DESIGNING A MULTILINGUAL CONTENT AUTHORING AND INFORMATION RETRIEVAL MODEL

SELVİHAN NAZLI YAVUZER

JULY 2005

DESIGNING A MULTILINGUAL CONTENT AUTHORING AND INFORMATION RETRIEVAL MODEL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL
OF
BAHÇEŞEHİR UNIVERSITY
BY

SELVİHAN NAZLI YAVUZER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

JULY 2005

Approval of the Graduate School of (Name of the Graduate School)

(Title and Name)

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

(Title and Name)

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

(Title and Name)	(Title and Name)
Co-Supervisor	Supervisor

Examining Committee Members

....	
....	
....	
....	
....	

ABSTRACT

DESIGNING A MULTILINGUAL CONTENT AUTHORING

AND

INFORMATION RETRIEVAL MODEL

Yavuzer, Selvihan Nazlı

M.S. Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Orhan Gökçöl

July 2005, 68 pages

Globalization through Internet has aroused the need for multilingual presentations especially within medium-large scale businesses. Although current practices on database-driven multilingual system development manage to provide multilingual content management and authoring, underlying database structures do not serve for the optimum performance and for maximum content automation. The objective of this thesis is to offer a database model, which classifies the pieces of information according to language-dependency and applies further normalization in order to provide most convenient means of data organization. The thesis pays attention to load on data source due to the number of users; as such a meticulous multilingual implementation is likely to occur in large-scale systems. The tests performed in the study employs a simple desktop application that simulates multiple users connected to the system, that is, the data source, each of which requests a number of transactions from the server. The tests are performed on two database models, replication model and normalized model offered in the thesis. Collected data was mixed; however, the results were enough to observe that offered model follows a rational rate of increase where the replication model peaks at a point where the number of users exceeds a certain value.

Key words: multilingual, cross-lingual, content management, content authoring, information retrieval, XML, database design, benchmark

ÖZET

ÇOKLU DİLDE İÇERİK YÖNETİMİ VE BİLGİ BULMA MODELİ TASARIMI

Yavuzer, Selvihan Nazlı

Yüksek Lisans, Bilgisayar Mühendisliği Bölumu

Tez Yöneticisi: Yrd. Doç. Dr. Orhan Gökçöl

Temmuz 2005, 68 sayfa

İnternet'in küreselleşmeye etkisi orta-büyük ölçekli işletmelerde kaynakların bir çok dilde sunulması ihtiyacını ortaya çıkarmıştır. Çoklu dilde sistem geliştirme uygulamaları üzerine şu an ki pratikler, çoklu dilde içerik yönetimini mümkün kılmalarına rağmen, veritabanı altyapısı ile optimum performans ve maksimum içerik otomasyonunu sağlamamaktadır. Bu çalışma, sistemin sunacağı tüm bilgilerin dil bağımlılıklarına göre guruplandırılarak veri tabanında daha fazla normalizasyona gidilmesiyle veri organizasyonu ve veri erişimi için daha uygun bir metod önermektedir. Veri modeli değerlendirilirken, titiz bir çoklu dil çalışmasının büyük ölçekli sistemlerde yapılması olasılığının daha yüksek olduğu düşünülerek, kullanıcıların veri kaynağına gönderdikleri iş yüklerine önem verilmiştir. Bu çalışmada yapılan testlerde, sisteme bağlanan ve herbiri belli sayıda işlem talebinde bulunan kullanıcıları simüle eden basit bir masaüstü uygulaması geliştirilmiştir. Testler iki farklı veri modelinde üzerinde gerçekleştirilmiştir; şu anda çoklu dil uygulamalarında kullanılan kopyalama modeli ve çalışmada önerilen ayırma modeli. Toplanan veriler farklılık göstermesine rağmen, ayırma modeli açıklanabilir bir artış grafiği çizerken, kopyalama metodunun kullanıcı sayısı belli bir limiti aştığında ani sıçramalar yapma eğilimini gözlemleye yeterlidir.

Anahtar Kelimeler: çoklu dil, diller arası, içerik yönetimi, içerik oluşturma, bilgi bulma, XML, veritabanı tasarımı, karşılaştırma testi

To My Parents

ACKNOWLEDGMENTS

This thesis is dedicated to **my parents** for their patience and understanding during my master’s study and the writing of this thesis. I am also grateful to **my sister**, who provided moral and spiritual support.

I would like to express my gratitude to **Asst. Prof. Dr. Orhan Gökçöl**, for not only being such a great supervisor but also encouraging and challenging me throughout my academic program.

I also wish to thank **Asst. Prof. Dr. Adem Karahoca**, who helped me on various topics in the area of database management systems, for his advice and time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Multilingual
MLIR	Multilingual Information Retrieval
CLIR	Cross-Lingual Information Retrieval
XML	Extensible Markup Language
XSL	Extensible Style Language
XSLT	Extensible Style Language Transformations
XSD	XML Schema Definition

1 INTRODUCTION

Current practices on multilingual system development do not evidently address the problems related to multilingual content management and authoring. Despite the significance of multilingual development, previous implementations generally lack performance in storage size, manageability, authoring automation, and structural ability.

A simple method for multilingual web site development is to create translations of each page in the native language and each file name has a language identification suffix such as '*index_tr.html*' for Turkish, '*index_en.html*' for English, etc. Example site created for bilingual content and crawled, the structure is shown in Figure 1.1.

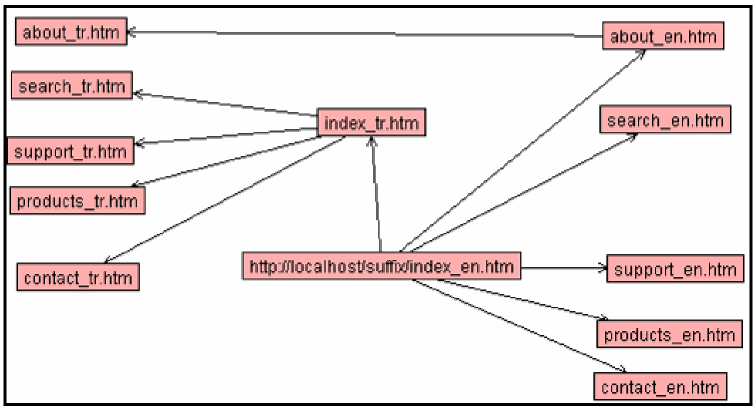


Figure 1.1 Structure for a Bilingual Website with Page Translation

In a system using page name suffixes for multilingual support, if a modification is to be done, n-1 complete translations (native language is ignored) and n updates (as each page needs to be modified) are needed where n is the number of languages the site supports. Furthermore, this method requires storage of multilingual images with proper suffixes, which along with the suffixed html files, creates a heap of files in the root directory. Starting with this file mass problem, a derivation of the previous method is to create a separate directory for each language that is accessible from top level, and inside each language directory the same structure is replicated. Example site created for bilingual content and crawled, the structure is shown below.

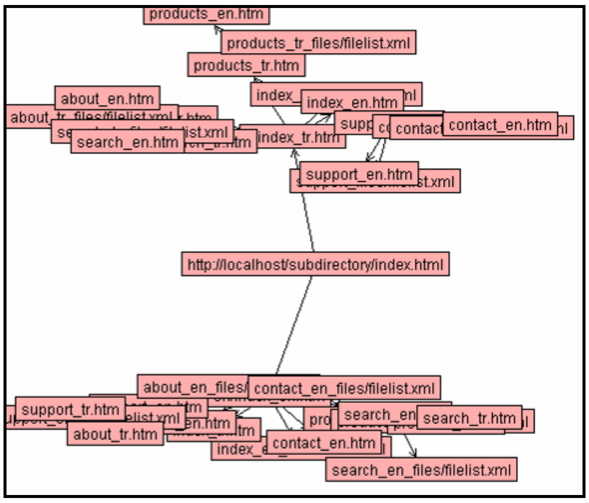


Figure 1.2 Structure for a Bilingual Website with Page Translation and Directory Separation

However, this change in file system does not eliminate the fact that this implementation requires manual maintenance performed by the developer. The developer should make sure that each link on each page points to the proper page in the same language, and that each page is correctly linked to their translations in each supported language. Besides, it is again the developer's responsibility to assure that a change in a language is propagated to all other languages.

Briefly, the above two methods based on replication of data are not proper solutions to multilingual problem. The site structure as shown in above figures becomes complicated even for bilingual development. It is, no doubt, not efficient to use any of these methods for a commercial site with 50 pages and supports 5 languages, which makes a total of 250 files (5 replicates of each page) in the file system excluding the number of replicated images and other related documents. In addition to storage size and structure, such content is hard to maintain and manage.

1.1 Motivation

With the onset of the Internet, traditional constraints such as geographical barriers have been abolished. Companies are no longer prevented from doing business simply because they are located in different parts of the world. However, there still remain barriers such as taxation, shipping, business practices, language and cultural differences that still prohibit a true global marketplace. To best serve customers, it is becoming increasingly important that companies address the wide range of business practices, languages and cultures when doing business through the Internet. This can be achieved by providing customers with content in multiple languages. As searchable multilingual text databases have become available globally, multilingualism and multilingual text retrieval have been a focus of research in the past years. This thesis presents an evaluation for a database that supports multilingual content in the best way possible. The motivation for such a work comes from the diversity of language of the computer users, especially Internet users as well as the diversity of the content and authors, and the fact that there are no definite answers to stated multilingual system development problems.

1.2 Related Work

More recent practices do not only focus on content management aspect, but they also integrate content authoring process with multilingual support. Such systems referred as MDA (Multilanguage Document Authoring) systems. Some early implementations of MDA systems were interactive as they allowed users to dynamically modify their internal representations; after any modification, the data is regenerated to reflect the choice given by the author. This is especially important in multilingual generation because a single change can be propagated to all other supported languages without manual response. DRAFTER is an early system of this type in which the authoring is performed at the semantic structure level (Paris et al.1995). Power and Scott (1998) presents a newer idea and rather than having the user modify the semantic representation directly, the generated text itself is used as interface to the semantic representation. Certain parts of the content are associated with menus presenting different choices for updating the semantic representation, which provides that the user never needs to access to the semantic representation directly.

Chevreau proposes a system that (2001) interactively generates weather reports in various languages; the choices in this system are more of a syntactical nature because of the semantic information is extracted directly from the weather forecast system. Ranta (1994, 2002) had developed an MDA approach at XRCE (Xerox Research Center Europe) which is a supporter of the Grammatical Framework (GF) and has been used on small-scale pharmaceutical documents. A well-formed semantic representation is the basis of GF-MDA. As MDA has a formal notion of well-formed semantic structure, it has theoretical and practical advantages such as; it has direct connection to XML theory and practice as it has similar grammar to a DTD or a Schema, it has both abstract and actual well-formed documents. MDA-XML, a multilingual authoring prototype (XRCE) fits well into an XML framework, as it is a variant representation of XML Schemas already in use where rules are implemented in XML.

In another XML based variation of multilingual generation, Tonella introduces MLHTML (2002). The technique proposed in MLHTML was to construct web sites so as to align the information provided in different languages and to make it consistent across languages. MLHTML extends XHTML with one additional tag (<*ml*>) for multilingual representation. Construction of multilingual pages using MLHTML involves two phases; first phase is page alignment which aims at aligning the pages in different languages, in the second phase, the aligned pages are merged into one MLHTML file where each multilingual contents is placed in a special <*ml*> tag.

XML based implementations are useful when all page content is assumed to be between html tags. However, no XML based technique has been able to include relational data models for multilingual management where content is dynamically generated and managed using data sources. Multilingual support for web driven database applications, on the other hand, is implemented in a few different methods. One method is to create a separate database for each supported language where the language of the database is mentioned in database name. This may provide a simple solution to multilingual web driven database applications, yet it is not easier to manage multiple databases. Another implementation to provide the functionality of data in the right language is to keep the database neutral in terms of language. And for an object, that is a database table, that includes multilingual items (attributes) a predefined language code is included in the primary key. Using this method, each table that has multilingual properties has an additional attribute for language code, and the record number and language code represents the primary key pair. An example is;

Table 1.1 Language-Neutrel Database Table

Record No	**Language Code**	**Data**
001	En	Graduate Student
001	Tr	Lisansüstü Öğrenci
001	Es	Alumno de posgrado

Although this system considers categorization for monolingual data, this categorization only includes individual names or other unique names. All other information is categorized into multilingual tables.

Before designing a multilingual information system, the most important success factors of a multilingual information system should be examined that are the degree of authoring automation and cultural customization it offers and cross-lingual processing capability.

Cross-Lingual Information Retrieval

Cross-lingual text retrieval is implemented with two dominant approaches 1) dictionary translation using machine-readable multilingual dictionaries and 2) automatic extraction of possible transition equivalents by statistical analysis of parallel or comparable corpora [1]. At this stage, in most cross-lingual information retrieval (CLIR) system, users are expected to formulate query specifying their information need by producing appropriate keywords. However, producing the appropriate set of keywords is a difficult task considering the limited linguistic skills of users.

To overcome the cultural and lingual barriers in multilingual information retrieval data mining techniques for keyword classification and cross-lingual queries are implemented in retrieval processes.

For multilingual text retrieval (MLTR), basic data-mining methods such as fuzzy multilingual keyword classification in which fuzzy clustering (Fuzzy c-means) is applied to obtain a classification of multilingual keywords by concepts are used. Labeling each concept with native language of the target user and associating it with relevant multilingual documents develop a multilingual concept directory.

A fuzzy clustering generates a partition of a multilingual keyword data set for revealing cross-lingual conceptual relationship among keywords with additional concept membership values.

The fuzzy c-means algorithm developed by Bezdek aims at minimizing the objective function

$$J(X,U,v)=\sum_{i=1}^{c}\sum_{k=1}^{n}(\mu_{ik})^{m}\,d^{2}(v_i,x_k) \tag{1}$$

under the constrains

$$\sum_{k=1}^{n}\mu_{ik}>0 \quad \text{for all } i\in\{1,\ldots,c\} \tag{2}$$

and

$$\sum_{i=1}^{c} \mu_{ik} = 1 \qquad \text{for all } k \in \{1,\ldots,n\} \tag{3}$$

Where $X = \{x_1, x_2, \ldots, x_n\} \subseteq R^p$ is the set of objects, c is the number of fuzzy clusters, $\mu_{ik} \in [0, 1]$ is the membership degree of object x_k to cluster I, v_i is the prototype (cluster center) of cluster i, and $d(v_i, x_k)$ is the Euclidean distance between prototype v_i and object x_k. The parameter $m > 1$ is called the fuzziness index.

Using cross-lingual queries allows the system users produce sets of keywords or phrases where the keywords may differ in language although they serve for the same search need. In systems where cross-lingual queries are supported, term extraction and then term translation is applied in order to reduce the unnecessary online translation processes. There are two types of term extraction methods employed. The first is language-dependent linguistics based method that relies of lexical analysis, word segmentation and syntactic analysis to extract named entities from documents. The second method is the language-independent statistics-based method that extracts significant lexical patterns without length limitation, such as the local maxima method and the PAT-tree-based method.

Authoring Automation

The authoring process is monolingual but the results are multilingual. In a multilingual information system, there are numerous issues related to content management and authoring. Management focuses on the engineering side, maintaining consistency of information (translations) across different parts of the system, which is time consuming and error prone. In a large system, it is vital that the system can realize the missing information (translations) and can convey the translation to related author. In semi-automatic system mostly used for large scale content, the system is capable of generating the translations of newly added or modified information using localization and globalization software and services such as online translators. In both manual and semi-automatic systems, it must be assured that even the author works in the language s/he knows, the system implicitly builds a language-independent representation of the document content.

Cultural Customization

On the other hand, cultural customization of the system, that is, how information is presented to the user is another issue on multilingual development. Current practice, as previously mentioned, uses different techniques from separate cultural design for each representation object (replicated content files) to XSLT and XML technology based on the size and goal of the system. Cultural variations are mostly implemented by the use of multilingual personalized information objects. The ultimate goal of personalized information objects is to experiment with an approach to multilingual generation that leverages on grammar commonalities across different languages, therefore allowing for a faster resource development and easier maintenance. It is observed that many languages share the same basic systems in functionality, but differ in their realization, or the way specific predicates or relations are phrased. Using multilingual objects, a separate rule-based component for each language is defined to achieve inflectional morphology.

