Spring 2010

Database Localization in a Test Environment

Thuy-Uyen Tran
Regis University

Regis University
College for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

DATABASE LOCALIZATION IN A TEST ENVIRONMENT

A THESIS

SUBMITTED ON 18TH OF MARCH, 2010

TO THE DEPARTMENT OF INFORMATION TECHNOLOGY

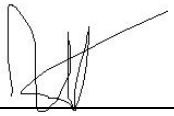
OF THE SCHOOL OF COMPUTER & INFORMATION SCIENCES

OF REGIS UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF MASTER OF SCIENCE IN

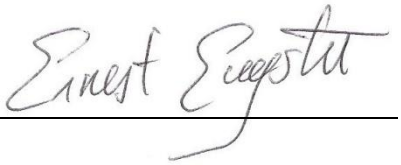
DATABASE MANAGEMENT

BY



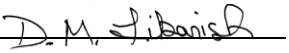
Thuy-Uyen Tran

APPROVALS



Ernest Eugster, Thesis Advisor

Shari Plantz-Masters



Daniel M Likarish

Abstract

One of the most significant challenges facing database software publishers is globalizing their products to tap into potentially profitable overseas business opportunities. The challenges of making products work overseas are many, ranging from budgeting for globalization projects to requiring developers to be skilled in localization. Over the years, software publishers have developed and implemented various tools and processes for globalization and localization. While there is considerable literature available on software development lifecycles and case studies, few studies have focused on globalization of database software. This thesis will examine the extent to which database publishers are using internationalization tools and processes to help them solve globalization challenges. Thesis deliverables will be specified tool evaluations and workarounds for database localization.

Keywords: Database localization, localization tools, globalization

Acknowledgements

It is a pleasure to thank those who made this thesis possible. I would like to gratefully acknowledge the supervision of Dr. Ernest Eugster in this thesis. He has provided support in numerous ways from evaluating and giving guidance for my work to attending the thesis presentation. I would like to thank Prof. Shari Plantz-Masters for her educational resources about thesis theory and practice. Her help in scheduling the thesis completeness is much valued.

I am grateful to all my professors from Regis University and colleagues who have given me the opportunity to discover and explore the foundation of this thesis.

Last but not least, I am forever indebted to my parents, in-laws, siblings, friends, husband and son for their understanding, endless patience and encouragement when it was most needed.

Table of Contents

List of Figures

List of Tables

Chapter 1 – Introduction

For decades, US database developers have been building systems based exclusively on domestic business and technical requirements. But in today's increasing global market, supporting only one language and market is no longer an option. To access more markets, databases must be localized. US developers need to adapt databases to different languages and regional differences, without engineering changes. In addition, databases have to support local-specific components and translating text. This chapter defines and distinguishes between globalization, internationalization and localization. It also describes challenges to globalization and the role of localization tools. This chapter sets the stage for the reminder of the research.

Globalization

Viewed historically, database designers have been conditioned to build systems based on domestic requirements. Conceptually, the requirements document focused on local and national business needs. Requirements lead to design, which lead to implementation, testing, deployment and maintenance (Yeo, 2001). These stages in the development life cycle made the implicit assumption that no differences in language, culture, legal and technical existed. This "Domestic by Design" approach worked well for many years, with any global design and development effort considered a luxury.

But globalization is no longer a luxury. It is an economic necessity for software publishers. According to a survey by LISA (Localization Industry Standards Association), some software companies have international revenues hovering around 70% (The Localization Industry Standards Association, n.d.). In the database industry, Oracle Technology Network is a leading global player. Despite the current economic recession, Oracle Technology Network 's revenues from Asia-Pacific clients were $3.4 billion in 2009, up 41% from 2008. The

geographic segment comprising of Europe, the Middle East and Africa accounted for $7.9 billion in revenues, a 33% gain from the prior year. Revenues from North, South and Central Americas were $11.9 billion, 17% more than in 2008 (Oracle Technology Network, 2009). Similarly, Sybase's international revenues grew from 46% in 2006 to 50% in 2008 (Sybase, 2009).

What has been driving the change for increased globalization? As the Internet and economic forces flatten the world, the idea of doing business in one country is being replaced by doing business in many countries as possible to maximize sales. "Just as the flattening of the world is starting to equalize and diversify the flow of investment – so globalization is no longer driven primarily by American or Western multinationals (Friedman, 2007)." The software industry is no exception. Microsoft Corporation, which has over 90,000 employees in 105 countries, saw its share of international revenue increase from 35% in 2007, to 46% in 2008, to 72% in 2009. As in Microsoft Corporation's case, strong international business can help offset slow domestic sales.

	2009	2008	2007
U.S.	**$5,529**	**$12,682**	$12,902
International	**14,292**	**11,132**	7,199
Income before income taxes	**19,821**	**23,814**	20,101

Table 1. Microsoft Corporation Income Before Tax (Millions of dollars. Source: Microsoft Corporation, 2009)

Globalization has emerged as a crucial part of the database. Database globalization encompasses the architecture and design to store, process, and retrieve data in native languages. It ensures that database utilities, error messages, sort order, and date, time, monetary, numeric, and calendar conventions automatically adapt to any native language and locale. The database architecture is developed as a first step toward designing a database that supports multilingual

applications. Software globalization involves market knowledge, linguistic and cultural skills as well as engineering (Yuri, 2007).

Globalization is a growing, but still young discipline. Conceptually, globalization is viewed as a cycle. According to LISA, globalization can best be thought of as a cycle rather than a single process, as shown in Figure 1. In this figure, the two primary technical processes that comprise globalization, internationalization and localization, are viewed as part of a global whole (The Localization Industry Standards Association, n.d.).

Figure 1. Globalization Cycle

There is no single, accepted definition. On the other hand, there is no shortage of definitions. Here are some often heard definitions:

- ***Globalization consists of internationalization and localization.*** People get the terms mixed up but in reality they are different concepts. This thesis focuses on localization.
- ***Globalization is not just internationalization.*** Internationalization encompasses the planning and preparation stages for a product which is built by design to support global markets. Preparing your product for international markets involves a myriad of technical considerations which must be addressed before a software product can achieve success in a new country. A localizable product or service should support double byte character sets and reserve enough space for languages that have longer text size. Internationalization process ensures product planning and product implementing that allow localization. Internationalization is sometimes written as "I18n", where 18 is the number of letters between 'I' and 'n'.
- ***Globalization is not just localization***. Localization is a process while globalization is a cycle. By translating texts and also adding locale specific components, localization processes adapt internationalized software for specific languages or regions. Localization goes beyond literal translation. In addition to idiomatic language translation, numerous locale features such as addresses, phone numbers, time zones, dates, national holidays, currencies, measurements, local color sensitivities, product or service names, and gender roles must all be considered (Sun Developer Network, n.d.). Localization is sometimes written as "l10n", where 10 is the number of letters between 'l' and 'n'.

- ***Globalization is not the same as translation.*** Language translation is only a part of localization. Being a large part of localization, language translation can sometimes be assisted with automatic language translation or machine translation. However, human translation is necessary in order to ensure the quality of the translations.

Database Development Life Cycle and Globalization Challenges

Database developers have been using a traditional system development life cycle which includes the following phases: planning, design, implementation and testing, deployment and maintenance (Yeo, 2001). To successfully provide a localized product or service, globalization needs to be involved in all phases of the development life cycle (Lionbridge, n.d.). In reality, many product or service providers have not always followed this goal. As the result, the globalization cycle faces a range of challenges from planning to design, from implementation to maintenance. Globalization challenges and their corresponding resolutions are described in each phase of the development life cycle as follows. This thesis focuses on the challenge of localization tools and the workarounds which fall into implementation phase of the development life cycle.

1. **Planning**

Business requirements can be gathered by many people including project managers, stakeholders and users. Questions such as who the application is going to be developed for, how is the application going to be used are introduced during the planning phase. The scope of the application development should be discussed in this phase: input, output and functionalities of the application. These are general questions that need to be answered during the gathering or

planning phase. A workflow which involves business logic of how data is collected, stored, retrieved and displayed through the user interface is an end product of this phase.

A common challenge during planning is that participants easily fail to ask previously mentioned questions to define the scope of a localization project. This also means that the participants could fail to identify what needs to be modified for the localizing product or service to operate properly in target languages. Planning also faces another challenge of getting budget-efficient and time-efficient workflows. Both product or service providers and localization vendors are required to acknowledge this workflow to communicate accordingly to achieve the goal. Localization vendors should educate their product or service providers about budget and time effective factors such as localization requirements, procedures, and tools. Localization vendors should also be able to manage localization resources accordingly. Localization resources can either be the technologies or the people that are involved in the localization process.

In accomplishing globalization success, the LISA organization mentioned that international markets should not be treated as secondary concern to cut costs (The Localization Industry Standards Association, n.d.). However, not all database products can be fully localized. Important factors which influence the degree of localization include “the nature and scope of the product concerned, the size of the target market and audience, the length of the product lifecycle and anticipated update frequencies, competitor behavior, market acceptance, and national or international legislation” (Yuri, 2007). Significant and thorough market research should help to avoid market disasters before product design takes place.

2. **Design**

The results from the planning phase are incorporated into a database design document which is produced during the design phase. Software architecture is the main focus of database designers in this phase. Unified Modeling Language or UML which includes visual models of the software application can be provided in the design phase. The details on how the system will work can be described in the design phase.

A common problem is late discovery of international design failures. Software internationalization has always been a "complex and time consuming task especially for the older legacy software products where the underlying technologies provide minimal support for internationalization" (Yuri, 2007). More than likely, software products have been designed without any consideration for other languages. To achieve localization, making a product or service localizable earlier in the development life cycle, during planning and designing, is important. The actual localization process can be more efficient and effective once a product or service is designed for localization.

