hospital Sex Town Age5 Age10 Diagnosis Outcome Drg Type Rate Threshold Requiring physic Type of surgery
Automated Tool (DWDesigner)	Admission	Code Outcome n-days	Date	Ward Division Hospital PatCode Sex Name Physician Town Diagnosis Drg Type Rate Threshold

and after the refinement is shown in Fig. 4 and the multidimensional model is depicted as in Fig. 5.

User performs the following refinement in order to arrive at the desired multidimensional model, namely refining measures by modifying S-Number and Income into Returns and No_Of_ Sales, refining Temporal dimension by modifying Date into Week Month Year and refining dimension hierarchies by pruning Item-Id, Name and Cust-ID and aggregating Address, City and adding State.

Testing and evaluation: In order to evaluate the consistency of the outputs generated by the DWDesigner, we have performed a series of testing using correct and incorrect data sets from several domains such as university, business and hospital. The test have shown that the DWDesigner is capable of synthesizing and diagnosing correct data in order to produce desired outputs. In addition, we have also tested the system to detect the design inconsistencies found in the set of incorrect data and resolve those errors either automatically or by initiating a dialog with user[15].

Some evaluations are also performed by comparing outputs generated by the DW Designer with outputs produced by other approaches, which are taken as benchmarks. For example, Table 1 shows output generated by the DFM approach and output generated by the DW Designer. The multidimensional constructs taken as a comparison is the Admission fact scheme.

As can be seen from Table 1, the DWDesigner generated similar results to those resulted by the DFM approach. For the measure constructs, for example, the DFM approach provides n. of days as the sum of number of days of the admission. Since this is a derived measure formulated by user, thereby the DWDesigner's user could refine the n-days measure generated by the tool to reflect the same task. The rest of the measures provided in the Admission fact scheme of the DFM approach are also derived measures, namely: number of admissions is the count of admissions, value is the sum of value from the Has entity's attribute and score is the sum of weight from Drg entity's attribute. Using a similar approach, the DWDesigner's users could use the admission code in order to count the number of admissions and derived similar measures such as value and score from the Has and Drg entities. In addition, using DWDesigner user could calculate the sum of outcomes obtained from each admission by using the measure outcome provided, which is recorded as dimension in the DFM approach.

Temporal dimension is a construct that should obtain more attention from the DWDesigner since this construct is very dependable on user's preference. Indeed, users may always need to refine this construct in order to fulfill specific requirements. Referring to the DFM approach, the temporal dimension of the Admission generated by the tool could be refined in order to capture the time interval needed for the analysis of the admissions.

Except for the Age dimension that is categorized into Age5 and Age10 in the DFM model, both the DFM approach and the DWDesigner produce very similar results in terms of dimension hierarchies of the Admission fact scheme. A minor differences found in both dimension hierarchies are the Type of surgery recorded in the DFM approach, which is optionally added to the hierarchy if the user is considering only the main operations so that the Surgery entity is included into the hierarchy through the Causes relationship. This dimension could not automatically be included into the dimension hierarchy produced by the DWDesigner since Causes relationship between the Admission entity and Surgery is recorded as a many-to-one relationship. Therefore further refinement from the user is necessary to put this dimension into the dimension hierarchy. Another minor difference is the patCode and Name dimensions found in the Admission fact produced by the DWDesigner, which has been grafted and pruned in the dimension hierarchy produced by the DFM approach.

Results obtained from the comparison therefore show that the DWDesigner is delivering correct and consistent outputs.

CONCLUSION

We have shown that the tedious tasks of designing the conceptual design of a data warehouse are indeed could be automated. Using the transformation-oriented approach, the automated tool could generate the data warehouse model automatically with only minimal user interactions. The tool incorporates a set of synthesis and diagnosis rules to check and resolve inconsistencies that might exist during the design process. Furthermore, the tool also provides a refinement facility to enable user refining the multidimensional constructs in order to meet specific requirements. Some test cases perform on the automated tool also shown that the tool produce correct and consistent outputs.