1.3 Roadmap

This study examines a multilingual system in components in order to obtain an improved database model required to achieve the system's functionality. Motivation to do such a work and previous research on multilingualism has been presented so far. In the rest of this paper, we initially take a look at multilingual content management issues. Then, the database perspective of a multilingual system is introduced and two database implementations are described. The test performed on both models is given in details including the testing environment and testing software. Collected results are discussed. Finally, a simple model multilingual content authoring system is proposed.

2 A MULTILINGUAL CONTENT MANAGEMENT SYSTEM

2.1 Background

The problem of presenting multilingual content to a range of audiences involves more than translating the text from one language to another. When translating text in computer-based environments, additional problems arise, such as the size of text blocks and field length in application software or databases, as the translated text can expand by %40. For example, a character in a Roman language takes 1 byte, while a character in Chinese takes two bytes. While the multilingualism of the content has an impact on the size of application data blocks, providing multiple translations of the content also requires interfaces to keep translation under control. In a multilingual system, distinct records and translations of records must be accurate in terms of information. In order to identify contents in original languages and corresponding translations into other languages, a strict translation control must exist. Data integrity to be achieved, translation interfaces for control applications must be available. These specifications yield characteristics to both the database model and to program structures since both are language dependent.

2.2 Database Models & Systems

Databases and database technology are having a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, engineering, medicine, law, education, and library science, to name a few.

Several criteria are normally used to classify DBMSs, one of which is the data model on which the DBMS is based. The two types of data models used in many current commercial DBMSs are the **relational data model** and the **object data model**.

Conceptual modeling is an important phase in designing a successful database application. Generally, the term database application refers to a particular database and the associated programs that implement the database queries and updates. This chapter concentrates on the database structures and constraints during the database design. Section 2.2.1 presents the modeling concepts of the Entity-Relationship (ER) Model, which is a popular high-level conceptual data model while defining key database terms. In Section 2.2.2 and Section 2.2.3, Relational and Object-Oriented database systems are introduced. Section 2.2.4 defines criteria for deciding a Relational DBMS for this study instead of an Object-Oriented or a Hierarchical DBMS.

2.2.1 Data Modeling Using the Entity-Relationship Model

The ER Model and its variations are frequently used for the conceptual design of database applications. The ER model describes data as entities, relationships, and attributes.

2.2.1.1 Entities and Attributes

An *entity* is the basic object that the ER model represents. It is a "thing" in the real world with an independent existence. An entity may be an object with a physical existence – a hotel, or a room- or it may be an object with a conceptual existence – a reservation, or an activity.

Each entity has *attributes* – the particular properties that describe it. A particular entity has a *value* for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database. Several types of attributes occur in the ER model: *simple* versus *composite*; *single-valued* versus *multi-valued*; and *stored* versus *derived*.

Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meaning. Attributes that are not divisible are called **simple** or **atomic** attributes. For example, a hotel phone number can be divided into country code, area code and phone number. The value of composite attribute Phone is the concatenation of the values of its constituent simple attributes.

Most attributes have a single value for a particular entity; such attributes are called **single-valued**. For example, Address is a single-valued attribute of a hotel. In some cases, an attribute can have a set of values for the same entity.

In some cases, two (or more) attributes values are related –for example, the Age and BirthDate attributes of a Customer. For a particular Customer entity, the value of Age can be determined from the current date and the value of that customer's BirthDate. The Age attribute is hence called a **derived attribute** and is said to be derivable from the BirthDate attribute, which is called a stored attribute.

In some cases a particular entity may not have an applicable value for an attribute. For such cases, a special value called null is created. The meaning of the former type of null is *not applicable*, whereas the meaning of the latter is *unknown*.

2.2.1.2 Entity Types, Entity Sets, Keys and Value Sets

A database usually contains groups of entities that are similar. An *entity type* defines a collection of entities that have the same attributes. The collection of all entities of a particular entity type in the database at any point in time is called an *entity set*. An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes. An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely. For example, Tax Identification Number or TR Identification Number would be a key of the Customer entity type, as no two individuals are given the same Tax ID or TR ID number. Sometimes, several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity. If a set of attributes possesses this property, then a composite attribute that becomes the key attribute of the entity type can be defined. An entity type may also have no key, in which case it is called a *weak entity type*, which is explained in following sections.

Another important constraint on the entities is domain, that is, the set of values that may be assigned to a specific attribute for each individual entity.

2.2.1.3 Relationships, Roles and Structural Constraints

Whenever an attribute of an entity type refers to another entity type, there exists some relationship. A relationship is an association among two or more entities. A relation type among n entity types defines a set of associations or a relationship set among entities from these types. For example, we may have the relation that Hotel Blue has suite rooms. A relationship can also have descriptive attributes that are used to record information about the relationship, rather than about any one of the participating entities; for example, we may wish to record that Hotel Blue has 10 suite rooms. A relationship must be uniquely identified by the participating entities, without reference to the **descriptive** attributes. As another example of an ER diagram, suppose that each hotel has different types of rooms and we want to record available room quota for each room type. This relationship is **ternary** because we must record an association between a room type, a hotel, and quota.

2.2.1.3.1 Relationship Constraints

Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set. These constraints are determined from the mini-world situation that the relationships represent. Relationship constraints can be distinguished in two main types: *cardinality ratio* and *participation*.

The **cardinality ratio** for a binary relationship specifies the number of relationship instances that an entity can participate in. For example, in the HotelRoom binary relationship type, a hotel can have many rooms, while a specific room can belong to at most one hotel, meaning that it is of cardinality 1:N.

The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints – total and partial. Since it is not expected for a hotel to have every kind of RoomType, the participation of the entity set RoomType in the relationship set HotelRoom (Hotel:RoomType) is said to be partial. On the other hand, if the database policy states that each hotel has to have rooms of each type, then the participation of entity set RoomType would be total.

The underlying key constraint concept can be extended to relationship sets involving three or more entity sets. If an entity E has a key constraint in a relationship set R, each entity in an instance of E appears in at most one relationship in (a corresponding instance of) R.

2.2.1.3.2 Weak Entities

Entity types that do not have key attributes of their own are called weak entity types. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type –called identifying or owner entity type- in combination with some of their attribute values. A weak entity type always has a total participation constraint (existence dependency) with respect to its identifying relationship, because a weak entity cannot be identified without an owner entity. For example a HotelRoom entity cannot exist unless it is related to a Hotel entity.

2.2.1.3.3 Aggregation

Aggregation allows indicating that a relationship set participates in another relationship set. It is an abstraction concept for building composite objects from their component objects. There are three cases where this concept can be related to ER model. The first case is the situation where we aggregate attribute values of an object to form the whole object. The second case is when an aggregation relationship is represented as an ordinary relationship. The third case, which the ER model does not provide for explicitly, involves the possibility of combining objects that are related by a particular relationship instance into a higher-level aggregate object.

2.2.2 Relational Database Models and Systems

The relational data model was proposed by Codd in 1970. At that time, most database systems were based on one of two older data models – the hierarchical model and the network model. The relational is very simple and elegant: a database is a collection of one or more relations, where each relation is a table with rows and columns. The major advantage of the relational model over the older data models are its simple data representation and the ease with which even complex queries can be expressed. After the introduction of the relational model, there was a flurry of experimentation with relational ideas. A major research and development effort was initiated at IBM leading to the announcement of two commercial relational DBMS products by IBM in the 1980s: SQL/DS for DOS/VSE and for VM/CMS (virtual machine/conversational monitoring system) environment (1981) and DB2 for the MVS (1983).Another relational DBMS, INGRES, was developed at the University of California, Berkeley, in early 1970s and commercialized by Relational Technology, Inc., in the late 1970s. Other popular relational DBMSs include Oracle of Oracle, Inc.; Sybase of Sybase, Inc.; RDB of Digital Equipment Corp (Compaq); INFORMIX of Informix, Inc.; and UNIFY of Unify, Inc. Besides the RDBMSs mentioned above, many implementations of the relational data model appeared on the personal computer (PC) platforms. These systems were initially single-user systems, but they have started offering client/server database architecture and became compliant with Microsoft's Open Database Connectivity (ODBC), a standard that permits the use of many front-end tools with these systems.

The main construct for representing data in the relational model is a **relation**. A relation consists of a **relation schema** and a **relation instance**. The relation instance is a table, and the relation schema describes the column heads for the table. The schema specifies the relation's name, the name of each **field** (or **attribute**, or **column**), and the **domain** of each field. A domain is referred to in a relation schema by the **domain name** and has a set of associated values.

An **instance** of a relation is a set of **tuples**, also called **records,** in which each tuple has the same number of fields as the schema. A relation instance can be thought of as a table in which each tuple is a row; and all rows have the same number of fields. A relation schema specifies the domain of each field or column in the relation instance. The **degree**, also called **arity**, of a relation is the number of fields, and the **cardinality** of a relation instance is the number of tuples in it.

An integrity constraint (IC) is a condition specified on a database schema and restricts the data that can be stored in an instance of the database. The **domain constraints** in the schema specify an important condition that we want each instance of the relation to satisfy: The values that appear in a columns must be drawn from the domain associated with that columns. Thus, the domain of a field is essentially the type of that field, in programming language terms, and restricts the values that can appear in the field. **Key constraint** is a statement that a certain minimal subset of the fields of a relation is a unique identifier. The statement has two parts; (1) two distinct tuples in a legal instance (an instance that satisfies all ICs) cannot have identical values in all the fields of a key, (2) no subset of the set of fields in a key is unique identifier for a tuple. Sometimes the information stored in a relation is linked to the information stored in another relation. The most common IC involving two relations is a **foreign key constraint**. The foreign key constraint states that the foreign key in referencing relation must match the primary key of the referenced relation.

Domain, primary key and foreign key constraints are considered to be a fundamental part of the relational data model and are given special attention in most commercial systems. Sometimes, however, it is necessary to specify more general constraints such as an age field is probably an integer but assuming an Employee table where employee information for a company is stored, the age field would be limited to a value between –for example- 18 and 60. Current relational database systems support general constraints in the form of table constraints and assertions. Table constraints are associated with a single table and checked whenever that table is modified. In contrast, assertions involve several tables and are checked whenever any of these tables is modified.

2.2.3 Object-Oriented and Extended Database Technologies

Relational database systems support a small, fixed collection of data types, which has proven adequate for traditional application domains such as administrative data processing. In many application domains, much more complex kinds of data must be handled. As the amount of data grows, the many features offered by a DBMS become increasingly attractive and, ultimately, necessary. To support such applications, a DBMS must support complex data types. Object database systems – grown with the need of more complex data types- have developed along two distinct paths:

- Object-Oriented Database systems are proposed as an alternative to relational systems and aimed at application domains where complex objects play a central role. The approach is heavily influence by object-oriented programming languages.
- Object-Relational Database Systems can be though of as an attempt to extend relational database systems with the functionality necessary to support a broader class of applications and provide a bridge between the relational and object-oriented paradigms.

Figure 2.1 illustrates how programming and database concepts have come together to provide what is now called object-oriented databases.

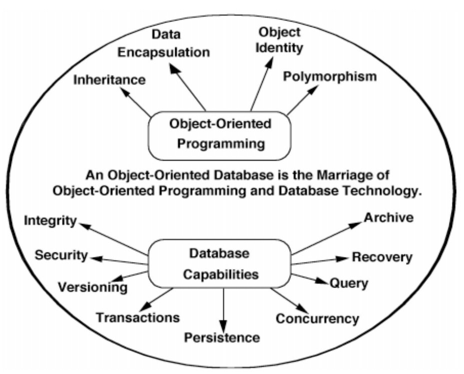


Figure 2.1 Makeup of an Object-Oriented Database

Perhaps the most significant characteristic of object-oriented database technology is that it combines object-oriented programming with database technology to provide an integrated application development system. There are many advantages to including the definition of operations with the definition of data.

First, the defined operations apply ubiquitously and are not dependent on the particular database application running at the moment. Second, the data types can be extended to support complex data such as multi-media by defining new object classes that have operations to support the new kinds of information. The basic problem confronting database designers is that they need support for considerably richer data types than is available in a relational DBMS: User-defined data types to create a specially designed and manipulated data type. User-defined abstract types are manipulated via their methods. Along with the new structured types available (i.e., array) in the data model, ORDBMS provide natural methods for those types.

Other strengths of object-oriented modeling are well known. For example, inheritance allows one to develop solutions to complex problems incrementally by defining new objects in terms of previously defined objects. Inheritance allows taking advantage of the commonality between different types that increases as the data types grow. In object database systems, unlike relational systems, inheritance is supported directly and allows type definitions to be reused and refined very easily. Object oriented databases have Object Identities to refer or 'point' to data from elsewhere in the data, which underscores the need for giving objects a unique object identity and prevents storing copies of objects. Use of reference types –called *oids*- is especially significant when the size of the object is large, either because it is a structured data type or because it is a big object such as an image or audio.

Polymorphism and dynamic binding are powerful object-oriented features that allow one to compose objects to provide solutions without having to write code that is specific to each object. All of these capabilities come together synergistically to provide significant productivity advantages to database application developers.

A significant difference between object-oriented databases and relational databases is that object-oriented databases represent relationships explicitly, supporting both navigational and associative access to information. As the complexity of interrelationships between information within the database increases, the greater the advantages of representing relationships explicitly. Another benefit of using explicit relationships is the improvement in data access performance over relational value-based relationships.

2.2.4 RDBMS versus OODBMS – Deciding the Testing Model

For Object-Oriented DBMSs, an initial area of focus has been the Computer Aided Design (CAD), Computer Aided Manufacturing (CAM) and Computer Aided Software Engineering (CASE) applications. A primary characteristic of these applications is the need to manage very complex information efficiently. All of these applications are characterized by having to manage complex, highly interrelated information, which is strength of object-oriented database systems.

However, the focus of this study is to find the implementation to manage the content and its multilingual representations. Considering the fact that content management systems do not need such complex structures, and relational model is the common practice in such content applications, this study uses the relation approach. The Relational DBMS (RDBMS) provides a relatively full set of DBMS features: atomic transactions, full concurrency support, network architecture, and support for journaling and recovery.

2.3 Multilingual Database Model

In terms of database model, specifications mentioned in the beginning of this chapter require a model that enforces accuracy of data across multiple translations. The model should also be able to tune data processing in terms of performance considering the fact that both the structure and the overall size of system data expands automatically when multiple languages are adopted by the system.

In this study, a simple online reservation system case is used to present database-modeling issues. A simple database schema for monolingual version of an online reservation system is shown in Figure 2.2.