The approach to software internationalization is to ensure language independence throughout the entire program code and the internationalized version can be localized into other languages. During the design phase, software designers and developers should consider all of the required features including addresses, phone numbers, time zones, and national holidays. These features should be identified early so they can be declared later in different culture locales. Adding these features later will only increase localization costs (Sun Developer Network, n.d.).

Other considerations might be translatable text format and third party products. Translatable text format includes translatable text encoding, size, and direction. A localizable product or service would support both single and double byte character sets (Sun Developer

Network, n.d.). This type of product or service would also reserve enough space for languages that have longer text size. Text resizing costs should be added into project accounts if translatable text size was not considered in advance, during internationalization. Same considerations should be drawn into attention to third party products so that they also support localization.

Another consideration is for a localizable product or service to support bidirectional languages as if these languages are part of the market requirement. Bidirectional languages contain texts in both directionalities: right-to-left (RTL) and left-to-right (LTR). Common bidirectional languages are Arabic, Persian, and Hebrew.

Besides considerations of translatable text format, a product or service can also use third party products to accomplish data input, data output, graphics, or user interactions. These third party products should be carefully studied before using to avoid any locale limitations such as character corruptions and truncations. A product or service analysis can help identify any locale limitations and issues before localization takes place. Localization would cost time and money to fix any internationalization errors due to design or development oversights. In some extreme cases, products or services that were not properly internationalized cannot be localized without reengineering.

The approach to database localization is database migration in which an original is transferred to a target database for localization. As required in any localizable software, target database should have proper encodings and data types. By properly encoding the target database, the database can support languages that use double byte character sets. Data type of database objects defines what kind of data and which length or size of its stored value. Database

developers should consider reserving more length or size for localizable fields since translated texts such as German are usually longer.

Another important aspect in designing localizable databases is multi-locale schemas. Multi-locale schemas can be field-oriented, row-oriented, or table-oriented (Sisulizer, n.d.). Chapter 3 of this thesis will cover more details about multi-locale schemas. Depending on different database architectures, warehouse or correlation databases, database designers would effectively choose which schema to use. Field-oriented schemas would be more effective for warehouse or library databases. In contrast to field-oriented, row-oriented schemas should be used for correlation or value-based databases (Stonebraker, Abadi, Batkin, Chen, Cherniack, Ferreira, Lau, Lin, Madden, O'Neil, Rasin, Tran & Zdonik, 2005).

3. **Implementation and testing**

In one of the most time consuming phases of development cycle, implementation and testing, code is produced and tested against the previous design phase by database developers and testers. Implementation might overlap with both the design and testing phases to fulfill both the business and technical requirements. CASE (Computer-aided software engineering) tools can be used to automate business modeling, data models with Entity-relationship or ER diagram and even database creation SQL and store procedures (Baik, 2000). Documenting of internal database design takes place during implementation and testing phase to ensure future development maintenance and enhancement.

On the product or service providers' side, implementation phase faces a common issue as developers attempt to update features later in the development cycle and rush into completing the product. If the updated features come during or after the localization process, another version of the product should be sent out to the localization vendor. This new version of the product might

require a new schedule for another round of translation analysis and preparation. As the result, this common issue delays localized product release and increases costs. By rushing the completion, product quality is lowered and causes assurance delays in the development cycle. During implementation and testing phase, developers should take time to implement and document the changes for the localizable produce or service. Carrying out this step before localization would lower the cost of managing the localization schedule and resources.

On the localization vendor side, various tools exist to help automate localization efforts. Tools such as SDLX, Trados and WorldServer help engineers in processing files by separating localizable texts from codes. However, these tools have their weaknesses which affect localization quality and schedule. These specified tools should support most file types including databases. In fact, most localization tools lack support for database file formats. This thesis investigates workarounds for engineers as they localize database file formats.

Another important aspect is to consider in-country and linguistic proof reviews to ensure translation quality. These reviews and user interface testing processes need to be included in both the schedule and the budget. User interface testing is also required as part of the review process since translations need to be verified in context of an integrated product or service. Version control is also essential to keep track of file versions during reviewing process. As working with multiple versions, a common issue is not managing the most updates properly.

4. **Deployment and maintenance**

Deployment and maintenance phase involves releasing the approved implementation into a production environment and also maintaining the application after release. This phase might include multiple rounds of deploying and maintaining to improve the application performance.

A well-known challenge from deployment and maintenance phase is the lack of maintenance specifications for localized products or services. Development life cycle used for developing United States market applications focuses on defining requirements exclusively for local businesses. As businesses are interested in globalizing their products, there are many locale factors that they should recognize. These factors reside in language considerations, configuring regional settings and language package (Microsoft Corporation, n.d.).

There is usually a lack of development support after the product has been deployed. Developers who used to work on the localizing product are assigned to other projects. If there was no documentation about this localizing product, project managers are unable to provide source files and product information required by the localization vendor. The approach to development support issue is to ensure developers archive and document different versions of the source files.

Summary

This chapter defined globalization and also discussed its relationship to internationalization and localization. We have explained that architecture is a key concept in database globalization, in terms of early insights it provides into adopting system to multiple languages and minimizing the challenges of adapting a product to global markets. While database localization is crucial, only a few localization tools exist that support the process as the next chapter shows. A goal of this thesis is to investigate workarounds to the common challenge of a lack of tool support. The next chapter reviews existing academic literature to show the importance of globalization in database development.

Chapter 2 – Review of Literature and Research

As the field of global software development has expanded, researchers and practitioners have shown increased interest in the globalization cycle which includes internationalization and localization processes. However, the literature is insufficient when it comes to databases and associated localization tools. This thesis attempts to redress this imbalance.

Globalization and Localization Challenges

A number of researchers examined the conditions for successful global software development. Yeo (1996) introduced the approaches used in designing global software by making a distinction between internationalization and localization. "The internationalization process is, in a sense, a means to an end; the end being localization" (Yeo, 1996). Similarly, Yeo and Barbour (1996) defined internationalization as "the process of making a system or application software independent of or transparent to natural language." While translation is "the process of converting written text or spoken words to another language" (Esselink, 2000), software localization refers to a process of software modification to include different culture locales.

Esselink (2000) described the difference between translation and localization. Translation is only one activity of localization which includes other tasks such as project management, software engineering, testing, and desktop publishing. Users around the globe expect software applications to not only speak their language, but also reflect their culture. To this end, Chroust (2007) stated that localization "involves much more than a pure language translation: it implies the transfer of the software product into another culture taking into account all aspects of cultural discrepancies."

Localization is an important part of the software development life cycle. Rombach (2007) cited the challenges that come from different development teams who are separated by various boundaries, such as contextual, organizational, cultural, temporal, geographical, and political. Many globalization organizations have distributed software development across the world to benefit from global resources, attractive cost structures, and round-the-clock development to achieve cycle-time acceleration and accommodate to local markets.

The main challenge from this method of global software development is apparent communication among global teams with different cultures, backgrounds, time zone, and organizations. Begel and Nagappan (2008), in survey of software engineers at Microsoft Corporation, found that a high proportion of engineers are directly involved with global software development. They found that 50% of the respondents regularly collaborate with people at least three time zones away. But they also mentioned the difficulties of global communication. Different time zones have caused the majority of the coordination problems with distributed teams. Meetings and trainings had to be recorded for later review among distributed teams. Working in non-standard business hours in some cases was required to communicate more efficiently with other teams. Email tags also delayed project process. Extending communications such as distant visits and in-advance contacts have helped quickly unblock distributed coworkers. However, the distributed teams had to sacrifice their personal time. Such difficulties were reported to be most critical in global software development.

Another difficulty is requirements gathering. Abufardeh and Magel (2009) described how globalization requirements are scattered through the entire system development life cycle. The impact of these crosscutting requirements introduced many challenges. For instant, the character set of application contents should be configured in design and implementation to avoid

issues during deployment. Crosscutting requirements inherent both scattering and tangling across multiple development classes. As a result, the reusability, extensibility, and traceability of the software artifacts were reduced. In the same paper, the authors suggested removing scattering and tangling properties by first identifying and then separating the design and code of crosscutting behavior into independent components. Abufardeh & Magel (2009) discussed two important globalization issues: when developers should identify and document crosscutting requirements; and where it should be handled throughout the software life cycle.

While there is considerable literature available on software development life cycles and case studies, few studies have focused on localization of database software.

Evers (1998) and Abufardeh & Magel (2009) stated how imperative database localization is. In earlier development efforts, software localization resided in the user interface or presentation layer since internationalization used to focus larger on English to Latin-based localization. Languages such as Asian require multi-byte character sets to accommodate additional accents and symbols. Bidirectional languages such as Hebrew or Arabic include texts in both text directionalities, both right-to-left (RTL) and left-to-right (LTR). The concept of internationalization therefore was extended to include the business logic layer – no longer just the presentation layer. Abufardeh & Magel (2009) noted that "data storage, data display, data formatting, searching, sorting, for bidirectional software such as Arabic, Hebrew, Urdu, and Farsi language are directly affected by data, character encoding, and the character set used. " A localization vendor, Lingobit Localization on Demand, also emphasized the need for database localization because so many modern applications are based on databases. "As companies go with their software worldwide, they need to localize their applications. As most of the data in

many applications is kept in database files, to localize application, you have to do database localization" (Lingobit Localization On Demand, n.d.).

Similarly, Sisulizer (n.d.) stated the importance of databases localization.