REFERENCES

1. Hüsemann, B., J. Lechtenbörger and G. Vossen, 2000. Conceptual data warehouse design. Proc. Intl. Workshop on Design and Management of Data Warehouse (DMDW '2000), pp: 6-1-6-11.
2. Agrawal, D., A. El Abbadi, A. Singh and T. Yurek, 1997. Efficient view maintenance at data warehouse. Proc. ACM SIGMOD Intl. Conf. on Management of Data, pp: 417-427.
3. Tryfona, N., F. Busborg and J.G.B. Christiansen, 1999. starER: a conceptual model for data warehouse design. Proc. ACM 2nd Intl. Workshop on Data Warehousing and OLAP, pp: 3-8.
4. Golfarelli, M., D. Maio and S. Rizzi, 1998. Conceptual design of data warehouses from E/R schemes. Proc. 31st Hawaii Intl. Conf. on System Sciences, pp: 334-343.
5. Cabibbo, L. and R. Torlone, 1998. A logical approach to multidimensional databases. Proc. 6th Intl. Conf. on Extending Database Technology (EDBT'98), pp: 253-269.
6. Moody, D. and M.A.R. Kortink, 2000. From enterprise models to dimensional models: a methodology for data warehouse and data mart design. Proc. of Intl. Workshop on Design and Management of Data Warehouses (DMDW'2000), pp: 5-1 – 5-12.
7. Phipps, C. and K.C. Davis, 2002. Automating data warehouse conceptual schema design and evaluation. Proc. 4th Intl. Workshop on Design and Management of Data Warehouses 2002 (DMDW'2002), pp: 23-32.
8. Hahn, K., C. Sapia and M. Blaschka, 2000. Automatically generating OLAP schemata from conceptual graphical models. Proc. 3rd Int.. Workshop on Data Warehousing and OLAP (DOLAP'00), pp: 9-16.
9. Golfarelli, M., S. Rizzi and E. Saltarelli, 2002. WAND: A CASE tool for workload-based design of a data mart. Proc. Decimo Convegno Nazionale su Sistemi Evoluti Per Basi Di Dati, pp: 422-426.
10. Sapia, C., M. Blaschka, G. Höfling and H. Dinter, 1998. Extending the E/R model for the multidimensional paradigm. In: Kambayashi, Y., Lee, D.K., Lim, E-.P., Mohania, M.K. & Masunaga, Y. (Eds.). Advances in Database Technology. Proc. 1st Int. Workshop on Data Warehouse and Data Mining (DWDM'98), LNCS 1552, pp: 105-116.
11. Boehnlein, M. and A. Ulbrich-vom Ende, 1999. Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems. Proc. ACM 2nd Intl. Workshop on Data Warehousing and OLAP (DOLAP'99), pp: 15–21.
12. Bækgaard, L., 1999. Event-entity-relationship modeling in data warehouse environments. Proc. ACM 2nd Intl. Workshop on Data Warehousing and OLAP (DOLAP'99), pp: 9-14.
13. Luger, G.F. and W. A. Stubblefield, 1998. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. 3rd Edn. London, Addison-Wesley.
14. Sitompul, O.S. and S.A.M. Noah, 2003. Rules for the automatic translation of ER model into multidimensional model. Proc. Conf. on Intelligent Systems and Robotics (CISAR 2003), pp: 98-105.
15. Sitompul, O.S., S.A.M. Noah, 2005. Integrating user's requirements in automated conceptual data warehouse design. In Kotsis, G., Taniar, D., Bressan, T., Ibrahim, I.K., Mokhtar, S. (Eds.) Proc. 7th Int.. Conf. on Information Integration and Web-based Applications and Services (iiWAS'2005), pp: 835-845. Austrian Computer Society, Kuala Lumpur, Malaysia.