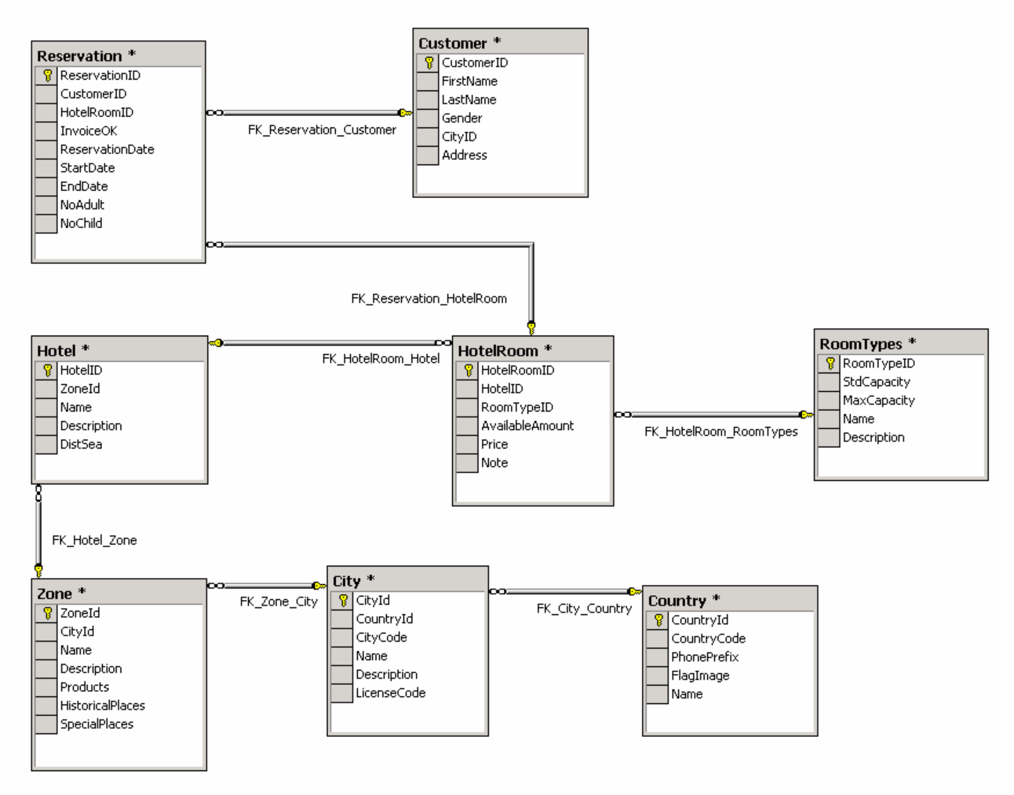


Figure 2.2 A simple Online Reservation Database Schema

The pattern described in Figure 2.3 shows in generic terms how to allow a hotel to be described in different languages while implementing multilingualism. The same pattern is used to implement presentations of various entities such as countries, cities, zones and room types.

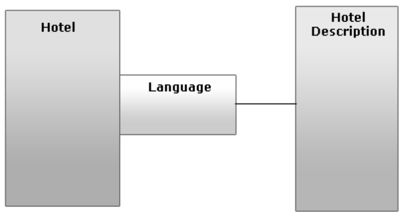


Figure 2.3 Multilingual Hotel Definition Diagram

This section first introduces the current replication model, that is, replication of records with an extra language code. Then, an alternative model is offered, the split model, in which an object (table) is split into two objects representing the monolingual and multilingual portions of the object. For the two models, advantages and disadvantages are discussed with basic comparison. Finally, load benchmarks performed on two databases are documented and the results are analyzed.

2.3.1 Replication Model

Replication model aims to keep database neutral in terms of language, by placing a predefined language code for each row in a table that identifies the language of the record. This language prefix combined with actual primary key for the objects uniquely identifies a piece of information in a specific language. Table 2.1 shows an example of how replication can be applied in an online reservation system for Hotel table.

Table 2.1 Replication applied to Hotel table

Hotel ID	Lang. ID	Zone ID	Name	Address	Description	Lat	Lon	Dist. Sea	Size	Year
1	1	27	Mavi Hotel	Poseidonos Bulvarı, Pk. 62303	Otel Paphos Limanı' na, arkeolojik alanlara ve bir çok geleneksel restoran, bar ve canlı disko olanağı sunan şehir merkezine yürüme mesafesindedir.	38° 53' 23'' N	77° 00' 27'' W	0	1000	1994
1	2	27	Blue Hotel	Poseidonos Avenue, P O Box 62303	The hotel is within walking distance of the Paphos Harbor, the archaeological sites as well as the city center which offers many traditional restaurants, bars and lively discos.	38° 53' 23'' N	77° 00' 27'' W	0	1000	1994
1	3	27	Hotel Azul	Avenida De Poseidonos, Caja De P O 62303	El hotel está dentro de la distancia que camina del puerto de Paphos, los sitios arqueológicos así como el centro de ciudad que ofrece muchos restaurantes tradicionales, barras y discos animados.	38° 53' 23'' N	77° 00' 27'' W	0	1000	1994
Integer 4 Bytes	Integer 4 Bytes	Integer 4 Bytes	Variable Char (100) 100 Bytes	Variable Char (250) 250 Bytes	Variable Char (500) 500 Bytes	Char (10) 10 Bytes	Char (10) 10 Bytes	Integer 4 Bytes	Integer 4 Bytes	Small integer 2 Bytes

Using this model, each record is replicated n times, where n is the number of supported languages, and an instance of an item is identified by a unique identifier for a specific item combined with a language code. Although this model seems to succeed in implementing a multilingual database, it ignores the fact that an object can be defined by a combination of both monolingual and multilingual fields. In other words, in replication model, monolingual fields are also treated as multilingual. To some extent, this redundant copying may seem an unimportant, however, for larger databases; it is a critical factor on performance. Assuming that the field capacities are fully used for variable length fields, each row shown in table is stored in 892 Bytes, and to represent a hotel in all supported languages total storage needed for each hotel item is 2676 Bytes.

In this database model, for example, the program contains the information that hotel 1 must be displayed. After the identification of the language parameter, the information is retrieved using (hotel id, language id) pair.

2.3.2 SPLITTING - *Normalizing Multilingual Database*

As in any database application, it is crucial that the data model for a multilingual system is designed properly. A well-organized data model definitely simplifies the business logic. It is important to identify the portions of the schema affected by multilingual support. Replication method introduced in previous section treats each database object as multilingual while claiming to keep the database language independent. In such an implementation, which violates the rules of normalization, it is inevitable to keep multiple copies of the same monolingual data.

The normalization process, as first proposed by Codd (1972), takes a relation schema through a series of tests to certify whether it satisfies a certain normal form. Normalization of data is a process of analyzing the given relation schemas based on their functional dependencies and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies. In a multilingual system, it is highly important to minimize possible redundancies based on missing or inaccurate translations of data, and irregularities in data processing, which is mostly related to content authoring component of such a system.

Functional dependencies of replication hotel entity in Figure 2.4 shows the hotel relation where primary key contains 2 attributes, the non-key attributes ZoneID, Latitude, Longitude, Sea_Distance, Area_Size and Year of Establishment are functionally dependent on a part of the primary key, that is, they are dependent on HotelID.

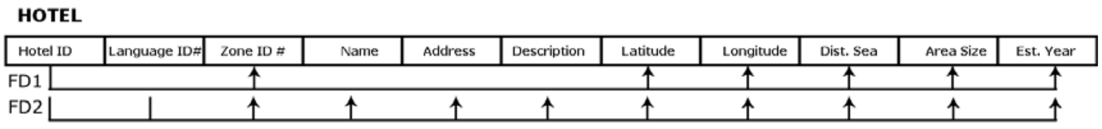


Figure 2.4 Functional Dependencies for Replicated Hotel Table

Table 2.2 and

Table 2.3 shows an example of how normalization can be applied to Hotel entity presented in previous section by decomposing and setting up a new relation for each partial key with its dependent attributes.

Table 2.2 Hotel Entity Derived by Normalization

Hotel ID	Zone ID	Lat	Lon	Dist. Sea	Size	Year
1	27	38º53' 23"N	77º00'27"W	0	1000	1994
1	27	38º53' 23"N	77º00'27"W	0	1000	1994
1	27	38º53' 23"N	77º00'27"W	0	1000	1994
Integer 4 Bytes	Integer 4 Bytes	Char (10) 10 Bytes	Char (10) 10 Bytes	Integer 4 Bytes	Integer 4 Bytes	Small integer 2 Bytes

Table 2.3 Hotel_Language Entity Derived by Normalization

Hotel ID	Lang. ID	Name	Address	Description
1	1	Mavi Hotel	Poseidonos Bulvarı, Pk. 62303	Otel Paphos Limanı' na, arkeolojik alanlara ve bir çok geleneksel restoran, bar ve canlı disko olanağı sunan şehir merkezine yürüme mesafesindedir.
1	2	Blue Hotel	Poseidonos Avenue, P O Box 62303	The hotel is within walking distance of the Paphos Harbor, the archaeological sites as well as the city center which offers many traditional restaurants, bars and lively discos.
1	3	Hotel Azul	Avenida De Poseidonos, Caja De P O 62303	El hotel está dentro de la distancia que camina del puerto de Paphos, los sitios arqueológicos así como el centro de ciudad que ofrece muchos restaurantes tradicionales, barras y discos animados.
Integer 4 Bytes	Integer 4 Bytes	Variable Char (100) 100 Bytes	Variable Char (250) 250 Bytes	Variable Char (500) 500 Bytes

Using this model, attributes identified as monolingual are placed in the actual database table, and for all multilingual attributes of an object a separate table is used. With this implementation, there are no records duplicated in Hotel entity and it is treated as fully monolingual. In multilingual version of Hotel entity (named

Hotel_Language) each record is replicated n times, where n is the number of supported languages, and an instance of an item is identified by a unique identifier for a specific item combined with a language code. Assuming that the field capacities are fully used for variable length fields, a hotel in one language is stored in 896 Bytes, that is 4 bytes more than replication model, however to represent a hotel in all supported languages total storage needed for each hotel item is 2612 bytes, 64 bytes less than replication model. For 1000 records, only Hotel table creates a difference of 62.5 Mbytes less storage than replication implementation.

2.4 Replication vs. Splitting

To test performance differences of replication and a further normalized model –splitting-, this study uses a simple online reservation system, which serves in 3 languages –English, Turkish and Spanish-. The system is first modeled with replication, and then the model is normalized to obtain the splitting design. Database schemas are given Chapter 3, Section 3.2.2.

As previously mentioned, normalization is essential to minimize data modification inconsistencies. To evaluate the two models in terms of data manipulation operations, using the online reservation system case, suppose that a hotel wants to increase its quota for a specific type of room. In replication model, available quota of a specific type of room for a specific hotel is stored in three rows (we are using a trilingual assumption), and to increase the quota, an update to all three rows is needed; otherwise information consistency would be corrupted. On the other hand, the same operation costs one update in splitting model. Update to any multilingual field is the same for both replication and split. For insert and delete operations, conversely, splitting model requires 1 additional row operation while replication needs three inserts or deletes.

3 Testing for Multilingual Database

Performance is a major issue in the acceptance of object-oriented and extended relational database systems. Measuring DMBS performance in a generic way is very difficult, since every application has somewhat different requirements. The most accurate measure of performance would be to run an actual application, representing the data in the manner best suited to each potential DMBS. However, as a generic measure is required, it is difficult or impossible to design an application whose performance would be representative of many different applications. So, the test was designed to be small and representative of multilingual systems.

In this study, an online reservation system is considered to explore the differences in replication and split models presented in previous chapter. Assumed system stores hotel information, defines various room types, each of which can be associated with one or more hotel many times with different properties such as quota, price and special services. The system also stores basic customer information and a reservation entity is used to record reservations made by customers. In all entities, names and descriptive fields are assumed to be multilingual where fields with numeric or list domains are accepted as monolingual.

This chapter introduces the test performed on multilingual database models. In Section 3.1, test model and basic test scenario is defined. Then in Section 3.2, testing environment is discussed and test features are presented. Section 3.3 explains the client simulation software used in this study.

3.1 Test Model

As a part of this study both the replication and splitting databases are tested with a special database stress utility developed with C#. As stated before, the test scenario is based on an online reservation system. First, a simple database is implemented with both replication and splitting methods. Next, the database stress software is used to simulate a number of clients connected to the system, each requesting various transactions to be executed. The requested operations include inserts, information updates and deletions for both customers and reservation. There are also detailed searching procedures and basic selects, which will probably be needed in actual development of such a system. As the purpose of the test is to determine the strength of the models, the rate at which procedures are executed is higher when compared to a real system, which simulates the performance of approximately 3-5 users per client modifying and retrieving data from the server.

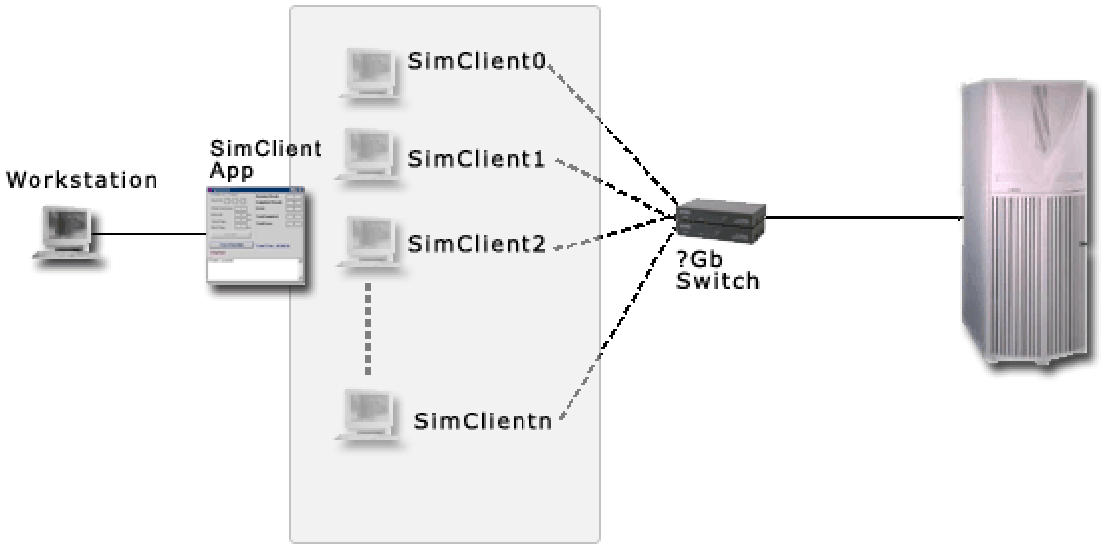


Figure 3.1 Benchmarked Configuration

3.2 Testing Environment

3.2.1 Configuration Diagrams

In order to reproduce comparable results, it is necessary to run the benchmarks and DBMS on a similar configuration. Table 3.1 and Table 3.2 shows configuration for both the database server machine and the client workstation where the remote benchmark program ran.

Table 3.1 Database Server Configuration

Database Server	
Model	
Quantity	One
Qty/Processes	One 2.40GHz/ Xeon Processor MP
Qty/Physical Memory	One 1GB
Disk Controller	Two Adaptec AIC-7902 Ultra320 SCSI Adapters
Qty/Disk Drive	One 34.22GB
LAN Connections	100Mbps Ethernet to internal network
Software	Microsoft Windows 2000 Server with SQL Server 2000

Table 3.2 Client Simulator Workstation

Database Server	
Model	
Quantity	One
Qty/Processes	One 2.40GHz/ Pentium Processor
Qty/Physical Memory	One 256MB
Qty/Disk Drive	One 34.22GB
LAN Connections	100Mbps Ethernet to internal network
Software	Microsoft Windows 2000

3.2.2 Database Table Definitions

Figure 3.2 shows database schema used for testing replication model.

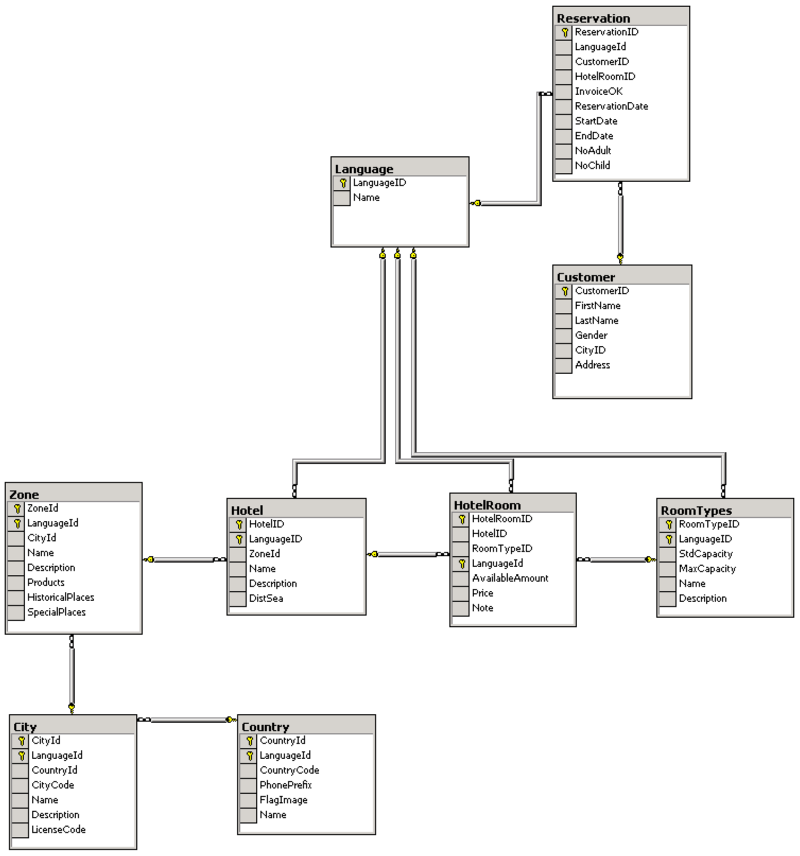


Figure 3.2 Replication Database Schema

Figure 3.3 shows database schema used for testing split model.