- The information in your database is the product you sell. When you localize this information, you can offer it in new markets and increase your revenues.
- You may retrieve strings for your GUI localization from a database.
- Your database may contain data for your application that need to be localized as well as the application itself. One example is a product database for an online shopping system.

While many localization companies have specific tools, few of them support database localization. A list of available tools to assist translators and localization managers, with hyperlinks to product websites has confirmed the lack of database support in localization tools (Texin, 2005). Of the sixty tools in this list, only seven supported database localization. While well known tools such as SDLX Trados and Idiom WorldServer support file format such as Microsoft Office, HTML, XML, Java and C++, they do not support database localization. The seven tools that support database localization are Catalyst, Lingobit Localizer, Passolo, Sisulizer, Visual Localize, Crystal Translator Versions and Excitic. Among these database support tools, Sisulizer which has a well defined framework will be further analyzed in this thesis.

Summary

In conclusion, previous research has shown the importance and challenges of globalization in the software development life cycle. But little research existed which analyzed localization challenges. Also a survey showed that few tools exist. Chapter 3 will describe a methodology to investigate workarounds to this database localization challenge.

Chapter 3 – Methodology

In Chapter 2, we saw that database localization is an important part of the software development life cycle. Many applications use databases to store language specific information. But the existing database localization literature is insufficient; with little empirical work. Also a survey of globalization software tools showed a lack of support for database localization. This chapter lays out an approach to localizing a database. Localization is a matter of localizing fields, rows, tables and cloning the entire database. The chapter presents research questions addressed by this thesis and a method to answer these questions using a commercial database localization tool.

Research Questions

This thesis focuses on answering the following six research questions.

1. How important is database localization?
2. What are database localization specifications?
3. How do specialized tools work in supporting localization process?
4. What deficiencies, if any, exist in specialized tools and how do they impact database localization and process?
5. Do workarounds exist for overcoming tool deficiencies?
6. What is the procedure to prepare database content for localization in specialized tools which do not support database?

Methodology

To answer these questions, this researcher developed a method to test common database localization methods: Field, Row, Table and Clone Database Localization. In each test case, the researcher used Microsoft Office Access 2003 which is widely used in database localization.

Database Localization Methods

Database localization methods provide a framework for localizing databases with a variety of options. This research used a commercial localization tool called Sisulizer as a framework to investigate four methods: Field, Row, Table and Clone Database Localization.

1. Field Database Localization

In Field Database Localization, values of the localized fields are updated in the same row. The localized fields are isolated from the original field by: field name and field data. Field name includes target language identifications and field data includes the target language. Using Field Database Localization eliminates data redundancy. However, the table structure needs to be changed to include the localized fields.

ITEM_ID	ITEM_NAME_en	ITEM_NAME_de	ITEM_DESCRIPTION_en	ITEM_DESCRIPTION_de
0	Keyboard	Tastatur	Computer keyboard	Computertastatur
1	Monitor	Monitor	Computer monitor	Computermonitor

Table 2. *Sample Field Database Localization:* **ITEM_NAME_en** *versus* **ITEM_NAME_de**, **ITEM_DESCRIPTION_en** *versus* **ITEM_DESCRIPTION_de**

Table 2 shows a sample **ITEM** table which stores sale items and their descriptions. Item names and their descriptions are localizable so they are available to display in available languages. In this case, the available languages are English and German. For that reason, localizable fields are defined with language extensions to be '**ITEM_NAME_en***'* or '**ITEM_DESCRIPTION_en**' for English and '**ITEM_NAME_de***'* or

‘**ITEM_DESCRIPTION_de**’ for German. This is called Field Database Localization where values of the localized fields are updated in the same row. For instant, ‘Keyboard’ item which had **ITEM_ID** of ‘0’ was updated with both English and German values.

2. Row Database Localization

Row Database Localization is another method where the original table row is copied for each language. The copied rows have different language field values. Users can configure what fields need to be localized so that these fields’ data will be localized values. To accommodate this method, changes in database structure such as ensuring Unicode compliance and extended size for localized field values are required.

ITEM_ID	LANG_ID	ITEM_NAME	ITEM_DESCRIPTION
0	de	Tastatur	Computertastatur
0	en	Keyboard	Computer keyboard
1	de	Monitor	Computermonitor
1	en	Monitor	Computer monitor

Table 3. *Sample Row Database Localization: Localized Item*

Instead of having all localizable field values in the same row as in Field Database Localization, Table 3 shows separate rows of localizable field values per language. Therefore, localizable fields are not defined with language extensions but with ‘**LANG_ID**’ to be either ‘*en*’ for English or ‘*de*’ for German.

3. Table Database Localization

Table Database Localization concerns adding localized tables into the original database. The localized tables only include primary key and the localized fields. In addition, the localized table name includes target language identifications.

ITEM_ID	ITEM_NAME	ITEM_DESCRIPTION
0	Keyboard	Computer keyboard
1	Monitor	Computer monitor

ITEM_ID	ITEM_NAME	ITEM_DESCRIPTION
0	Tastatur	Computertastatur
1	Monitor	Computermonitor

Table 4. *Sample Table Database Localization:* **ITEM_en** *versus* **ITEM_de**

Table 4 displays tables **ITEM_en** for English and **ITEM_de** for Geman in which exact contents were included. The only difference between the two tables is the table names. In Table Database Localization, the table structure of a localizable database grows in respect to the number of available languages.

4. Clone Database Localization

Clone Database Localization method is concerned with creating a duplicate or an exact copy of the original database. Both original and cloned databases have the same tables and fields. The only difference is that the cloned database name has the target language identifications. This method offers the important advantage of not having to change the database structure, which can save time and money. The cloned database is localized in particular target languages. The following example shows an original versus a cloned database, **SUPPLY_en** versus **SUPPLY_de**. Sales database includes two tables of **BUSINESS** and **ITEM**.

BUSINESS_ID	BUSINESS_NAME	BUSINESS_ADDRESS
0	Dell	
1	HP	

ITEM_ID	ITEM_NAME	ITEM_DESCRIPTION
0	Keyboard	Computer keyboard
1	Monitor	Computer monitor

BUSINESS_ID	BUSINESS_NAME	BUSINESS_ADDRESS
0	Dell	
1	HP	

ITEM_ID	ITEM_NAME	ITEM_DESCRIPTION
0	Tastatur	Computertastatur
1	Monitor	Computermonitor

Table 5. *Sample Clone Database Localization:* **SUPPLY_en** *versus* **SUPPLY_de**

Instead of copying a table in Table Database Localization approach, a complete database was added in Clone Database Localization. Table 5 shows databases **SUPPLY_en** for English and **SUPPLY_de** for Geman in which exact contents were included. The only difference between the two databases is the database names. In Clone Database Localization, the number of localizable database grows in respect to the number of target languages. For instant, if the localization project aims for seven markets, there would be seven separate databases.

Workarounds for Database Localization

Chapter 2 stated that most specialized tools do not support database localization. This forces engineers to seek workarounds. Many workarounds may exist. This research investigated on one workaround for each database localization method.

Database localization consists of preparing the database to exclusively support localizable information. The following localization life cycle can be identified:

1. Analysis: This includes bid analysis, project schedule, project analysis, glossary and translation memory leverages.
2. Engineering and DTP (Desktop Publishing) Preparation: This includes text extraction, pseudo-translation and built environment testing.
3. Linguistic and Translation: Deliverables include text translation, translation reviews, terminology and translation memory maintenance.
4. Engineering and DTP (Desktop Publishing) Post Processing: This includes text insertion, localized testing environment and bug fixing.
5. Testing and QA: This phase includes testing environment, screenshots, linguistic/formatting/functional checks, bug reports.

Workarounds in this thesis focused on Step 1 - Analysis, Step 2 – Engineering Preparation and Step 4 - Engineering Post Processing. Regardless of what database structure is (e.g., Field, Row, Table or Clone Database Localization), the following procedures take place to exchange data between database technologies and specialized tools that do not support database localization.

a) Engineering Preparation for analysis or for actual project: query and export localizable strings into spreadsheet(s) for translation
b) Engineering Post Processing: import localized spreadsheet(s) into language separate tables which then be used to update database table(s)

The table below shows what process steps need to take place.

Database Localization Steps vs. Methods	Field	Row	Table	Clone
Analysis	X	X	X	X
Engineering & DTP Preparation	X	X	X	X
Engineering & DTP Post Processing	X	X	X	X

Table 6. Data Exchange in Database Localization Steps

Since most tools do not provide a filter for database format, a solution for Analysis and Engineering Preparation (Step 1 and 2) is to export all localizable contents into a format supported by the tools. In this case, Microsoft Office Excel 2003 format is supported by specialized tools and Microsoft Office Access 2003. Even though Microsoft Office Access 2003 was used in this research as a sample database type, these same workarounds can be used in other database platforms such as Oracle and MS SQL Server. The exported spreadsheet(s) can then be accepted by specialized tools during localization project creation. This project is localized in Step 3 - Linguistic and Translation. Once the translation is complete, localization tools process the project into its localized format, in this case, Microsoft Office Excel 2003 spreadsheet(s). Step 4 - Engineering Post Processing involves importing the localized spreadsheet(s) back into the database as separate table(s). The imported tables can be used to update the original database table(s).

To summarize, the following table compares different processes in specialized tools.

Localization Steps	Tools That Support Database	Tools That Do Not Support Database
Step 1 - Analysis	In specialized tool: ▪ define and configure database localization project ▪ analyze database localization project for word counts	In Microsoft Office Access 2003: ▪ restructure database or database table as needed ▪ query all localizable contents ▪ export query results to Microsoft Office Excel 2003 spreadsheet(s) In specialized tool, analyze exported spreadsheet(s) for word counts
Step 2 - Engineering Preparation	In specialized tool: ▪ define and configure database localization project ▪ analyze database localization project for word counts	In Microsoft Office Access 2003: ▪ restructure database or database table as needed ▪ query all localizable contents ▪ export query results to Microsoft Office Excel 2003 spreadsheet(s) In specialized tool, analyze exported spreadsheet(s) for word counts
Step 4 - Engineering Post Processing	In specialized tool, post process database localization project for localized database	In specialized tool: ▪ post process localized spreadsheet(s) In Microsoft Office Access 2003: ▪ import localized spreadsheet(s) into temporary localized table(s) ▪ update original table using localized table(s) ▪ clean up temporary localized tables

Table 7. Tools that Support vs. Tools that Do Not Support Database Localization

Table 7 showed that there are two to three additional steps required in specialized tool that do not support database localization. During Analysis and Engineering Preparation, the additional steps include 1) Select query creation and 2) Select query results' export to Microsoft

Office Excel 2003 spreadsheet(s). The goal of these additional steps is to convert a database to spreadsheet format which can be supported by specialized tools. During Engineering Post Processing, the additional steps are 1) localized spreadsheet(s) import, 2) Update query creation and operation and 3) temporary table clean up. With a couple of additional steps, localization issue in specialized tools that do not support database can be resolved.

Summary

In this chapter, we identified the following six research questions:

1. How important is database localization?
2. What are database localization specifications?
3. How do specialized tools work in supporting localization process?
4. What deficiencies, if any, exist in specialized tools and how do they impact database localization and process?
5. Do workarounds exist for overcoming tool deficiencies?
6. What is the procedure to prepare database content for localization in specialized tools which do not support database?

To answer these questions, a method was presented to test common database localization methods: Field, Row, Table and Clone Database Localization. This chapter also showed workarounds for tools that do not support database localization which engineers seek. The next chapter presents the results. Its purpose is to show the suitable solution for specialized tools that do not support database localization.

Chapter 4 – Data Analysis and Findings

In Chapter 3, we saw an approach to localizing a database in terms of fields, rows, tables, and cloning databases. Research questions and a method to answer them were also addressed. This chapter presents the results of four test cases: Field Database Localization in which we can analyze how values of the localized fields are updated in the same database row; the Row Database Localization test case exams how original database row is copied for each target language for localization; the Table Database Localization case evaluates the method of adding localized tables for each target language and the Clone Database Localization test case which involves copying the entire database instead of just the localized table. After presenting the results in terms of these four test cases, the next chapter will discuss how the thesis workarounds are applicable to localization engineers and database designers.

Test Case 1: Field Database Localization

Figure 2 presents table **ITEM** which was created for localization during Engineering Preparation. It shows the field name and data type. **ITEM_ID** shows item identification numbers. **ITEM_NAME** and **ITEM_DESCRIPTION** are localizable fields with language extensions – "**en**" for English and "**de**" for German. The original database only included **ITEM_ID**, **ITEM_NAME_en** and **ITEM_DESCRIPTION_en**. To accomplish Field Database Localization in which values of the localized fields are updated in the same row, **ITEM_NAME_de** and **ITEM_DESCRIPTION_de** were added into the table structure as German localized fields. In other words, **ITEM_NAME_de** and **ITEM_DESCRIPTION_de** fields were added so can be later updated with localized values.

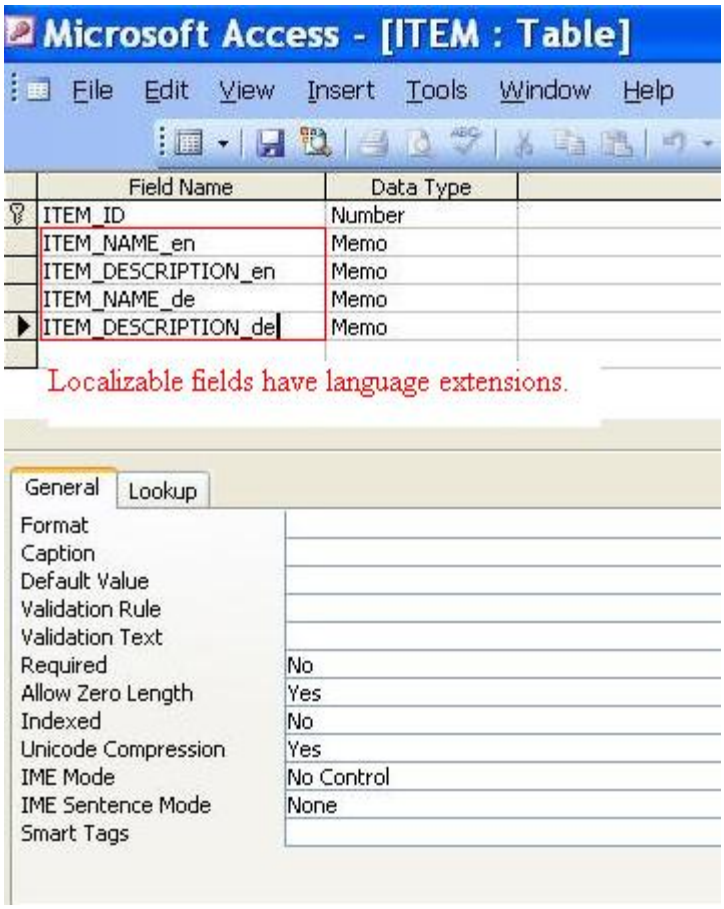


Figure 2. *Field Database Localization:* Table **ITEM**

Figure 3 identifies localizable fields to be exported for translation which were **ITEM_NAME_en** and **ITEM_DESCRIPTION_en**. During Engineering Preparation, German **ITEM_NAME_de** and **ITEM_DESCRIPTION_de** fields were blank and their values were updated during Engineering Post Processing.

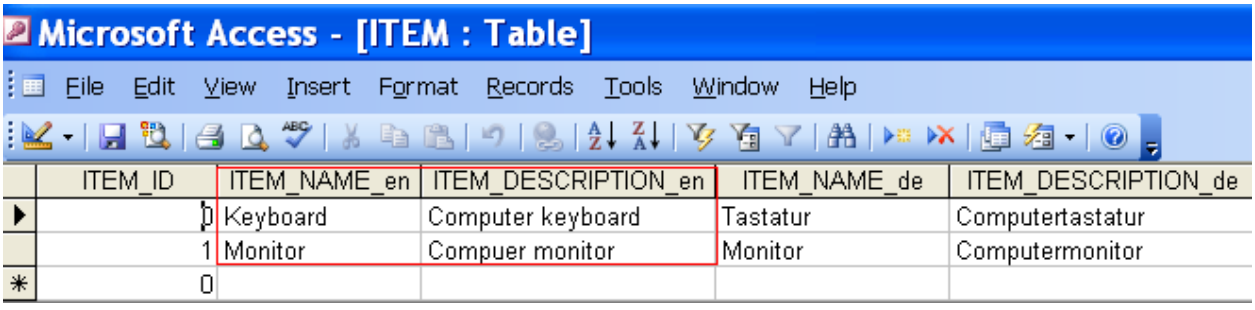


Figure 3. *Field Database Localization:* Fields of Table **ITEM**

To export localizable fields for translation, a **SELECT** query was created. This query selects not only **ITEM_NAME_en** and **ITEM_DESCRIPTION_en** but also **ITEM_ID**. Although **ITEM_ID** is not localizable, it was included in the query to identify one item from another. The ability to identify items was necessary during table update in Engineering Post Processing. The goal is to query only what was needed for localization process, not the complete table. After all, we only need to update the table with localized fields.

SQL statement of this **SELECT** query is the following:

```
SELECT ITEM.ITEM_ID, ITEM.ITEM_NAME_EN,
ITEM.ITEM_DESCRIPTION_EN
FROM ITEM;
```

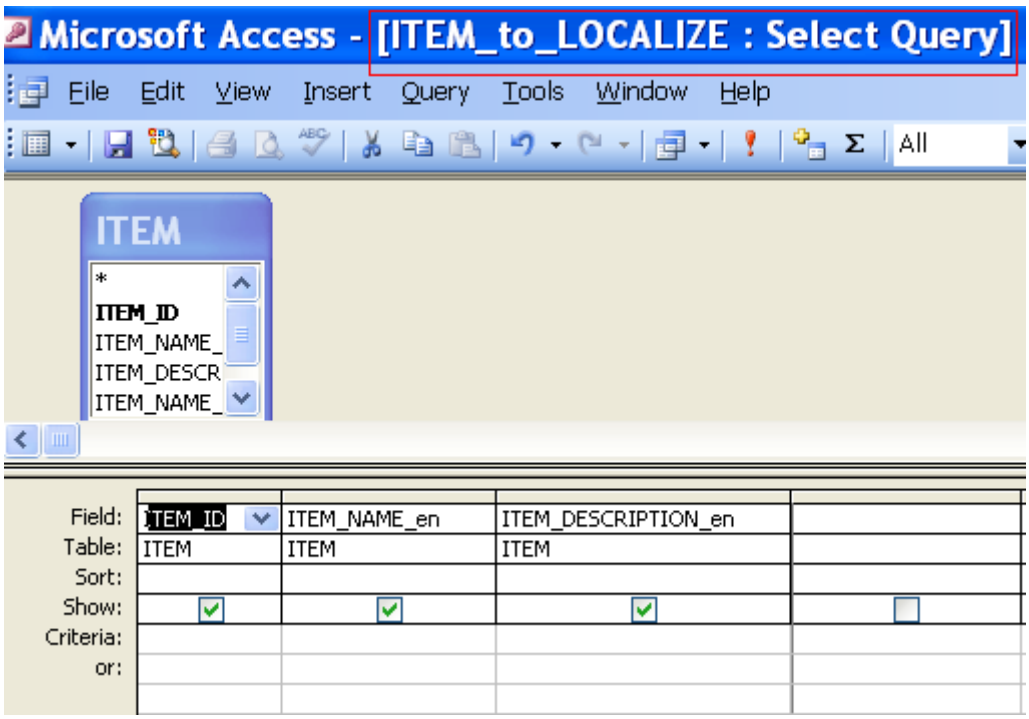


Figure 4. *Field Database Localization:* **SELECT** Query **ITEM_to_LOCALIZE**

As shown in Figure 5, the results from the **SELECT** query included not only the fields needed for translation, but also **ITEM_ID**, the item identification field. These results were

exported into Microsoft Excel spreadsheet (e.g. ITEM_en.xls). The advantage of creating and executing the **SELECT** query is that query results can be easily exported into a format that is accepted by tools which do not support database. This accepted format in this case is Microsoft Office Excel 2003 spreadsheet.