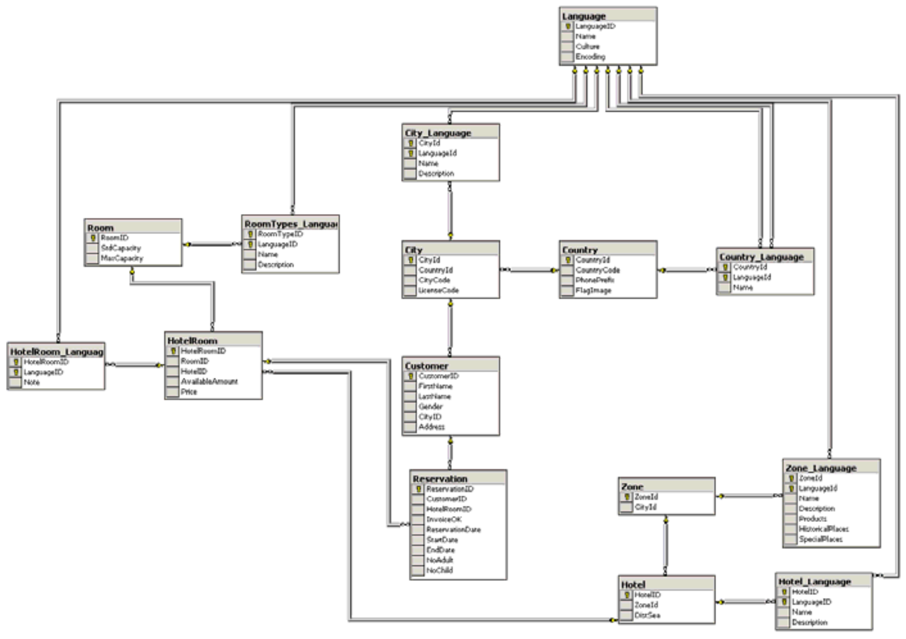


Figure 3.3 Split Database Schema

3.2.3 Cardinality of Tables

The data sets used in testing are configured to allow comparisons across two different models, besides, prior to each test, the data is restored and statistics are updated. The following table contains defined tables and the number of rows for each table used in the multilingual database benchmark as they were initially populated. The numbers given are in actual monolingual count, i.e. there are 8 countries, 4 room types, and 5007 hotels. Replication and splitting creates different row counts, which is the focus of this test.

Table 3.3 Database Table Cardinality after Initial Population

Table Name	# of Rows
Country	8
City	109
Zone	600
Hotel	5007
RoomType	4
HotelRoom	3688
Customer	11447
Reservation	6599

3.2.4 Test Procedures

The stored procedure executed by client threads during the test includes a variety of both simple and complex queries that may be used in an actual reservation system. The procedure includes simple selects for country selection, city and zone selections, and language selections. These statements are executed in a logical order so that the procedure itself is similar to the actual usage of a reservation system. For example, the procedure first selects the navigation language. Then, a country is selected and city records are filtered according to the country identifier. The procedure then creates random keywords according to which the zones are filtered along with the previously set city identifier. One other scenario uses a detailed search case where the user inputs some keywords for room features (such as TV, air-condition, cable TV, etc.), a price range, the maximum distance of hotel from the sea, and matching rooms along with the hotel and zone data is retrieved. Then, a room is randomly picked from the resulting set and a reservation is made. Next, the previously inserted reservation is updated (e.g. day extension).

The test procedure also calls batch update procedures such as increasing quota of a specific room type for all hotels. Each test procedure execution results in a new customer registration, a new reservation, reservation cancellation and customer removal. Table 3.4 shows a list of procedures used in the test scenario and their definitions.

Table 3.4 Stored procedures and their definitions used in database test

Stored Procedure	**Definition**
dbo.Countries	Retrieves country list
bbo.Cities	Retrieves city list filtered by a random country
dbo.Zones	Retrieves zone list filtered by a random city

Dbo.NewReservation	Inserts a new record in to reservation table with randomly generated field values
Dbo.CancelReservation	Selects a random reservation, deletes it and re-inserts the reservation using dbo.NewReservation
Dbo.NewCustomer	Inserts a new record in to Customer table with randomly generated field values
Dbo.GeneralSearch	Includes simple filtering queries such as zone filtering according to country, hotel filtering by name or country with randomly generated criteria
Dbo.DetailedSearch	Sets price interval, display language, room description and hotel distance (from sea) criteria and executes a detailed search query. Randomly selects a hotel room record from the result set and performs a reservation for a randomly selected customer
Dbo.UpdateCustomer	Generates a random criteria string and updates a randomly selected customer's address
Dbo.UpdateQuota	Selects a random hotel room and decreases the quota by %30
Dbo.UpdateQuota1	Selects a random hotel room and increases the quota by %5
Dbo.UpdateReservation	Selects a random reservation record and modifies the reservation with a random operation e.g. change number of child, day extension

3.3 Testing Software

The tuning phase of any data access application development implies the analysis of the database response time, possibly followed by a refinement of some database design aspects (i.e. reviewing relations and some poorly coded stored procedures or the adopted indexing strategies). In such an optimization, high load scenarios cannot be ignored, so it's a good practice to test the application under "stressed" conditions.

When designing a system that provides information for many cultures, it is important to measure the pure database response time for this complex data. This is true especially during the development and testing of applications where you want to measure the database response time to evaluate the database performance excluding the influences of upper software layers.

This study uses a simple database stress utility called ***SimClient*** that tries to address performance problems, focusing on the database performance analysis and keeping away any other application layers (data access, business logic, user interface components) that might add noise to the database response time measurement.

SimClient is coded as a Windows Forms C#. NET application and it is designed to work on an SQL Server 2000 database using the managed provider classes of System.Data.SqlClient namespace.

SimClient simulates multiple database users submitting a T-SQL script that executes a stored procedure designed for testing purposes. When you run SimClient, the software asks for some execution options and if exists, it uses the default configuration file to obtain these settings. The following properties can be configured in SimClient:

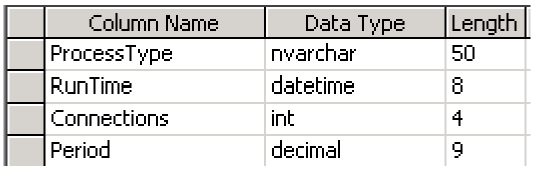
- Number of clients to simulate (that is, the number of simultaneous threads to create),
- Starting time (in order to provide the testing intervals, each application instance is given a specific start time)
- The time between the start of the activity of a user and of the subsequent user (this shift value is used to make sure that no process starves due to simultaneous requests, each execution is slightly shifted by a small amount of time while making sure that the total shifting time does not dominate total testing duration),
- The interval between each command executed by each single simulated user (Total time a client has to execute the test procedure once).

When you start the stress test, a number of concurrent threads are created and each of these begins the execution of the TestDB stored procedure on the database server.

The TestDB procedures executes many T-SQL statements including basic selects, insert updates and other stored procedures such as New Customer, New Reservation, Cancel Reservation, Detailed Search, etc. To assure that data does not change the test procedure uses complementary statements, i.e. if an insert is executed, the inserted records is deleted in the next execution. Each client uses the same connection string: so, keeping into account the effects of database connection pooling, SimClient does the following to provide pure execution data:

- A client does not start executing the test procedure until all other clients gets a connection from the pool
- Test data is generated and stored on the database server instead of sending data on an execution back to the owner thread, so the possible network effect and the communication time between the database server and the client machine is excluded

Before and after execution of each data manipulation command (insert, update, delete and select statements), execution time in milliseconds is measured and the result is stored in a test data table in master database along with process time, process type, execution time and number of connected simulation users at the time of execution for an easy post-processing of collected data. Figure 3.4 shows the attributes of table where test data is stored.



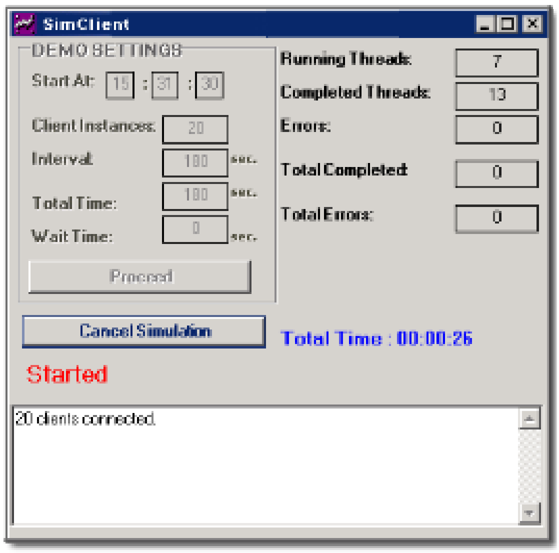
Column Name	Data Type	Length
ProcessType	nvarchar	50
RunTime	datetime	8
Connections	int	4
Period	decimal	9

Figure 3.4 Structure of Collected Data

During the test, SimClient displays a status report including;

- The number of currently running client instances (incomplete)
- The number of the total executed commands (successful completions)
- The number of the total errors (instances failed to execute the procedure)
- Last error (details on the last error occurred during simulation)

Figure 3.5 shows the running SimClient application.



SimClient

DEMO SETTINGS

Start At:	15 : 31 : 30	
Client Instances:	20	
Interval:	180	sec.
Total Time:	180	sec.
Wait Time:	0	sec.

Running Threads:	7
Completed Threads:	13
Errors:	0
Total Completed:	0
Total Errors:	0

Total Time : 00:00:26

Started

```
20 clients connected.
```

Figure 3.5 SimClient Application Snapshot

A test run has to be considered over when the given interval is complete even if there is instances that have not completed its execution yet. This may create a difference in the number of data collected; however, by giving enough amount of time as test interval, this difference is avoided.

3.4 Testing Plan

3.4.1 Primary Issues

No data may be cached at the time the benchmark is started. The database must be closed and reopened before each benchmark measure, to empty cache. This is a subtle point that can make a major difference in the results. Thus, in the general order of executing a specific benchmark, first the caches are dropped to clear all related pages out of buffers.

As the remote results are a more realistic model of related systems, the data is located on a remote database server. The DBMS is implemented using client-server architecture, i.e. every database call from the application must go over a network to the DBMS server where data is stored. This is very inefficient for an application or benchmark as there is no way to cache data locally. However, a stored procedure was also defined for each of the benchmark measures, so the client/server interaction consisted of a single customized call per user thread.

3.4.2 Measurement Intervals

Tables below contain the duration, start time and stop time of each Measurement Interval reported from the same run. Each Measurement Interval is delayed to demonstrate intervals are non-overlapping.

Table 3.5 shows intervals for test performed on replication database.

Table 3.5 Measurement Interval Start-End Times and Duration for Replication Test

REPLICATION			
Connections	**Start Time**	**End Time**	**Duration**
1	10:07:28	10:08:12	0:00:44
50	10:11:13	10:18:08	0:06:55
100	10:31:26	10:43:26	0:12:00
200	11:07:22	11:31:39	0:24:17
300	12:14:46	12:48:53	0:34:07
500	13:48:51	14:54:56	1:06:05

Table 3.6 shows intervals for split database test.

Table 3.6 Measurement Interval Start-End Times and Duration for Split Test

SPLIT			
Connections	**Start Time**	**End Time**	**Duration**
1	10:13:40	10:14:24	0:00:44
50	10:25:35	10:32:30	0:06:55
100	10:55:16	11:07:16	0:12:00
200	11:44:39	12:08:56	0:24:17
300	13:03:39	13:37:46	0:34:07
500	15:03:21	16:09:26	1:05:25

3.4.3 Database Interaction Percentages

Table 3.7 shows the percentage of each Database interaction executed during each measured interval.

Table 3.7 Database Interaction Percentages

Interactions	**%**
Select	75,60
Insert	10,90
Update	13,50

Test measures included inserting, modifying and looking up objects. As table shows, the number of data retrieval commands is much higher than insertions and modifications. The test procedure is designed so that the measures are executed approximately in proportion to their frequency of occurrence in representative applications (more reads than writes).

4 RESULTS & DISCUSSIONS

As stated before, this study focuses on offering a method for better database design in multilingual applications. This chapter presents the results obtained from the tests performed on two different multilingual database models, where in the first model – replication- each tuple that represent a unique item is replicated n times where n is the number of languages supported. The second model was obtained by applying further normalization to the replication model in order to increase the quality of the database design. In Section 4.1, database table cardinality for both models is given. Section 0 compares the complexity of queries each model requires. Average and total response time data collected during the tests grouped by database operation types are listed in Section 4.2, and in Section 4.3, reproducibility of the test results is presented. After evaluating the models both according to the test results and database design concepts in Section 4.4, a complete model for a multilingual system is offered in Section 4.5.

4.1 Cardinality of Database Tables

Table cardinalities given in Section 3.2.3 are monolingual count of rows, that is, each number show the actual number of *objects* represented in the database independent of language. Table 4.1 shows the actual number of tuples needed in replication to represent the number of objects given in Section 3.2.3.

Table 4.1 Database Table Cardinality in Replication

Table Name	Tuples
Country	24
City	327
Zone	1800
Hotel	15021
RoomType	12
HotelRoom	11064
Customer	11447
Reservation	6599
Total	**46296**

Table 4.2 show how these object counts are affected by the split model implemented. As shown in table, split model requires 9414 extra tuples for the same data, however, it is not would not be objective to jump to conclusion that replication of data is better just because it creates less rows for exactly same data. The results are discussed in Section 4.4.

Table 4.2 Database Table Cardinality in Split

Table Name	Tuples
Country	8
Country_Language	24
City	109
City_Language	327
Zone	600
Zone_Language	1800
Hotel	5007
Hotel_Language	15021
RoomType	4
RoomType_Language	12
HotelRoom	3688
HotelRoom_Language	11064
Customer	11447
Reservation	6599
Total	**55710**

4.2 Database Interaction Response Times

The minimum, maximum, average and 90th percentile response times are given in tables below for each database interaction and measurement interval. Response time measured by the test is the real time elapsed from the point where the a particular query statement in test procedure is called, until the results of the query, if any, have been placed into the procedure's variable. The k^{th} percentile is the number which has k% of the values below it. 90^{th} percentile is included in results below to show the value which has 90% of the values below it. Table 4.3 shows minimum, maximum, average and 90^{th} percentile response times measured for retrievals using both replication and split models.

Table 4.3 Minimum, Maximum, Average and 90th Percentile Response Times for RETRIEVALS

SELECT	REPLICATION				SPLIT			
Connections	Minimum	Maximum	Average	90th	Minimum	Maximum	Average	90th
1	0,0	970,0	109,1	250,0	0,0	860,0	102,0	340,4
50	0,0	25576,0	1567,0	3430,7	0,0	32076,0	1734,0	4769,0
100	0,0	46360,0	2752,0	6375,1	0,0	68403,0	3001,0	8784,0
200	0,0	92390,0	4665,3	10589,0	0,0	159593,0	4800,0	14656,0
300	0,0	175343,0	7628,6	15875,4	0,0	238110,0	7539,0	21830,0
500	0,0	289920,0	11207,4	22906,0	0,0	283343,0	11178,0	31341,7

Table 4.4 shows minimum, maximum, average and 90th percentile response times measured for insertions in both replication and split models.