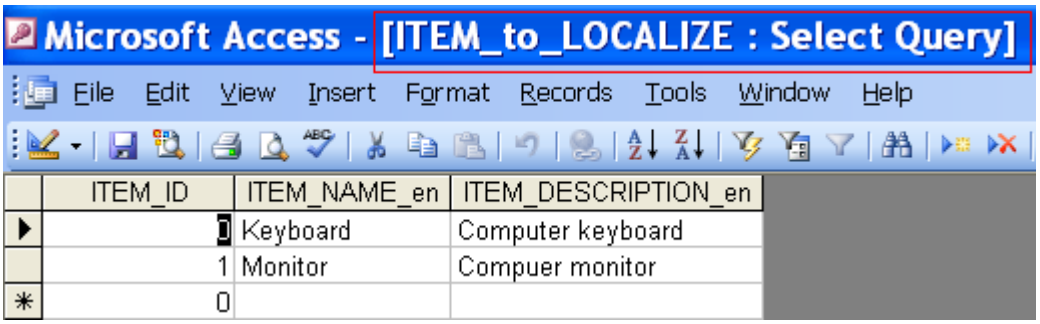


Figure 5. *Field Database Localization:* Results of Query **ITEM_to_LOCALIZE**

As previously mentioned **ITEM_ID** field contained identification keys which were needed during post-processing. For translation purposes, it was hidden or excluded as shown in Figure 6. Although the headers of **ITEM_NAME_en** and **ITEM_DESCRIPTION_en** fields are not localizable, they were not hidden. There is an option for localization tools to exclude these fields from translation.

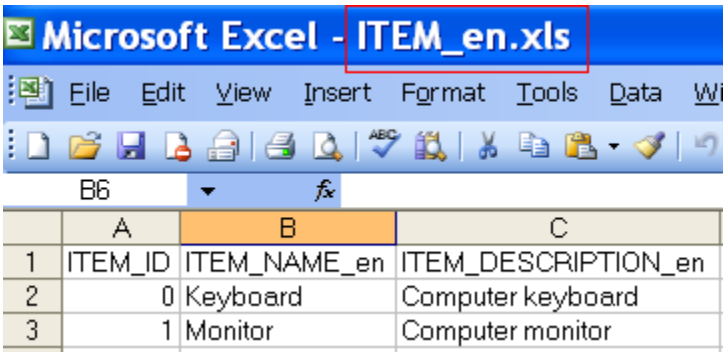


Figure 6. *Field Database Localization:* ITEM_en.xls with Hidden **ITEM_ID**

During Engineering Post Processing, the localized spreadsheet was imported into the original database as a separate table. This table was used to update the original database table. Once translation was complete, **ITEM_ID** field in the localized spreadsheet was unhidden. It is

necessary to rename the localized spreadsheet with a proper target language extension before importing it back into the original database. The reason for renaming was to differentiate between pre- and post-translation spreadsheets. This also meant that *Item_de.xls* has been localized with German translations. Thus, it was ready for database re-import.

Figures 7 and 8 show the original database with the imported localized table, **ITEM_de**, and its contents.

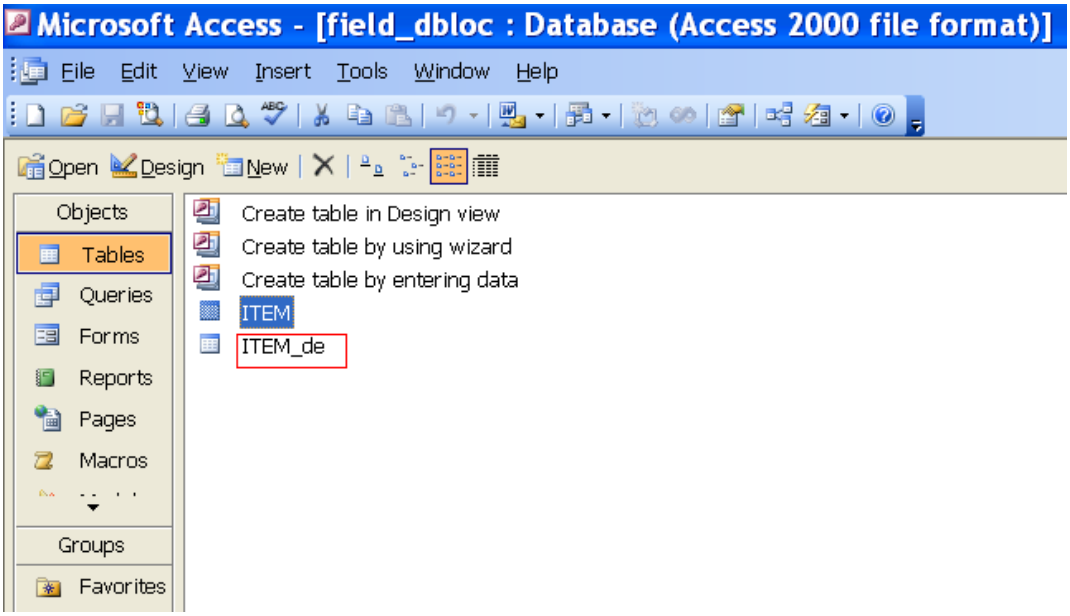


Figure 7. *Field Database Localization:* Database with Imported **ITEM_de**

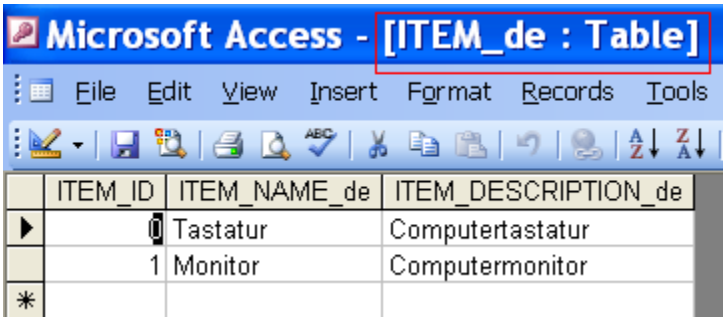


Figure 8. *Field Database Localization:* Imported **ITEM_de**

ITEM_de and **ITEM_DESCRIPTION_de** were updated with translations from the localized **ITEM_de** table. An Update query was created to instruct Microsoft Office Access

2003 to update **ITEM** table using corresponding values from **ITEM_de** table. To cross reference between these two tables, **ITEM_ID** field values were used as identification keys. For instant, **ITEM_ID** of '0' which previously contained 'Keyboard' and 'Computer keyboard' were updated with 'Tastatur' and 'Computertastatur' in order respectively.

SQL statement of this Update query is the following:

```
UPDATE ITEM INNER JOIN ITEM_DE ON ITEM.ITEM_ID =
ITEM_DE.ITEM_ID SET ITEM.ITEM_NAME_DE =
ITEM_DE!ITEM_NAME_DE, ITEM.ITEM_DESCRIPTION_DE =
ITEM_DE!ITEM_DESCRIPTION_DE
WHERE ((([ITEM]![ITEM_ID])=[ITEM_DE]![ITEM_ID]));
```

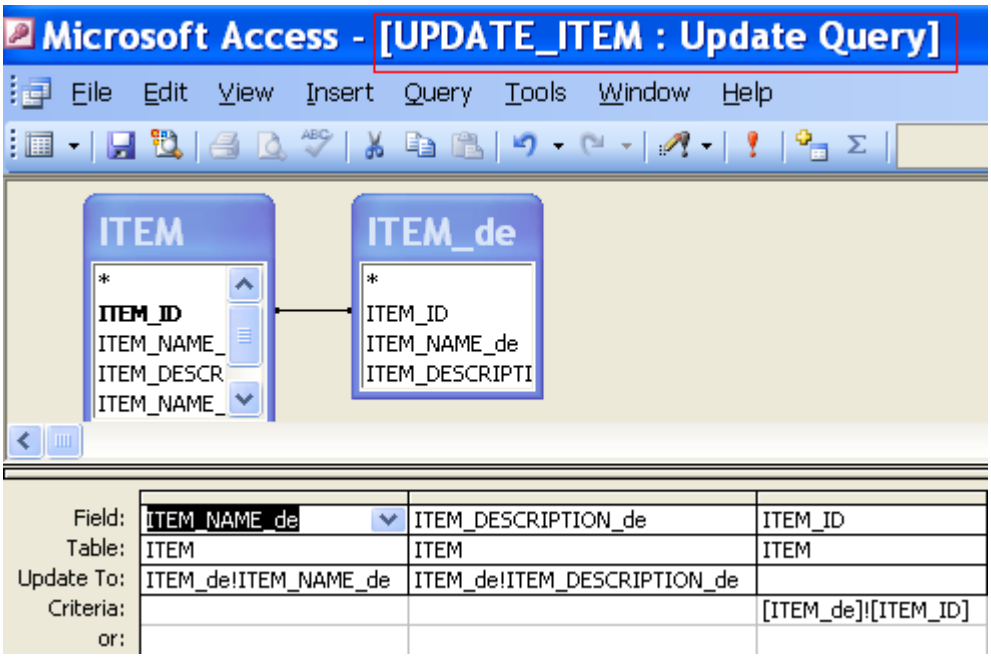
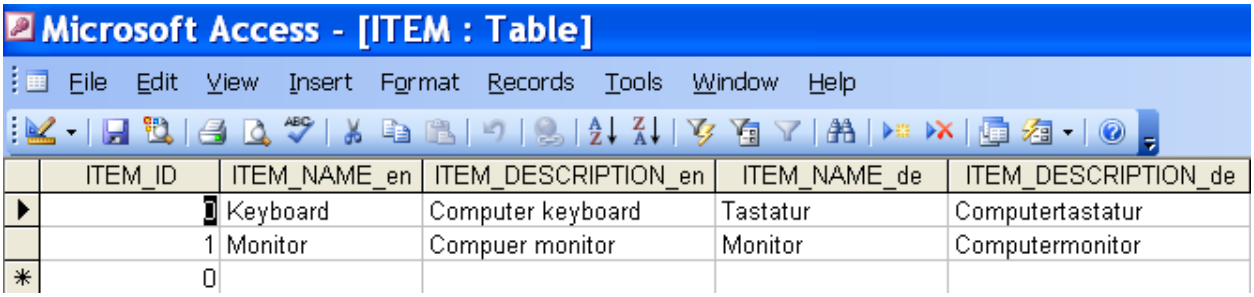


Figure 9. *Field Database Localization:* **UPDATE** Query **UPDATE_ITEM**

After running the Update query, the updated **ITEM** table contained the following data as shown in Figure 10.



	ITEM_ID	ITEM_NAME_en	ITEM_DESCRIPTION_en	ITEM_NAME_de	ITEM_DESCRIPTION_de
▶	0	Keyboard	Computer keyboard	Tastatur	Computertastatur
	1	Monitor	Compuer monitor	Monitor	Computermonitor
*	0				

Figure 10. *Field Database Localization:* Updated Table **ITEM**

Table **ITEM** was successfully localized. This table contains item identification keys, item names and item descriptions. For the localizable fields of **ITEM_NAME** and **ITEM_DESCRIPTION**, all available languages of English and German were included.

Test Case 2: Row Database Localization

Figure 11 shows the structure of table **ITEM** in which the original database row was copied for each target language for localization. **LANG_ID** was added to classify the language identifications so therefore **ITEM_NAME** and **ITEM_DESCRIPTION** did not include language extensions. Both **ITEM_ID** and **LANG_ID** were primary keys for **ITEM** table since there would be more than one entry with the same **ITEM_ID** and **LANG_ID**. **LANG_ID** can either be character or number coded. For instant, with character codes which were used in this test case, **LANG_ID** would include values such as '*en*' or '*de*' for English or German, respectively. On the other hand, if number codes are used, **LANG_ID** would include values such as '*1033*' or '*1031*' for English or German in respective order.

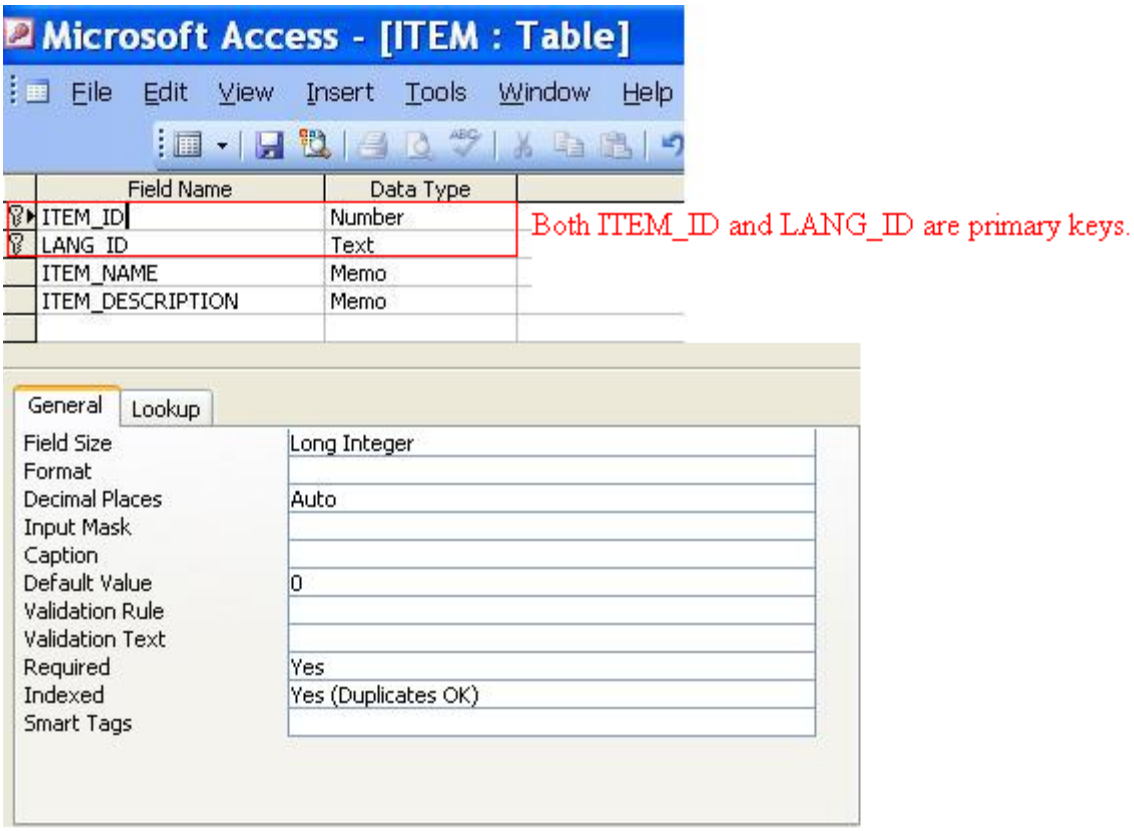


Figure 11. *Field Database Localization:* Table **ITEM**

The following table shows the localizable fields of **ITEM_NAME** and **ITEM_DESCRIPTION**. During Engineering Preparation, German **ITEM_NAME** and **ITEM_DESCRIPTION** fields did not exist. These fields were appended during Engineering Post Processing.

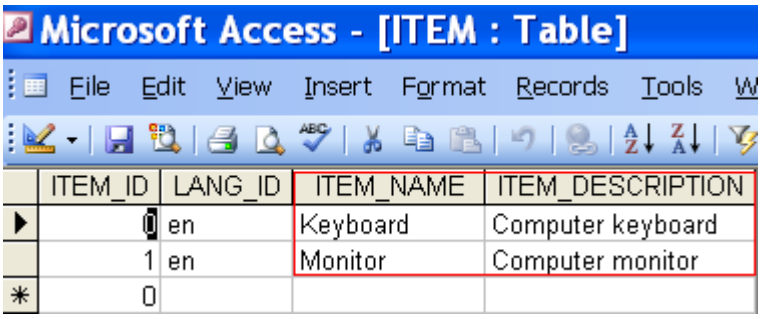


Figure 12. *Row Database Localization:* Fields of Table **ITEM**

Similar to the query created in Field Database Localization, this **SELECT** query also defined what to be exported for translation. In addition to localizable **ITEM_NAME** and **ITEM_DESCRIPTION** fields, **LANG_ID** field was also exported. **LANG_ID** is localizable and should be exported for translation. For instant, data in **LANG_ID** field was localized from '*en*' to '*de*' for German. Although **ITEM_ID** is not localizable, it was included in the query to identify one item from another. The ability to identify items is necessary during table update in Engineering Post Processing. The goal is to query only what was needed for localization process, not the complete table.

SQL statement of this **SELECT** query was the following:

```
SELECT ITEM.ITEM_ID, ITEM.LANG_ID, ITEM.ITEM_NAME,
ITEM.ITEM_DESCRIPTION
FROM ITEM;
```

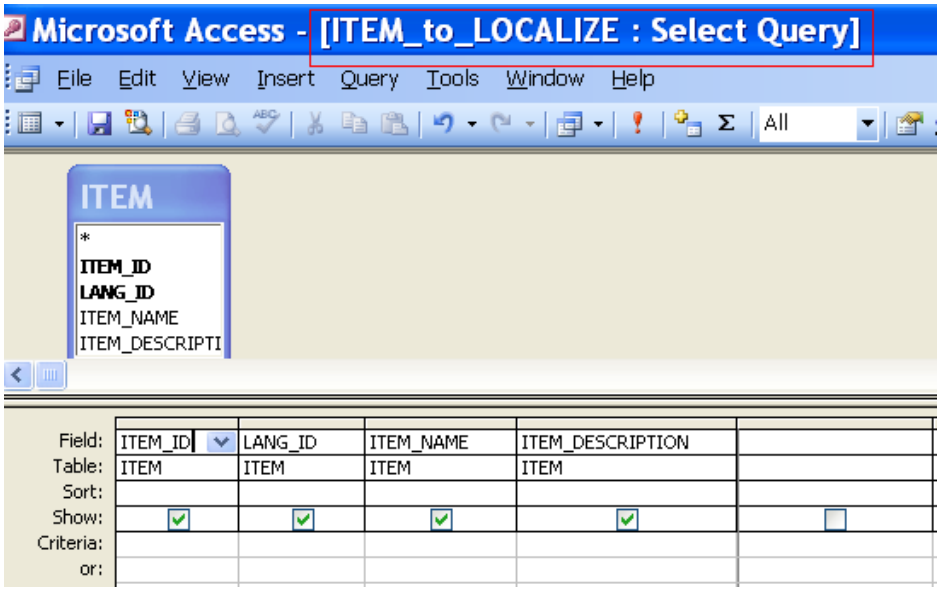


Figure 13. *Row Database Localization:* **SELECT** Query **ITEM_to_LOCALIZE**

The results from the **SELECT** query in this test case were similar to the Field Database Localization test case with the addition of **LANG_ID** field in the exported spreadsheet. As

shown in Figure 14, the exported spreadsheet included **ITEM_ID**, **LANG_ID**, **ITEM_NAME** and **ITEM_DESCRIPTION**.