Table 4.4 Minimum, Maximum, Average and 90th Percentile Response Times for INSERTIONS

INSERT	REPLICATION				SPLIT			
Connections	Minimum	Maximum	Average	90th	Minimum	Maximum	Average	90th
1	0,0	33,0	15,1	19,4	13,0	50,0	18,0	22,8
50	0,0	486,0	28,9	50,0	0,0	233,0	27,0	46,0
100	0,0	1250,0	49,7	92,1	0,0	1236,0	30,0	60,0
200	0,0	1983,0	96,8	234,8	0,0	1280,0	57,0	110,0
300	0,0	2360,0	130,2	280,0	0,0	1716,0	78,0	186,0
500	0,0	3263,0	151,5	433,4	0,0	2233,0	85,0	263,9

Table 4.5 shows minimum, maximum, average and 90th percentile response times measured for update queries in replication and split models.

Table 4.5 Minimum, Maximum, Average and 90th Percentile Response Times for UPDATES

UPDATE	REPLICATION				SPLIT			
Connections	Minimum	Maximum	Average	90th	Minimum	Maximum	Average	90th
1	0,0	33,0	15,3	30,0	0,0	16,0	7,0	16,0
50	0,0	656,0	59,5	127,4	0,0	390,0	22,0	46,0
100	0,0	3453,0	209,2	442,0	0,0	1173,0	39,0	76,0
200	0,0	3440,0	241,1	682,1	0,0	2543,0	107,0	250,0
300	0,0	5813,0	388,6	1250,6	0,0	3516,0	206,0	623,3
500	0,0	6890,0	556,6	1830,0	0,0	5326,0	246,0	610,0

In Table 4.6, average response times for select queries and difference between two models in percent is given.

Table 4.6 Response Times and % Difference for RETRIEVALS

SELECT	REPLICATION	SPLIT	
Connections	**Average**	**Average**	**% Difference**
1	109,1	102,0	-6,49%
50	1567,0	1734,0	10,66%
100	2752,0	3001,0	9,05%
200	4665,3	4800,0	2,89%
300	7628,6	7539,0	-1,18%
500	11207,4	11178,0	-0,26%

Table 4.7 shows average response times and difference between two models in percent for insert operations.

Table 4.7 Response Times and % Difference for INSERTIONS

INSERT	REPLICATION	SPLIT	
Connections	**Average**	**Average**	**% Difference**
1	15,1	18,0	19,12%
50	28,9	27,0	-6,72%
100	49,7	30,0	-39,66%
200	96,8	57,0	-41,14%
300	130,2	78,0	-40,08%
500	151,5	85,0	-43,90%

Table 4.8 shows average response times and difference between two models in percent for updates.

Table 4.8 Average Response Times and % Difference for UPDATES

UPDATE	REPLICATION	SPLIT	
Connections	**Average**	**Average**	**% Difference**
1	15,3	7,0	-54,10%
50	59,5	22,0	-63,02%
100	209,2	39,0	-81,36%
200	241,1	107,0	-55,63%
300	388,6	206,0	-46,98%
500	556,6	246,0	-55,80%

Table 4.9 shows total response times (sum of select, insert and updates) and difference between two models in percent.

Table 4.9 Total Response Times

Connections	REPLICATION	SPLIT	Difference %
1	0,1395	0,127	-9,84%
50	1,6554	1,783	7,16%
100	3,0109	3,07	1,93%
200	5,0032	4,964	-0,79%
300	8,1474	7,823	-4,15%
500	11,9155	11,509	-3,53%

4.3 Reproducibility of the Measurement Results

Reproducibility is the variation in average measurements obtained in two or more test runs using the same technique under same conditions. Figures Figure 4.1 and Figure 4.2 show the results obtained in 3 different runs.

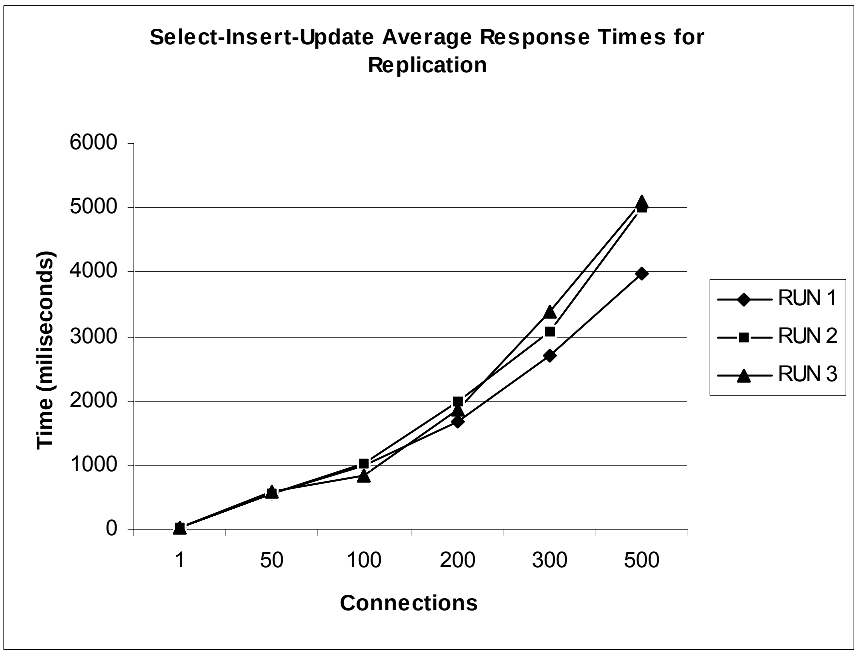


Figure 4.1 Average Response Times of 3 Runs for Replication Database

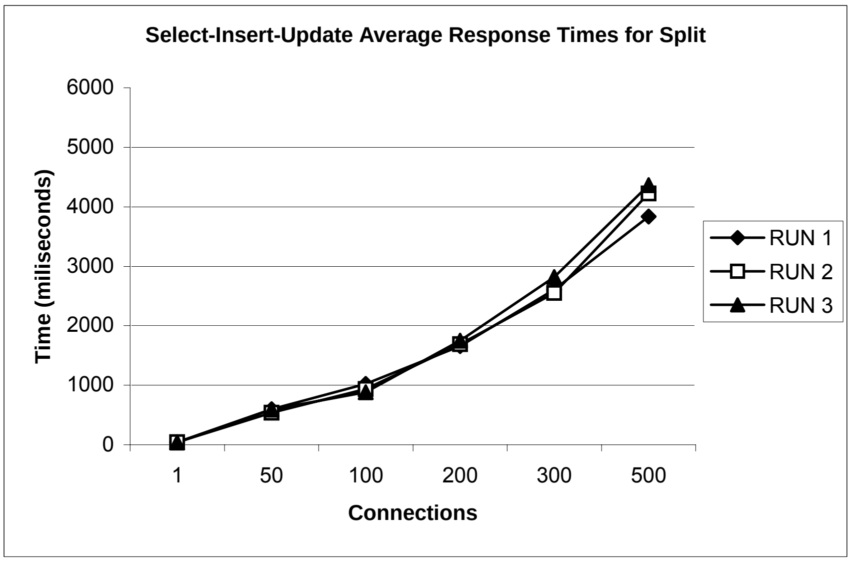


Figure 4.2 Average Response Times of 3 Runs for Split Database

4.4 Interpreting the Test Results

Before expecting to get a good performance out of a system, it is essential to make sure that both the logical and the physical designs of the database layer are right. Otherwise, once the application development is started, it might be too late to fix database design problems after the application has been implemented. In that case, no amount of fast, expensive hardware can fix the poor performance caused by poor logical database design.

Cardinalities listed in Section 4.1 shows that replication requires less tuples; however, this does not always mean that there is less data. By normalizing the replication model into split model, the total amount of redundant data based on repeated language-independent data in the database is extremely reduced. The less data there is, the less work the system has to perform, speeding its performance. Splitting the object attributes into monolingual and multilingual tables helps to reduce the number of columns in tables, which means that more rows can fit on a single data page, which helps to boost database server read performance. Splitting also helps to maximize the use of clustered indexes, which are the most powerful and useful type of index. The more data is separated into multiple tables because of normalization, the more clustered indexes become available to help speed up data access. By reducing the number of columns in tables, multiple indexes to retrieve it is less needed. In addition, reducing the total number of indexes reduces the negative effect of INSERTS, UPDATES and DELETES on performance.

When normalized design forces to create queries with many multiple joins, denormalizing some of the tables is considered in order to reduce the number of required joins. *Denormalization* is the process of selectively taking normalized tables and re-combining the data in them in order to reduce the number of joins needed them to produce the necessary query results. Sometimes the addition of a single column of redundant data to a table from another table can reduce a 4-way join into a 2-way join, significantly boosting performance by reducing the time it takes to perform the join. While denormalization can boost join performance, it can also have negative effects. For example, adding redundant data to tables risks the following problems:

- Increased amount of data means more data pages has to be read than otherwise needed which decreases performance.
- Redundant data can lead to data anomalies and bad data.
- In many cases, extra procedures will have to be coded to keep data in synch, which adds to database overhead.

However, before considering whether to denormalize tables to speed joins, it should be first assured that the proper indexes are available on the tables to be joined as it is possible, in some cases, that join performance problem is related to a lack of appropriate indexes instead of several table joins.

Figure 4.3 and Figure 4.4 shows graphs of average retrieval and modification times measured in multilingual database models test. Figure 4.3 proves that the retrievals are not that affected by the increased number of joins mentioned previously. This is probably due to the simplicity of database model used to evaluate the effects of normalization for multilingualism. As the only columns to be indexed are the primary and foreign keys, the figure below actually confirms that the increased number of joins is negated by the fact that replication model is larger in amount of stored data because of unnecessarily duplicated data and replicated tables have more columns which reduces the read performance.

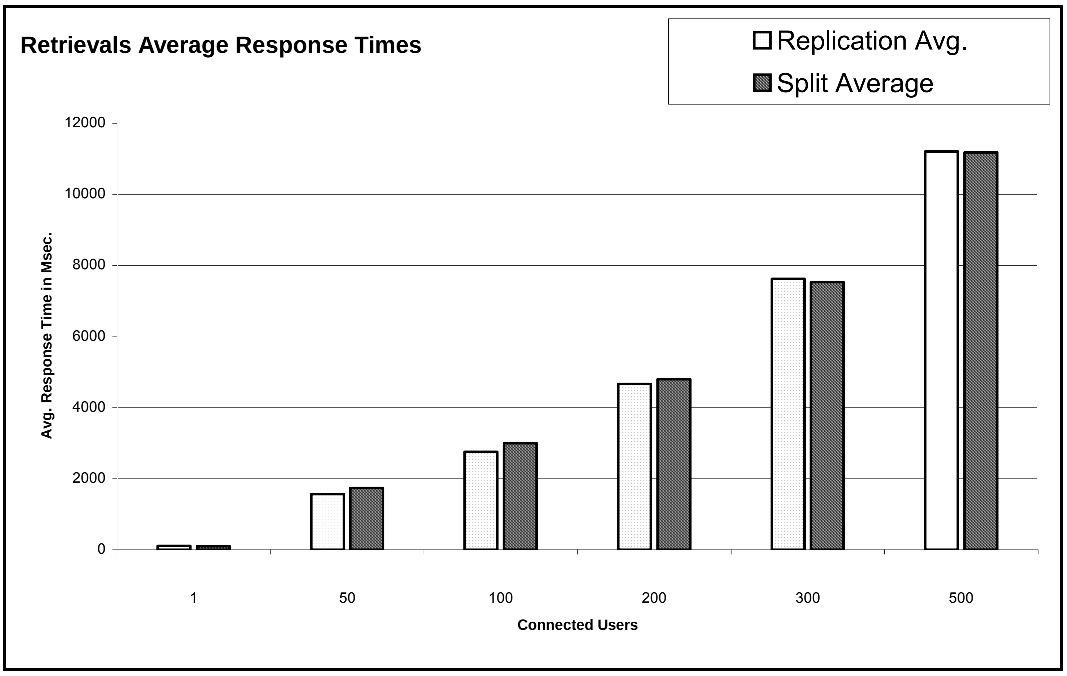


Figure 4.3 Retrievals Average Response Times

In Figure 4.4, the performance effect of normalization on modification operations (inserts, updates and deletes) is obvious. As can be seen from the figure, after 100 users (in this test case this number can be perceived as 300 users as test procedures simulates the performance of almost 3 users), replication model shows a peak and the response times dreadfully increase as the number of users increase, while more normalized split model show a stable and reasonable augmentation.

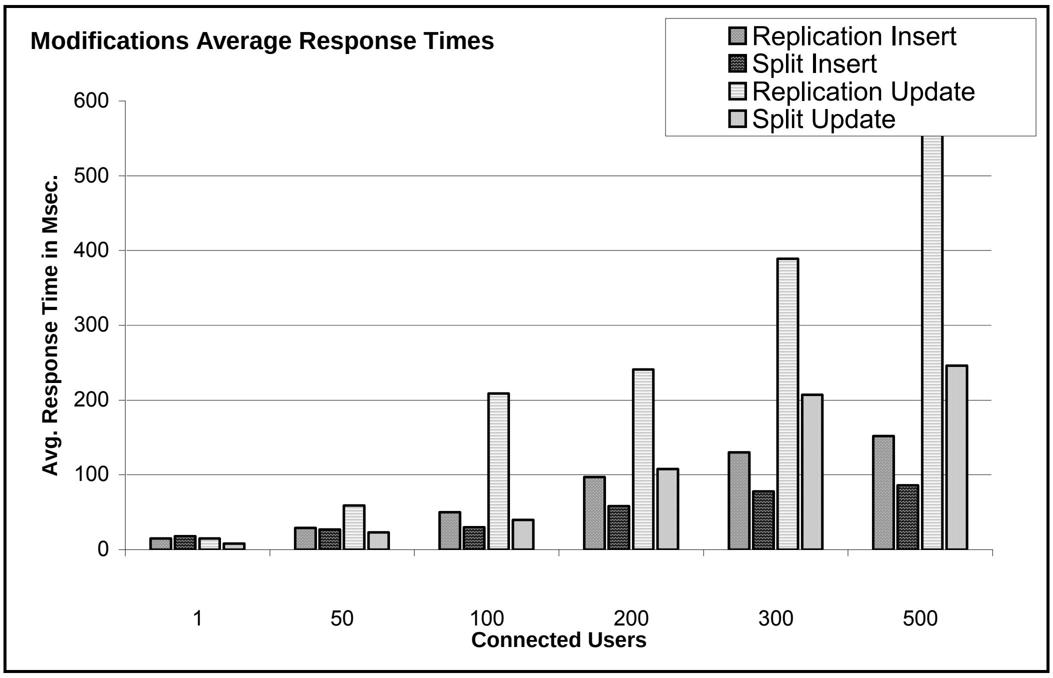


Figure 4.4 Insertions and Updates Average Response Times

This is the distinct indicator of fewer indexes on split database tables creates an evident response time difference. The resulting response times are also effected by the fact that, in some statements, a monolingual field update means 1 record update for split database whereas in replication such an update results in 3 (the number of languages assumed in this study) record updates.

Considering the database performance issues introduced early in this section and the outcome of the database load tests, it is possible to construct some criteria on implementing split model.

As discussed before, normalization may lead to excessive table joins, which adds extra time to perform table joins. As it is the primary focus of this study to find the best database implementation for multilingual content, denormalization should be considered at some point. Although the results obtained shows splitting improves the database performance, it still can be enhanced by taking the amount of multilingual data a table contains as a metric for deciding whether or not to normalize the table.

Figures Figure 4.5, Figure 4.6, Figure 4.7 and Figure 4.8 show the number of total disk read/writes caused by basic select, insert, update and delete commands executed on two databases.