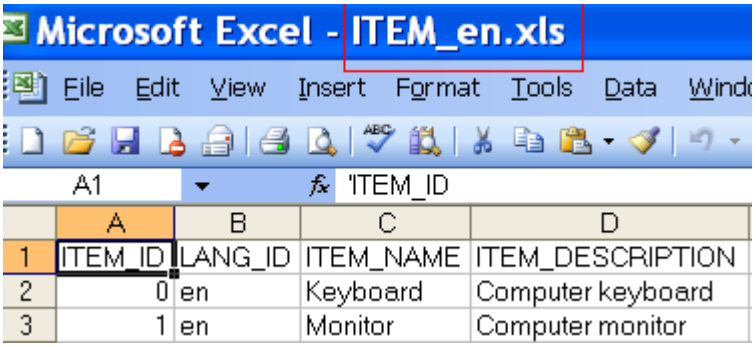


Figure 14. *Row Database Docalization:* Exported Excel Spreadsheet ITEM_en.xls

Similar to what was done in Field Database Localization, both **ITEM_ID** and **LANG_ID** were hidden for translation purposes. The export spreadsheet was sent to translation.

During Engineering Post Processing, the localized spreadsheet was imported back into the original database as a separate table. This table was used to update the original database table. After translation was complete, both **ITEM_ID** and **LANG_ID** were unhidden. The localized spreadsheet and in turn the imported **ITEM_de** table included **LANG_ID** field as shown in Figures 15 and 16.

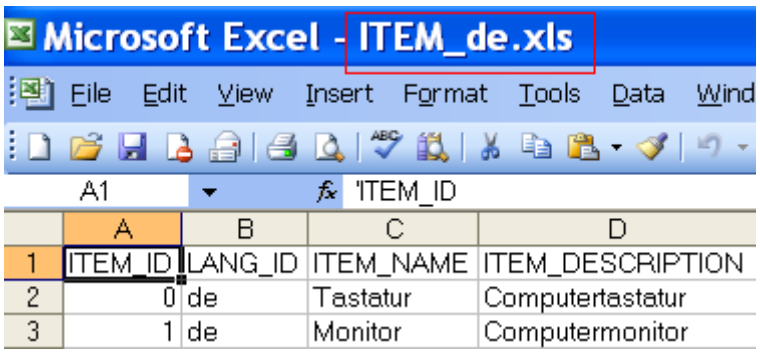


Figure 15. *Row Database Localization:* Localized Spreadsheet **ITEM_de**

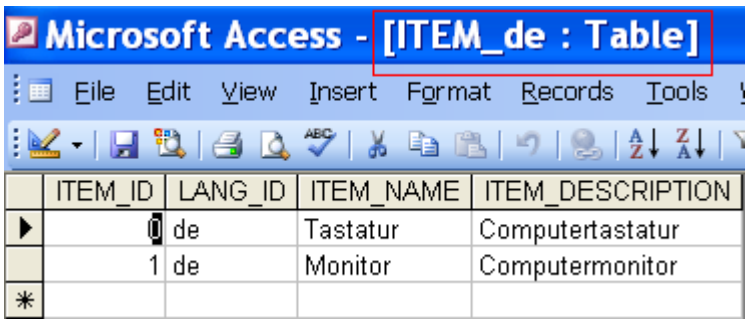


Figure 16. *Row Database Localization:* Imported Table **ITEM_de**

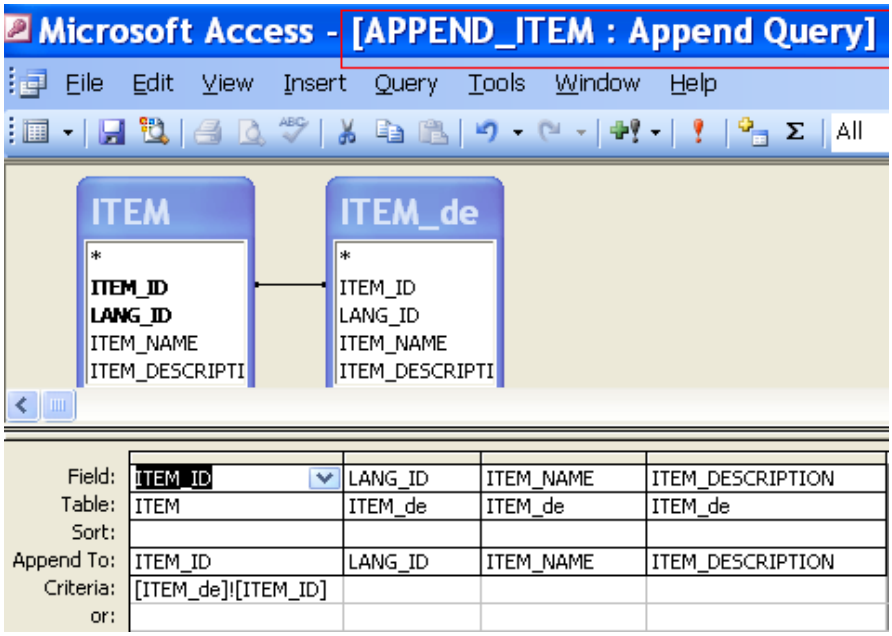


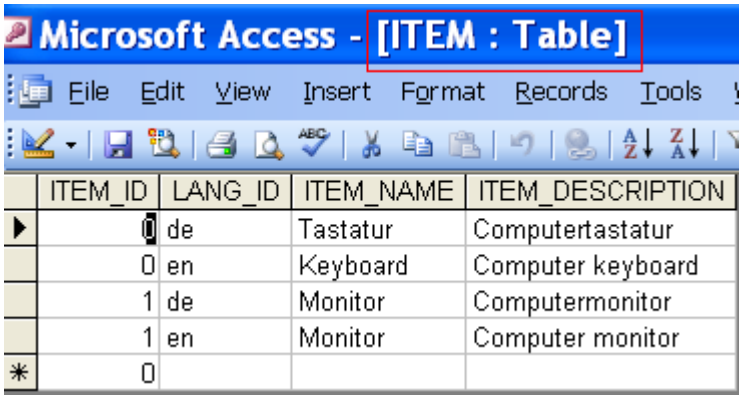
Figure 17. *Row Database Localization:* **APPEND** Query **APPEND_ITEM**

When executing the **APPEND** query, **ITEM** table was appended with German translations from the localized **ITEM_de** table. The **APPEND** query instructed Microsoft Office Access 2003 to insert *Item* table using corresponding values of **LANG_ID, ITEM_NAME** and **ITEM_DESCRIPTION** from **ITEM_de** table. To cross reference between these two tables, **ITEM_ID** field values were used as identification keys.

SQL statement of this **APPEND** query was the following:

```
INSERT INTO ITEM ( ITEM_ID, LANG_ID, ITEM_NAME,
ITEM_DESCRIPTION )
SELECT ITEM.ITEM_ID, ITEM_DE.LANG_ID, ITEM_DE.ITEM_NAME,
ITEM_DE.ITEM_DESCRIPTION
FROM ITEM INNER JOIN ITEM_DE ON ITEM.ITEM_ID =
ITEM_DE.ITEM_ID
WHERE ((([ITEM]![ITEM_ID])=[ITEM_DE]![ITEM_ID]));
```

ITEM table was updated with the localized fields as shown in Figure 18.



ITEM_ID	LANG_ID	ITEM_NAME	ITEM_DESCRIPTION
0	de	Tastatur	Computertastatur
0	en	Keyboard	Computer keyboard
1	de	Monitor	Computermonitor
1	en	Monitor	Computer monitor
0			

Figure 18. *Row Database Localization:* Updated Table **ITEM**

Table **ITEM** was successfully localized. This table included item identifications, language identifications, item names and item descriptions. For the localizable fields of **ITEM_NAME** and **ITEM_DESCRIPTION**, all available languages of English and German were included.

Test Case 3: Table Database Localization

In this test case, localized tables were added into the original database. The localized tables contained primary key and the localized fields. The localized tables' name included target language identifications, for example, '**ITEM_de**'. Table Database Localization method used the

same engineering prepping procedure listed in Row Database Localization to export **ITEM_ID**, **ITEM_NAME** and **ITEM_DESCRIPTION** fields for translation. For Engineering Post Processing, the task only required importing the localized spreadsheet into a separate table, '**ITEM_de**', and the process was complete. No table update was required so an **APPEND** query was unnecessary in the Table Database Localization test case.