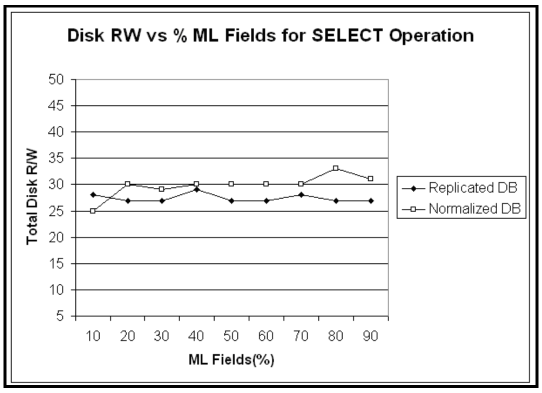


Figure 4.5 Disk RW vs. % ML Fields for SELECT Operation

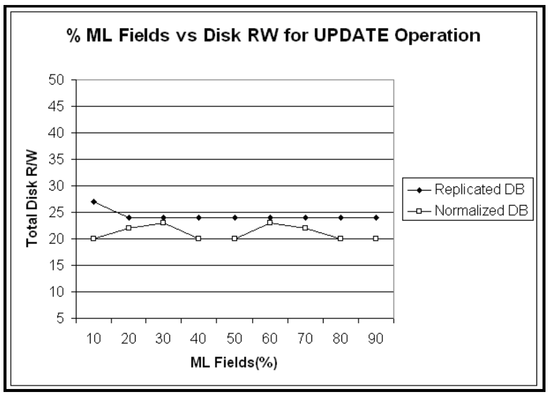


Figure 4.6 Disk RW vs. % ML Fields for UPDATE Operation

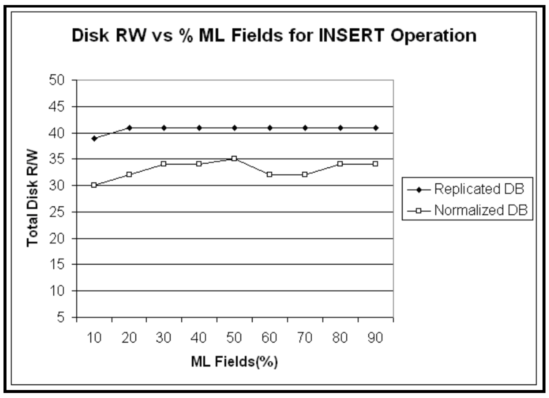


Figure 4.7 Disk RW vs. % ML Fields for INSERT Operation

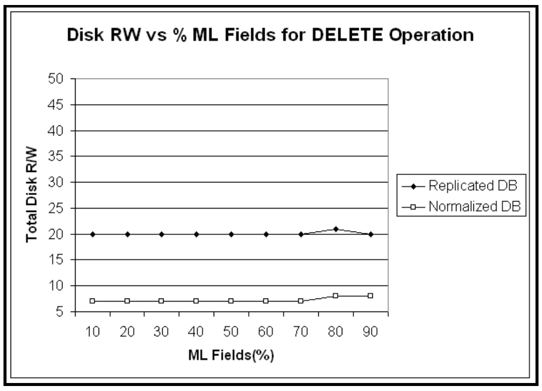


Figure 4.8 Disk RW vs. % ML Fields for DELETE Operation

Figure 4.6 shows that update commands executed on normalized database require less disk read/write than replicated database for all multilingual field percentages. Similar to update command, based on Figures Figure 4.7, Figure 4.8, it is possible to say that disk read/writes for normalized database are less than replicated database regardless of multilingual field percentage for insert and delete command.

Although there is a consistently low read/write count for modification commands, graph for select command shows that as percentage of multilingual fields increase, read/write count also tends to increase at a reasonable rate.

According to the results, it would be fair to say that if the system is expected to perform excessive data modification where there will be fewer select statements, such as online newsletter sources, travel agency system where data is stored on a central database and numerous reservations are created, modified and deleted by agencies from various places, normalizing database may provide a performance increase. However, for a system that will perform too many select operations (such as search engines, online library catalogs, etc.) and fewer inserts, updates and deletes using replication on basic database objects may speed up system processes.

There are also other criteria that should be evaluated before deciding which parts of the data should be normalized or replicated. One criterion should be the database server software and its configurations. Data page size of database software differs from one version to another as well as the software itself. As previously mentioned, normalizing a table can reduce the row length and it means that more rows can fit on a single data page which speeds up the server performance. To give an example, SQL Server 7.0 and 2000 data pages are 8K (8192 bytes) in size. Of the 8192 available bytes in each data page, only 8060 bytes are used to store a row. The rest is used for overhead. To optimize SQL Server performance, the rows should be designed in such a way as to maximize the number of rows that can fit into a single data page. The more densely rows are stored in data pages, the less I/O SQL Server has to perform when reading data pages from disk, and the more rows it can squeeze into the buffer. The more rows that are fitted into the buffer increases the likelihood that the data needed is in the buffer and not on the disk, saving even more valuable I/O resources [32].

For example, for a data page of 8060 bytes size, if a row is 4031 bytes long, then only one row could fit into a data page, and the remaining 4029 bytes left in the page would be empty. This is a great waste of space which can affect the I/O performance of the server. In such a situation, the row can be redesigned (normalized), if possible, so that the row is 4030 bytes or less. As a result, two rows can fit in each page and I/O performance would be greatly enhanced. This can also apply to cases where three, four, or more records should fit efficiently into a single data page. Briefly, before deciding whether to replicate or normalize the database object, initial row length may be considered to calculate number of rows in a data page in each implementation and the best fit can be used.

Next criterion is the storage capacity of the database server. As replication needs more the disk space, using replication can cause a storage bottleneck on the database server. Disk drives also have physical limitations on how many I/Os a disk is capable of handling. Depending on the workload, this limit can be reached. When the number of I/O requests exceeds the disks I/O capacity, the I/O requests will take longer and be queued for its turn on the disk. For a database server where numerous concurrent requests will be executed, either hardware upgrade or reducing disk access by normalization should be considered if the disk IO capacity is insufficient.

4.5 Further Considerations

In a typical content management system there are four basic layers; creation, storage, distribution and management layers. In the early days of computers, the creation layer only consisted of word processors. Data was stored on a file system and data distribution meant printing of those documents on paper. Workflow was managed by habits or written or unwritten rules. Over the years, as the effects of technological progress on each of these basic layers, word processors became more powerful tools, documents were managed by specific document management systems and workflow systems to track the status of documents on the system were introduced.

However, the advent of the Internet had a different effect on the distribution layer in the sense that development of Internet technologies provided a wealth of opportunities to service for a specific audience. Although it is not feasible to produce personalized documents on paper, it is quite easy to produce personalized output on the worldwide web.

Personalization –according to language and culture- has been implemented using a few basic methods. In some implementations, a directory for each language is created and the same structure and content is replicated. One other implementation uses the same directory for all languages; however, the directory contains a copy of the content (page) for each language, where the files (pages) have the language prefix in their names (e.g., en- for English, fr- for French). The third solution on the file system is to translate the filenames along with the content, and the correspondence depends on this translation. As a final point, other systems use a disorganized mixed of above options with some additional options (e.g., multilingual page numbering, etc.).

In all these cases, maintenance of different parts of the system is the responsibility of the developer that is performed manually. While the consistency depends on the reliability of the programmers, these practices are even more complicated in design, quality check and evolution phases.

In a multilingual site every page in a certain language is expected to have a corresponding page in each other language the system serves for, meaning a change in a document must be propagated to all other parallel documents. As a result, not only the site structure but also the information presented on the web site must be consistent across multiple languages. Such models require extra effort to provide the same information in the same presentation format with compliant intra and inter-language hyperlinks in different languages.

To overcome this lack of automatic support to multilingual content development, recent practices use database support instead of using replicated content files. In this method, regardless of the standard data model, the database includes additional data objects for multilingual information storage and management. Using this model, the number of objects in the data model is extended by the additional culture, language region and system dictionary objects. Simply, the database implementation treats each attribute and its value in each data object as a multilingual piece. After language and culture data is updated, a dictionary composed of all pieces is created for each language and appropriate relations are drawn between culture, language and dictionary objects, which are then used to retrieve the corresponding translation of information retrieved.

Although this technique seems to solve the multilingual content management problem, it is not possible to claim that it is the optimum solution. The database implementation lacks efficiency in the sense that it treats the entire database as dynamic and multilingual. However, the content of a particular web site is never completely dynamic and multilingual. Instead of translating every single piece of information, the targeted content should be analyzed carefully to categorize the content parts. For example, in its simplest form, a typical content may be composed of (1) mono-lingual fields such as universal conventions (e.g., numeric values, date-time values, private names, etc.), and (2) multilingual fields. Multilingual fields can also be subdivided into 2 basic classes: (a) static fields (e.g., labels, list values), and (b) dynamic fields (e.g., author inputs, documents). Considering this simple classification, storing multiple copies of mono-lingual data and storing static values on the database will add to data retrieval cycle leading to reduced system efficiency and significantly waste the storage, for an enterprise system for example, where unclassified storage can bring thousands of additional rows.

Each technique mentioned above has its own strengths in a particular aspect of multicultural content presentation. Thus, a better model can be a composition of methods above, each implemented in a part where they would work best. Foremost, pieces of information should be analyzed carefully and categorized according to their dependence on culture and frequency of updates.

A typical web-based content management system will have a user interface and information to represent and manage. User interface is culture dependent as previously mentioned, because people from the different cultures and languages will have a different behavior of using a system. The choices for colors, fonts, and graphics are defined by the purpose and the type of site as well as the cultures of the target audience. Technical considerations for user interface include average word length, reproduction, font definitions, file locations, and embedded text. The user must be presented with clear navigation methods. Ideally, these are content based and synchronized across languages. This method requires editorial understanding of the material. Another method is the icon approach. While the icon method does work, it relies on nonverbal communication, which may not be effective in cross-cultural settings or may not work at all for more complex and abstract ideas.

Figures Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13 show how user interface design changes for The Coca Cola Company® web site according to country and culture.

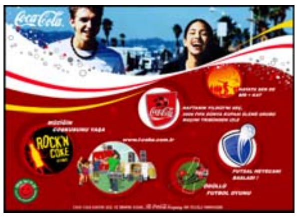


Figure 4.9 The Coca Cola Company (Turkey)

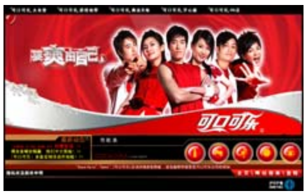


Figure 4.10 The Coca Cola Company (China)

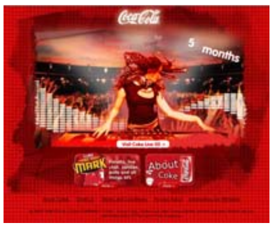


Figure 4.11 The Coca Cola Company (Australia)

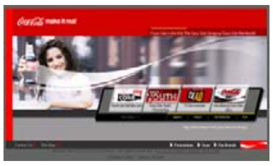


Figure 4.12 The Coca Cola Company (USA)

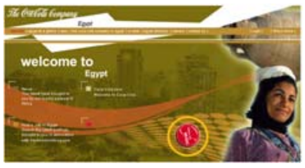
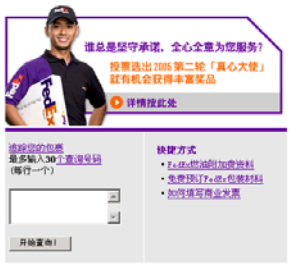


Figure 4.13 The Coca Cola Company (Egypt)

Static labels are texts and images that are used to inform the system user on what information to enter in a form field, error/warning/info messages or titles and paragraphs that make up a page. These labels can either be monolingual or multilingual. Figures Figure 4.14, Figure 4.15 and Figure 4.16 show how these static labels differ in FedEx shipment tracking form from one culture to another.


谁总是坚守承诺，全心全意为您服务？

投票选出 2005 第二轮「真心大使」
就有机会获得丰富奖品

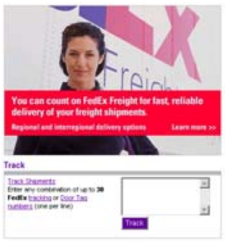
详情按此处

追踪您的包裹
最多输入30个查询号码
（每行一个）

开始查询！

快捷方式
- FedEx燃油附加费资料
- 免费预订FedEx包装材料
- 如何填写商业发票

Figure 4.14 FedEx Shipment Tracking (China)


You can count on FedEx Freight for fast, reliable delivery of your freight shipments.

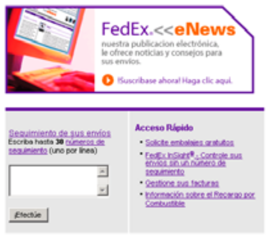
Regional and interregional delivery options　Learn more >>

Track

Track Shipments
Enter any combination of up to 30 FedEx tracking or Door Tag numbers (one per line)

Track

Figure 4.15 FedEx Shipment Tracking (USA)


FedEx.<<eNews

nuestra publicacion electrónica, le ofrece noticias y consejos para sus envios.

¡Suscribase ahora! Haga clic aquí.

Seguimiento de sus envíos

Escriba hasta 30 números de seguimiento (uno por línea)

¡Efectúe

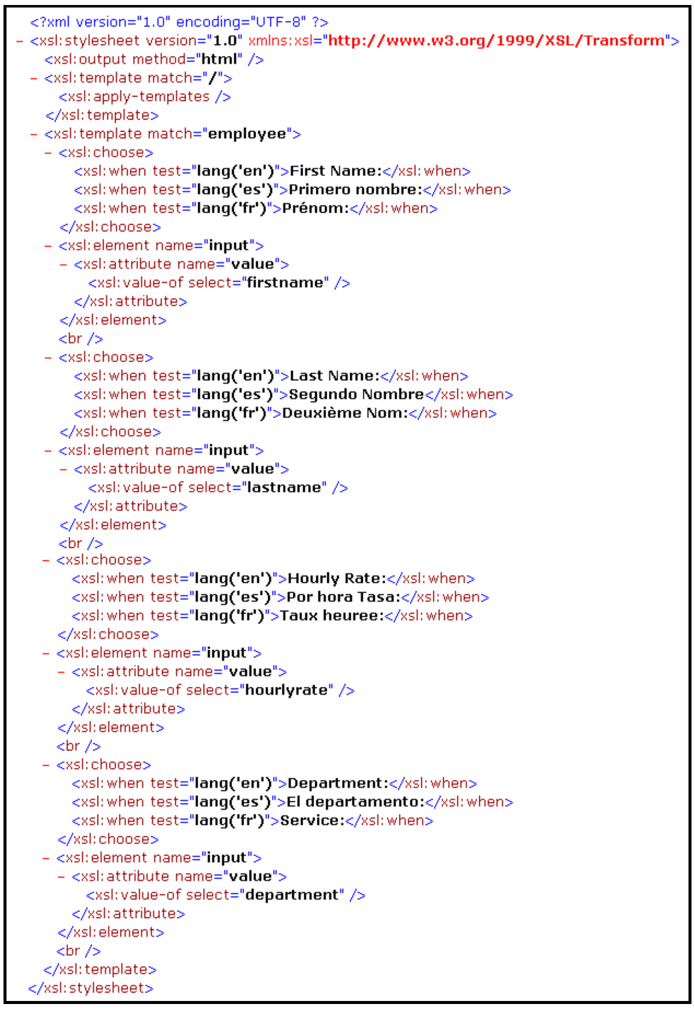
Acceso Rápido

- Solicite embalajes gratuitos
- FedEx InSight® - Controle sus envíos sin un número de seguimiento
- Gestione sus facturas
- Información sobre el Recargo por Combustible

Figure 4.16 FedEx Shipment Tracking (Spain)

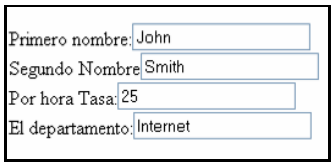
Therefore, for the general layout of the site, there should be some distinction between templates of each country or culture. This can be achieved by creating a generic page template for each culture and deciding programmatically which one to use when the site is loaded. As this is not a dynamic but culture-dependent part, it would be reasonable to store the templates contents such as images and labels in a culture specific folder on the file system. It is essential to separate content from presentation in order to provide an easy management workflow where a change in presentation does not affect the content or vice versa. XSLT is a language for transforming the structure of a data source. XSLT serves for the need to separate information (such as a weather forecast) from details of the way it is to be presented on a particular device and the need to transmit information (such as orders and invoices) from one organization to another without investing in bespoke software integration projects [33].

Figure 4.17 demonstrates use of XSL to create output for different languages. This method can be used to design page templates for different countries. By designing a style sheet for each culture, there will be no need for separate html documents. The style sheet will get the parameter for language and will decide which attributes to display on the page that calls the XSL. Sample output for Spanish is given in Figure 4.18.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" />
  - <xsl:template match="/">
      <xsl:apply-templates />
    </xsl:template>
  - <xsl:template match="employee">
    - <xsl:choose>
        <xsl:when test="lang('en')">First Name:</xsl:when>
        <xsl:when test="lang('es')">Primero nombre:</xsl:when>
        <xsl:when test="lang('fr')">Prénom:</xsl:when>
      </xsl:choose>
    - <xsl:element name="input">
      - <xsl:attribute name="value">
          <xsl:value-of select="firstname" />
        </xsl:attribute>
      </xsl:element>
      <br />
    - <xsl:choose>
        <xsl:when test="lang('en')">Last Name:</xsl:when>
        <xsl:when test="lang('es')">Segundo Nombre</xsl:when>
        <xsl:when test="lang('fr')">Deuxième Nom:</xsl:when>
      </xsl:choose>
    - <xsl:element name="input">
      - <xsl:attribute name="value">
          <xsl:value-of select="lastname" />
        </xsl:attribute>
      </xsl:element>
      <br />
    - <xsl:choose>
        <xsl:when test="lang('en')">Hourly Rate:</xsl:when>
        <xsl:when test="lang('es')">Por hora Tasa:</xsl:when>
        <xsl:when test="lang('fr')">Taux heuree:</xsl:when>
      </xsl:choose>
    - <xsl:element name="input">
      - <xsl:attribute name="value">
          <xsl:value-of select="hourlyrate" />
        </xsl:attribute>
      </xsl:element>
      <br />
    - <xsl:choose>
        <xsl:when test="lang('en')">Department:</xsl:when>
        <xsl:when test="lang('es')">El departamento:</xsl:when>
        <xsl:when test="lang('fr')">Service:</xsl:when>
      </xsl:choose>
    - <xsl:element name="input">
      - <xsl:attribute name="value">
          <xsl:value-of select="department" />
        </xsl:attribute>
      </xsl:element>
      <br />
    </xsl:template>
  </xsl:stylesheet>
```

Figure 4.17 Translating XSL file content for a simple input form



Primero nombre:	John
Segundo Nombre	Smith
Por hora Tasa:	25
El departamento:	Internet

Figure 4.18 Output of login form for 'es' (Spanish)

In this example, different elements are translated in-line using the xsl:choose element. This technique is used best for documents that require only small sections to be translated.

For a content management system, one of the most important issues is designing and managing the way each part of the content is displayed in different cultures and languages. Similar to general system layout, content presentation can be achieved by using XML and its related technologies such as XSL, XSD, etc.

To present a sample approach, data extracted from the data source can be converted to XML using a predefined data structure and then XSLT style-sheet is applied.

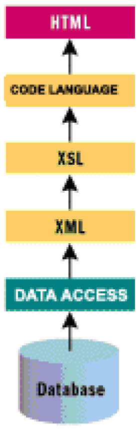
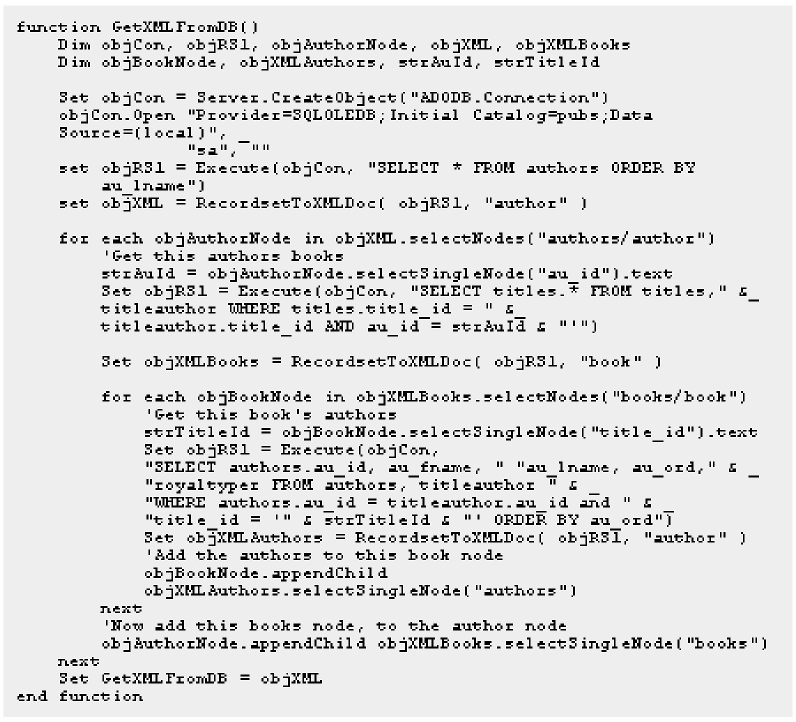


Figure 4.19 Concept diagram for XML/XSL data presentation approach

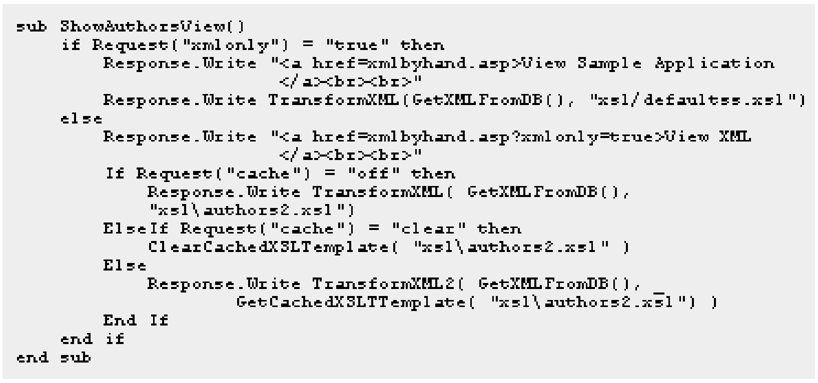
In this case, the data has been completely removed from any presentation formatting [34]. Sample code required to build the XML from the database is shown in figures below.



```
function GetXMLFromDB()
    Dim objCon, objRS1, objAuthorNode, objXML, objXMLBooks
    Dim objBookNode, objXMLAuthors, strAuId, strTitleId

    Set objCon = Server.CreateObject("ADODB.Connection")
    objCon.Open "Provider=SQLOLEDB;Initial Catalog=pubs;Data
    Source=(local)", _
                "sa", ""
    set objRS1 = Execute(objCon, "SELECT * FROM authors ORDER BY
        au_lname")
    set objXML = RecordsetToXMLDoc( objRS1, "author" )

    for each objAuthorNode in objXML.selectNodes("authors/author")
        'Get this authors books
        strAuId = objAuthorNode.selectSingleNode("au_id").text
        Set objRS1 = Execute(objCon, "SELECT titles.* FROM titles," &_
        titleauthor WHERE titles.title_id = " &_
        titleauthor.title_id AND au_id = strAuId & "'")

        Set objXMLBooks = RecordsetToXMLDoc( objRS1, "book" )

        for each objBookNode in objXMLBooks.selectNodes("books/book")
            'Get this book's authors
            strTitleId = objBookNode.selectSingleNode("title_id").text
            Set objRS1 = Execute(objCon,
            "SELECT authors.au_id, au_fname, " "au_lname, au_ord," & _
            "royaltyper FROM authors, titleauthor " & _
            "WHERE authors.au_id = titleauthor.au_id and " & _
            "title_id = '" & strTitleId & "' ORDER BY au_ord")
            Set objXMLAuthors = RecordsetToXMLDoc( objRS1, "author" )
            'Add the authors to this book node
            objBookNode.appendChild
            objXMLAuthors.selectSingleNode("authors")
        next
        'Now add this books node, to the author node
        objAuthorNode.appendChild objXMLBooks.selectSingleNode("books")
    next
    Set GetXMLFromDB = objXML
end function
```

Figure 4.20 GetXMLFromDB Function

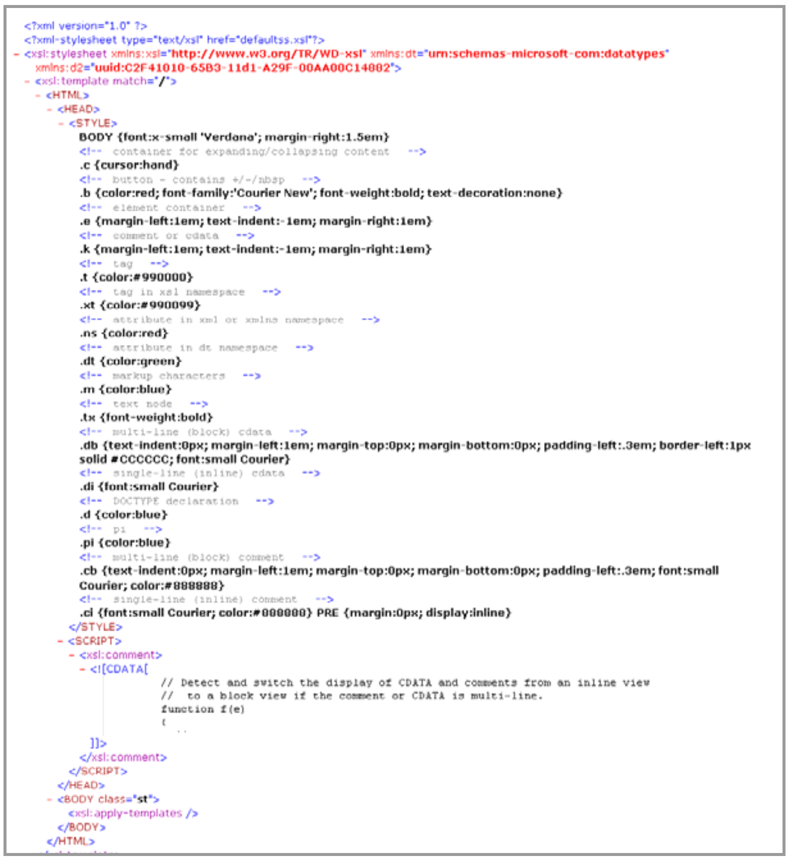
GetXMLFromDB function given in Figure 4.20 simply retrieves data from the data source and converts the data to XML document.



```
sub ShowAuthorsView()
    if Request("xmlonly") = "true" then
        Response.Write "<a href=xmlbyhand.asp>View Sample Application
                        </a><br><br>"
        Response.Write TransformXML(GetXMLFromDB(), "xsl/defaultss.xsl")
    else
        Response.Write "<a href=xmlbyhand.asp?xmlonly=true>View XML
                        </a><br><br>"
        If Request("cache") = "off" then
            Response.Write TransformXML( GetXMLFromDB(),
            "xsl\authors2.xsl")
        ElseIf Request("cache") = "clear" then
            ClearCachedXSLTemplate( "xsl\authors2.xsl" )
        Else
            Response.Write TransformXML2( GetXMLFromDB(), _
                    GetCachedXSLTTemplate( "xsl\authors2.xsl") )
        End If
    end if
end sub
```

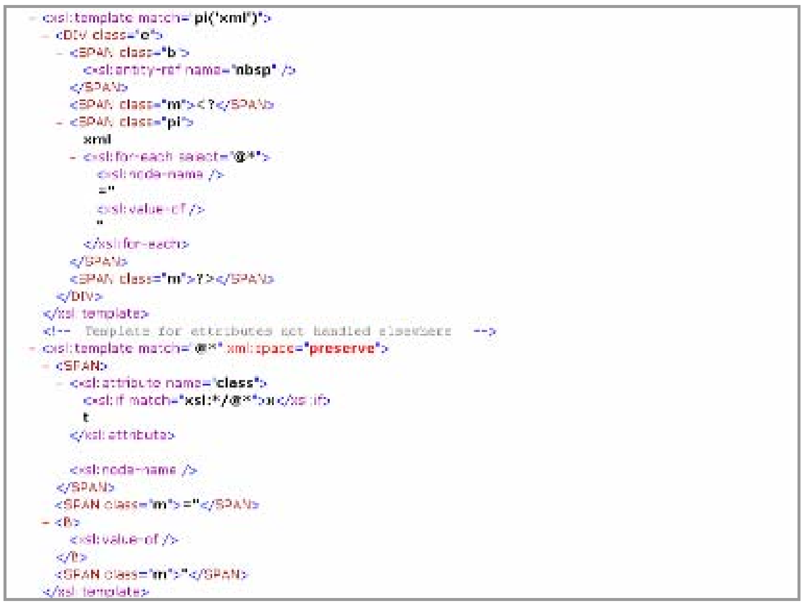
Figure 4.21 ShowAuthorsView Function

ShowAuthorsView function shown in Figure 4.21 translates the data in XML format to style defined in XSL document in Figure 4.22 and Figure 4.23. As previously shown, using decide block for large size of data may cause extra load. Therefore, using a separate XSL style document for each language (e.g. **en.xsl** or **en/default.xsl**) will prevent overloads due to excessive use of decision operations in XSL.