The localized table from this test case is shown in Figure 19. The structure of **ITEM_de** table was the same as the originals with item identifications, item names and item descriptions. However, for the localizable fields of **ITEM_NAME** and **ITEM_DESCRIPTION**, only values in German were included.

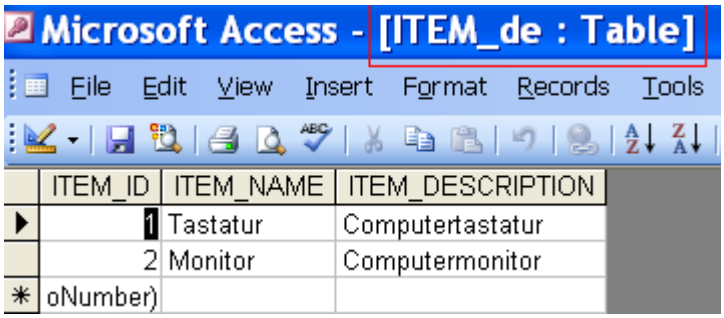


Figure 19. *Table Database Localization:* Localized Table **ITEM_de**

Test Case 4: Clone Database Localization

When localizing a database using a cloning method, a duplicate or an exact copy of the original database is created using the same table structure and data. The only difference was the cloned database name had the target language identifications, '**SUPPLY_de**'. The Clone Database Localization method used the same procedure listed in Row Database Localization test case, without **LANG_ID** field in the database structure.

Figure 20 shows the cloning method used in this effort. The structure of **SUPPLY_de** database was the same as the originals with **ITEM** table. However, for the localizable fields of **ITEM_NAME** and **ITEM_DESCRIPTION**, only values in German were included.

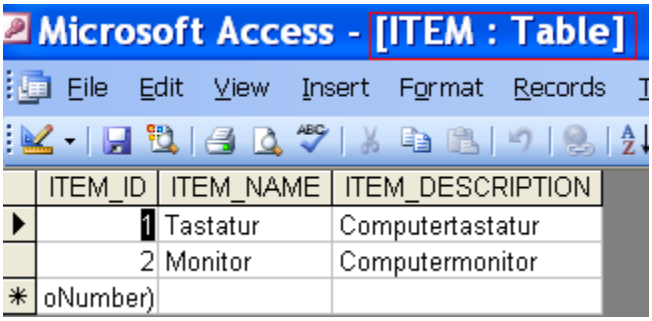


Figure 20. *Clone Database Localization:* Localized Database **SUPPLY_de**

Summary

Field, Row, Table and Clone are different methods that were tested to support database localization. The findings show that database localization is possible in an environment with few database localization tools. There are advantageous and disadvantages from each method. While all methods except Clone Database Localization require changes in database structure, Field Database Localization method also eliminates data redundancy. In Clone Database Localization, there is no need to change the database structure but localized databases are independently kept from the original database. It also provides answers to the research questions regarding localization posed in Chapter 3 and discusses if the results of the workarounds are applicable to localization engineers and database designers.

Chapter 5 – Discussion and Conclusion

This researcher believes that this project contributes to academic literature which is weak when it comes to localization of databases. The findings in Chapter 4 help localization engineers and designers understand database localization methods. They suggest engineering workarounds to overcome the lack of database localization tools. The findings also highlight the decisions that localization engineers and designers must make in deciding on localization methods. Since there are advantageous and disadvantages from each method, localization engineers and designers should consider the best fit for their current localization project.

In designing the research, a method was developed to answer six questions about localization. This method introduced an approach to localizing a database. Localization is a matter of localizing fields, rows, tables and cloning the entire database. Test cases shown in Chapter 4 corresponded to Field, Row, Table and Clone Database Localization methods. The test case proved that database localization can be successfully carried out without specialized tools. Engineers should follow the procedures described in Chapter 4 to prepare database contents for localization. Engineers also need to put into considerations of what type of database localization method can be used appropriately.

Yes. Database localization is important because so many modern applications are based on databases. When you localize this information, new markets are opened and sales can increase. Database localization specifications are based on Field, Row, Table and Clone Database Localization methods. There are specialized tools which help engineers in preparing files for localization. These tools separate localizable texts from codes during Engineering Preparation and merge contents back during Engineering Post Processing. In fact, these tools have their weaknesses which affect localization quality and schedule. Most localization tools

lack support for database file formats. As a result, manual engineering steps need to take place which are inefficient and ineffective. Workarounds to prepare database content for localization were shown in Chapter 4 to resolve specialized tools' database deficiency.

One of the noticeable features in today's business and information technology is globalization. "The world is flat" (Friedman, 2005). The interweaving of business markets and information technology has shrunk the world. Thanks to the Internet, for example, each of us can reach around the world farther, faster and cheaper than ever before. And for a growing number of US companies, globalization represents a greater share of total sales which can help overcome times of domestic slowdowns.

This globalization in turn has created the demand for the localization of software such as databases. Gone are the days when a database vendor can just develop and market a product in English. Product groups have to think about localization. Globalization cycle consists of internationalization and localization processes. Internationalization process ensures product planning and product implementing that allow localization. By translating texts and also adding locale specific components, localization processes adapt internationalized software for specific languages or regions.

This project was designed to fill a gap in the literature and tools. However, it is only a start and this researcher encourages others studies to be carried out. The literature would surely benefit from studies of using automation for these workarounds. The localization process can be more efficient and effective with automation. For instant, query process for contents export during Engineering Preparation and update process localizable tables during Engineering Post Processing can both be automated. Considering there might be more than one language to be localized, the automation should be available per target language. If we could save five minutes

per engineering process per language with automation, we actually save fifty minutes when localizing ten languages.

References

Abufardeh, S., & Magel, K. (2009). Culturalization of software architecture: Issues and challenges. *2008 International conference on computer science and software engineering*, csse(2), 436-439. Retrieved from http://doi.ieeecomputersociety.org/10.1109/CSSE.2008.1414

Baik, J. (2000). The effects of CASE tools on software development effort (Doctoral dissertation). University of Southern California, Los Angeles, CA.

Begel, A., Nagappan, N. (2008). Global software development: Who does it? *ICGSE Proceedings of the 2008 IEEE international conference on global software engineering*, 195-199. doi: 10.1109/ICGSE.2008.17

Chroust, G. (2007). Software like a courteous butler - Issues of localization under cultural diversity. *Proceedings of the 51st annual meeting of the ISSS*. Retrieved from http://journals.isss.org/index.php/proceedings51st/article/view/569/275

CodeBetter.Com. (n.d.). Software development life cycle models. Retrieved from http://codebetter.com/blogs/raymond.lewallen/archive/2005/07/13/129114.aspx

Esselink, B. (2000). *A practical guide to localization*. Philadelphia, PA: John Benjamins B.V.

Evers, V. (1998, August). Cross-cultural understanding of metaphors in interface design. *Cultural attitudes towards technology and communication*. Retrieved from http://www.it.murdoch.edu.au/~sudweeks/catac98/pdf/22_evers.pdf

Friedman, T. L. (July 23, 2007). *The world is flat: A brief history of the twenty-first century*. (Further updated and expanded: Release 3.0). New York: Farrar, Straus and Giroux.

Lingobit Localization On Demand. (n.d.). Database localization. Retrieved from http://www.lingobit.com/solutions/articles/database-localization.htm

Microsoft Corporation. (September 1, 2009). Corporation annual report 2009. Retrieved from http://www.microsoft.com/msft/reports/ar09/10k_dl_dow.html

Oracle Technology Network. (May 31, 2009). Corporation annual report 2009. Retrieved from http://www.oracle.com/corporate/investor_relations/index.html

Rombach, H. D. (2007). Software development and globalization. *Lecture notes in computer science*, vol. 4589, 1-1. doi: 10.1007/978-3-540-73460-4_1

SDL International. (n.d.). New! SDL Passolo 2009. Retrieved from http://www.sdl.com/en/sites/sdl-passolo/about/

Sisulizer. (n.d.). Sisulizer is the database localization specialist. Retrieved from http://www.sisulizer.com/localization/software/server-desktop-database.shtml

Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., … Zdonik, S. (2005). C-Store: A Column-oriented DBMS. *National science foundation*, IIS-0086057 & ISS-0325525.

Sun Developer Network. (n.d.). I18n in Software Design, Architecture and Implementation. Retrieved from http://developers.sun.com/dev/gadc/technicalpublications/articles/archi18n.html

Sybase. (February 27, 2009). Annual report and proxy statement for the 2009 Annual Meeting of Stockholders. Retrieved from http://investor.sybase.com

Texin, T. (2005). Translation, localization, globalization, internationalization tools. Retrieved from http://www.i18nguy.com/TranslationTools.html#top

The Localization Industry Standards Association. (n.d.). What is globalization? Retrieved from http://www.lisa.org/What-Is-Globalizatio.48.0.html?&no_cache=1&sword_list[]=globalization

The Localization Industry Standards Association. (n.d.). Why globalized? Is this end with an "ed?" Retrieved from http://www.lisa.org/Why-Globalize.50.0.html

Yeo, A. W. (1996). Software Internationalization and Localization. *OZCHI '96, 6th Australian conference on computer-human interaction*, ISBN 0-8186-7525-X.

Yeo, A. W. (2001). Global-software development lifecycle: An exploratory study. *Proceedings of the SIGCHI conference on human factors in computing systems*, 1-58113-327-8, 104-111. doi: 10.1145/365024.365060

Yeo, A., & Barbour, R. H. (1996). Software for the rest of the world. *Computer science working papers* 96/02. Retrieved from http://hdl.handle.net/10289/1155

Yuri, D. (March 28, 2007). Go global with software globalization. Retrieved from http://forums.guestbook.com.tw/thread-47582-1-1.html