```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="defaultss.xsl"?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl" xmlns:dt="urn:schemas-microsoft-com:datatypes"
    xmlns:d2="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882">
  - <xsl:template match="/">
    - <HTML>
      - <HEAD>
        - <STYLE>
            BODY {font:x-small 'Verdana'; margin-right:1.5em}
            <!--  container for expanding/collapsing content   -->
            .c {cursor:hand}
            <!--  button - contains +/-/nbsp   -->
            .b {color:red; font-family:'Courier New'; font-weight:bold; text-decoration:none}
            <!--  element container   -->
            .e {margin-left:1em; text-indent:-1em; margin-right:1em}
            <!--  comment or cdata   -->
            .k {margin-left:1em; text-indent:-1em; margin-right:1em}
            <!--  tag   -->
            .t {color:#990000}
            <!--  tag in xsl namespace   -->
            .xt {color:#990099}
            <!--  attribute in xml or xmlns namespace   -->
            .ns {color:red}
            <!--  attribute in dt namespace   -->
            .dt {color:green}
            <!--  markup characters   -->
            .m {color:blue}
            <!--  text node   -->
            .tx {font-weight:bold}
            <!--  multi-line (block) cdata   -->
            .db {text-indent:0px; margin-left:1em; margin-top:0px; margin-bottom:0px; padding-left:.3em; border-left:1px
            solid #CCCCCC; font:small Courier}
            <!--  single-line (inline) cdata   -->
            .di {font:small Courier}
            <!--  DOCTYPE declaration   -->
            .d {color:blue}
            <!--  pi   -->
            .pi {color:blue}
            <!--  multi-line (block) comment   -->
            .cb {text-indent:0px; margin-left:1em; margin-top:0px; margin-bottom:0px; padding-left:.3em; font:small
            Courier; color:#888888}
            <!--  single-line (inline) comment   -->
            .ci {font:small Courier; color:#888888} PRE {margin:0px; display:inline}
          </STYLE>
        - <SCRIPT>
          - <xsl:comment>
            - <![CDATA[
                        // Detect and switch the display of CDATA and comments from an inline view
                        //  to a block view if the comment or CDATA is multi-line.
                        function f(e)
                        {
                          ..
              ]]>
            </xsl:comment>
          </SCRIPT>
        </HEAD>
      - <BODY class="st">
          <xsl:apply-templates />
        </BODY>
      </HTML>
```

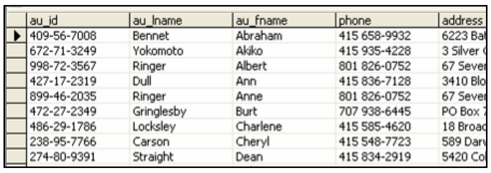
Figure 4.22 XSL Contents (html style definitions and embedding script)



```
- <xsl:template match="pi('xml')">
  - <DIV class="e">
    - <SPAN class="b">
        <xsl:entity-ref name="nbsp" />
      </SPAN>
      <SPAN class="m"><?</SPAN>
    - <SPAN class="pi">
        xml
      - <xsl:for-each select="@*">
          <xsl:node-name />
          ="
          <xsl:value-of />
          "
        </xsl:for-each>
      </SPAN>
      <SPAN class="m">?></SPAN>
    </DIV>
  </xsl:template>
  <!--  Template for attributes not handled elsewhere   -->
- <xsl:template match="@*" xml:space="preserve">
  - <SPAN>
    - <xsl:attribute name="class">
        <xsl:if match="xsl:*/@*">x</xsl:if>
        t
      </xsl:attribute>

      <xsl:node-name />
    </SPAN>
    <SPAN class="m">="</SPAN>
  - <B>
      <xsl:value-of />
    </B>
    <SPAN class="m">"</SPAN>
  </xsl:template>
```

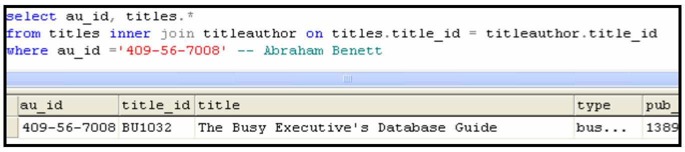
Figure 4.23 XSL Contents (representing the data fields)

In this example, SQL Server database contains the authors and books tables. Data stored in these tables are shown in Figures Figure 4.24 and Figure 4.25**.**



au_id	au_lname	au_fname	phone	address
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bat
672-71-3249	Yokomoto	Akiko	415 935-4228	3 Silver (
998-72-3567	Ringer	Albert	801 826-0752	67 Sever
427-17-2319	Dull	Ann	415 836-7128	3410 Blo
899-46-2035	Ringer	Anne	801 826-0752	67 Sever
472-27-2349	Gringlesby	Burt	707 938-6445	PO Box 7
486-29-1786	Locksley	Charlene	415 585-4620	18 Broac
238-95-7766	Carson	Cheryl	415 548-7723	589 Darv
274-80-9391	Straight	Dean	415 834-2919	5420 Co

Figure 4.24 Authors Table on SQL Server

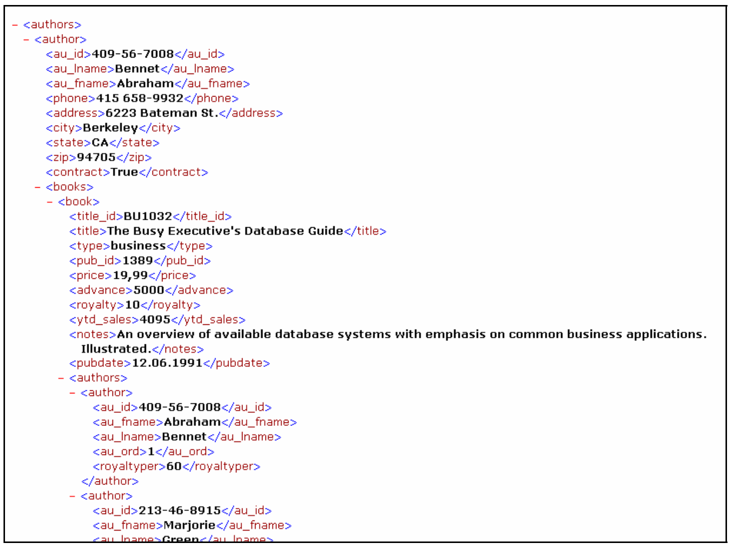


```
select au_id, titles.*
from titles inner join titleauthor on titles.title_id = titleauthor.title_id
where au_id ='409-56-7008' -- Abraham Benett
```

au_id	title_id	title	type	pub_
409-56-7008	BU1032	The Busy Executive's Database Guide	bus...	1389

Figure 4.25 Title_Author Table (SQL Server)

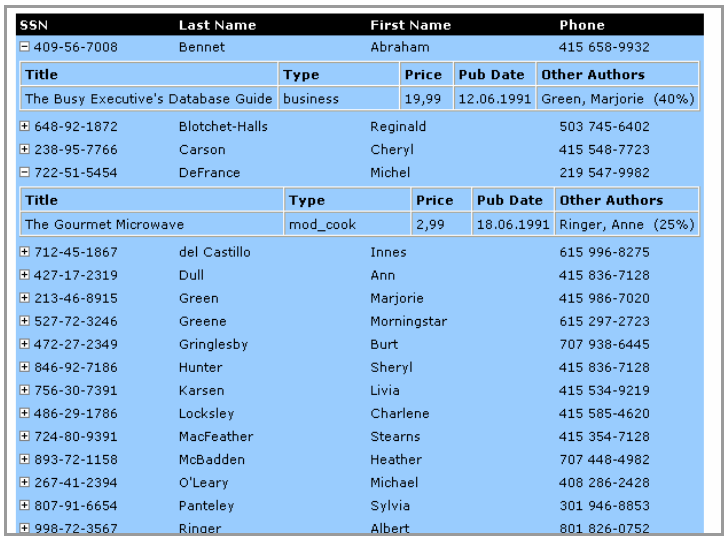
When author data and books for each author is retrieved from the database, GetXMLFromDB function converts the records in the result set as shown in Figure 4.26.



```
- <authors>
  - <author>
      <au_id>409-56-7008</au_id>
      <au_lname>Bennet</au_lname>
      <au_fname>Abraham</au_fname>
      <phone>415 658-9932</phone>
      <address>6223 Bateman St.</address>
      <city>Berkeley</city>
      <state>CA</state>
      <zip>94705</zip>
      <contract>True</contract>
    - <books>
      - <book>
          <title_id>BU1032</title_id>
          <title>The Busy Executive's Database Guide</title>
          <type>business</type>
          <pub_id>1389</pub_id>
          <price>19,99</price>
          <advance>5000</advance>
          <royalty>10</royalty>
          <ytd_sales>4095</ytd_sales>
          <notes>An overview of available database systems with emphasis on common business applications.
            Illustrated.</notes>
          <pubdate>12.06.1991</pubdate>
        - <authors>
          - <author>
              <au_id>409-56-7008</au_id>
              <au_fname>Abraham</au_fname>
              <au_lname>Bennet</au_lname>
              <au_ord>1</au_ord>
              <royaltyper>60</royaltyper>
            </author>
          - <author>
              <au_id>213-46-8915</au_id>
              <au_fname>Marjorie</au_fname>
              <au_lname>Green</au_lname>
```

Figure 4.26 XML output file of GetXMLFromDB function

Finally, XML data generated in previous step is combined with XSL document (that is, XSL schema is applied) and HTML content is generated as shown in Figure 4.27.



SSN	Last Name	First Name	Phone
⊟ 409-56-7008	Bennet	Abraham	415 658-9932

Title	Type	Price	Pub Date	Other Authors
The Busy Executive's Database Guide	business	19,99	12.06.1991	Green, Marjorie (40%)

SSN	Last Name	First Name	Phone
⊞ 648-92-1872	Blotchet-Halls	Reginald	503 745-6402
⊞ 238-95-7766	Carson	Cheryl	415 548-7723
⊟ 722-51-5454	DeFrance	Michel	219 547-9982

Title	Type	Price	Pub Date	Other Authors
The Gourmet Microwave	mod_cook	2,99	18.06.1991	Ringer, Anne (25%)

SSN	Last Name	First Name	Phone
⊞ 712-45-1867	del Castillo	Innes	615 996-8275
⊞ 427-17-2319	Dull	Ann	415 836-7128
⊞ 213-46-8915	Green	Marjorie	415 986-7020
⊞ 527-72-3246	Greene	Morningstar	615 297-2723
⊞ 472-27-2349	Gringlesby	Burt	707 938-6445
⊞ 846-92-7186	Hunter	Sheryl	415 836-7128
⊞ 756-30-7391	Karsen	Livia	415 534-9219
⊞ 486-29-1786	Locksley	Charlene	415 585-4620
⊞ 724-80-9391	MacFeather	Stearns	415 354-7128
⊞ 893-72-1158	McBadden	Heather	707 448-4982
⊞ 267-41-2394	O'Leary	Michael	408 286-2428
⊞ 807-91-6654	Panteley	Sylvia	301 946-8853
⊞ 998-72-3567	Ringer	Albert	801 826-0752

Figure 4.27 HTML output

The last part of a multilingual content management system is the database design issue, which has been discussed in detail in previous chapters. To summarize, before designing the multilingual database, the first thing that should be done is to decide what parts of the content should be stored on the database. After classifying the content parts according to their language and culture dependency and the frequency at which these parts will be changed, initial database objects can be identified. Considering the database server's physical and software capabilities, effect of normalization and replication should be analyzed for each data object and the best method should be implemented with object-based approach.

REFERENCES

[1] G. Miller, “Five Papers on WordNet,” Int’l J. Lexicography, vol. 4, no. 3, 1990.

[2] R. Riloff, “Automatically Constructing A Dictionary for Information Extraction Tasks,” Proc. 11th Nat’l Conf. Artificial Intelligence, AAAI/MIT Press, Menlo Park, Calif., 1993, pp. 811–816.

[3] S. Sonderland et al., “Crystal: Inducing a Conceptual Dictionary,” Proc. 14th Int’l Joint Conf. Artificial Intelligence, Morgan Kaufmann, San Francisco, 1995, pp. 1314–1319.

[4] R. Gaizauskas and K. Humphreys. “XI: A Simple Prolog-Based Language for Cross classification and Inheritance,” Proc. 6th Int’l Conf. Artificial Intelligence: Methodologies, Systems, Applications (AIMSA 96), 1996, pp. 86–95.

[5] K. Spark Jones and P. Willett, eds., Readings in Information Retrieval, Morgan Kaufmann, San Francisco, 1997.

[6] S. Sanda Harabagiu and D. Moldovan, “Textnet: A Text-Based Intelligent System,” Natural Language Engineering, vol. 3, 1997, pp.171–190.

[7] Fluhr, C., Schmit, D., Ortet, P., Elkateb, F. and Gurtner, K., SPIRIT-W3: A Distributed Cross-Lingual Indexing and Search Engine. in Inet 97, (Kuala Lumpur, 1997), The Internet Society.

[8] Aone, C., Charocopos, N. & Gorlinsky, J.: An Intelligent Multilingual Information Browsing and Retrieval System Using Information Extraction, Proceedings: (1997).

[9] P. Vossen, EuroWordNet: A Multilingual Database with Lexical Semantic Networks, Kluwer Academic Publishers, 1998.

[10] R. Power, D. Scott, and R. Evans. What You See Is What You Meant: direct knowledge editing with natural language feedback. In: Proceedings of the 13th Biennial European Conference on Artificial Intelligence, pages 675 – 681, 1998.

[11] Barber, W., & Badre, A.: Culturability: The Merging of Culture and Usability. Paper presented at the "Our Global Community": 4th Conference on Human Factors & the Web: (1998).

[12] Franz, M., McCarley, J.S. and Roukos, S., Ad hoc and Multilingual Information Retrieval at IBM. in NIST Special Publication 500-242: The Seventh Text Retrieval Conference (TREC-7), (Gaithersburg, MD., 1998), NIST.

[13] K. Humphreys et al., “University of Sheffield: Description of the LaSIE-ii System as Used for MUC7,” Proc. 7th Message Understanding Conf. (MUC-7), Morgan Kaufman, San Francisco, 1998; www.saic.com.

[14] R. Power, D. Scott. Multilingual authoring using feedback texts. In: Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Comp. Linguistics, pages 1053 – 1059, 1998.

[15] Braschler, M., Peters, C. and Schäuble, P., Cross-Language Information Retrieval (CLIR) Track Overview. in NIST Special Publication 500 - 246: The Eighth Text Retrieval Conference (TREC-8), (Gaithersburg, MD., 1999), NIST.

[16] Halavais, A.: National borders on the World Wide Web. New Media & Society, 2(1), 7-28, (2000).

[17] C. Brun, M. Dymetman, and V. Lux: Document structure and multilingual authoring. In Proceedings of First International Natural Language Generation Conference (INLG '2000): (2000).

[18] M. Dymetman, V. Lux, A. Ranta: XML and Multilingual Document Authoring: Convergent Trends. COLING 2000: 243-249, (2000).

[19] S. Schulz, and U. Hahn. Knowledge engineering by large-scale knowledge reuse: experience from the medical domain. In: Proceedings of the 7th International Conference on Knowledge Representation and Reasoning, pages 601 – 610, 2000.

[20] Huang, Shihong and Tilley, Scott: "Issues of Content and Structure for a Multilingual Web Site". Proceedings of the 19th Annual International Conference on Systems Documentation (SIGDOC 2001: Santa Fe, NM; October 21-24, 2001): 103-110, (2001).

[21] Morgan, T., Luttrell, C. and Liu, Y.: Designing Multilingual Web Sites: Applied Authoring Techniques, Proceedings of the 19th Annual International Conference on Computer Documentation: 230-231, (2001).

[22] Zahedi, F., Pelt, W.V. van & Song, J.: A Conceptual Framework for International Web Design. IEEE Transactions on Professional Communication: 44(2), 83-103, (2001).

[23] Sabarís, M.F., Alonso, J.L.R, Dafonte, C., Arcay, B.: Multilingual Authoring Through an Artificial Language. MT SUMMIT-VIII, , (2001)

[24] Androutsopoulos, I., Calder, J., Not, E., Pianesi, F., Roussou, M.: Universal Translation Language: a New Approach to Multilingual Human Assisted Machine Translation. Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue: 25-29, (2001).

[25] D. Brandon Jr.: Localization of web content, Journal of Computing Sciences in Colleges: 17(2), 345 – 358, (2001).

[26] R. Basili et al., “Multilingual Authoring: The Namic Approach,” Proc. Workshop Human Language Technology and Knowledge Management (ACL-EACL 2001), 2001.

[27] R. Basili and F.M. Zanzotto, “Parsing Enginering and Empirical Robustness,” J. Natural Language Eng., vol. 8, nos. 2–3, 2002, pp. 97–120.

[28] Howlett S., Jennings D.: SQL SERVER 2000 AND XML, Developing XML-Enabled Data Solutions for the Web. MSDN Magazine: 17(1), (2002).

[29] Tonella, P., Ricca, F., Pianta, E. and Girardi, C. : Restructuring Multilingual Websites. Proceedings of the International Conference on Software Maintenance (ICSM’02): 290-299, (2002)

[30] Dymetman, M.: Document Content Authoring and Hybrid Knowledge Bases. Proceedings of the 2003 ACM symposium on Document Engineering: 193 – 202, (2003).

[31] G. Lapalme, C. Brun and M. Dymetman : XML Based Multilingual Authoring. Proceedings of PACLING'03: 191-199, (2003)

[32] J. A. Rydberg-Cox, L. Vetter, S. Rüger, D. Heesch : Cross-lingual searching and visualization for Greek and Latin and old Norse texts. Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries: 4, 383 – 383, (2004).

[33] Sql-Server-Performance.Com. Tips on Optimizing SQL Server Database Design. Internet WWW-page, URL: http://www.sql-server-performance.com/database_design.asp (02.05.2005).

[34] TopXML. XSLT in Context. Internet WWW-page, URL: http://www.topxml.com/xsl/articles/xslt_what_about/31290101.asp (02.05.2005).

VITA

Selvihan Nazlı Yavuzer was born in Ankara. She received his B.S. degree in Computer Engineering from the Bahçeşehir University in 2003. Since then she has been a research assistant in the Department of Computer Engineering. Her main areas of interest are content management systems, computer-aided learning and computer graphics.