THE MINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

On the manuscript
right
UDC 681.3.069

ERGASHEV ULUGBEK ORIFJON UGLI

DEVELOPMENT OF A METHOD FOR DATA MIGRATION FROM XML REPOSITORIES INTO RELATIONAL DATABASE

Specialization: 5A330501 – Computer engineering (Computer systems design)

Dissertation for getting master degree

Supervisor

c.e.s. Associate Professor

Kogay V.N

Tashkent-2016

Аннотация

Данная диссертационная работа посвящена разработке метода миграции данных из XML в реляционную базу данных. В данной работе были рассмотрены теоретические основы переноса данных и методы миграции данных, определены проблемы и выработаны требования для реализации метода, приведена постановка задачи и алгоритм миграции данных решающий определенную проблему.

Был разработан и апробирован метод для миграции данных из XML в реляционную базу данных на основе схем ContextMap.

Научной новизной диссертационной работы является использование моделей на основе нотации ContextMap при решении проблемы синхронизации данных. Показана практическая ценность предлагаемого метода при разработке корпоративных распределенных информационных систем.

Abstract

This thesis is dedicated to the development of a method for data migration from XML repositories into a relational database. In this work, theoretical basis for data migration and data migration techniques were considered, the issues and requirements for the implementation of the method were identified, problem statement and data migration algorithm that solves particular problem were elucidated.

The method for data migration from XML repositories into a relational database were developed based ContextMap schemas and tested.

The scientific novelty of the thesis is the use of models based on ContextMap notation in resolving the problem of data synchronization. The practical value of the proposed method in the development of enterprise distributed information systems was shown.

TABLE OF CONTENTS

INTRODUCTION

Modern world in practically all spheres of life use computers. For each application there is their software that uses different ways to store the necessary data. Increase the degree of automation of vital processes leads to new business processes, among which you can select a group of business scenarios affecting change, add, delete existing data. Such a group could include mergers and acquisitions, modernize legacy systems, consolidating applications, ERP and CRM systems or upgrading, implementation of management systems of normative-reference data, business process outsourcing. In this regard, a problem regarding data migration.

The solution to this problem uneasy. Data migration is a complex process that involves risks. At the moment, the solution to this issue is the urgent task for the IT-world. At the current time it is research and development of options for implementing tools to migrate data. It should be noted that a separate branch of research is devoted to the reasons of failure data migration projects. Some of them are lack of methodology, unrealistic proportions projects, improper use of tools, lack of attention to data quality, lack or absence of the necessary expertise.

Some examples of data migration may differ, that leads to the development of different methodologies. One of the reasons for the classification of such processes is a way of storing data. The formation of another class is associated with the cause of the need to migrate data. It's worth noting that even in the same class, there is no universal approach to solving this problem, as well as a separate project data migration often refer to multiple classes, the complexity of its implementation is increased.

At the current moment to store data often use relational DBMS. Therefore, consideration of this project type is one of the most urgent areas of research. Widespread motif of data migration is releasing a new version of the software. Perspective in such cases can be illustrated by the following example.

Actuality. Today, the most widely distributed systems are information systems. In this regard, these systems are attracting the attention of researchers.

In distributed systems, the issue of data transfer of various formats between subsystems are becoming relevance. Data migration task between XML document and a relational database are becoming attractive, because the vast amount of data, particularly at the grass-roots level, are presented in tables Excel, based on XML.

Particular relevance this work acquires connection with the decision of the President of the Republic of Uzbekistan "On measures for further implementation and development of modern information and communication technologies" [1]. Given the rapid growth of information systems and the complexity of the services in the web space, the question of their effective functioning and providing rapid responses to user requests are provided. Obvious examples of such systems demand in our country and important for its development, we can see in the Resolution of the Cabinet of Ministers "On Measures for the organization of the Centre of the" electronic government "[2], and the ruling PP-2158 President of the Republic Uzbekistan from April 3, 2014 "On measures to further the implementation of information and communication technologies in the real economy"[3]. This kind of information systems are complex, not only in implementation but also in the design, they are being given a wide range of services working in a real time with a huge number of citizens of our Republic. Questions of speed of such information systems are major and important.

Thus, the example shown illustrates several causes of migration data stored in relational database systems:

- changing the schemas of databases;
- go to another RDBMS.

At this point the above problem is solved in a variety of ways. At the time when the study began the task, the main way of data transfer scenarios were created for each individual client in accordance with the analysis of the source and target databases. Thus, the creation of a universal tool to address this problem, remains a challenge.

Object of research. After moving data, synchronization problem is not solved in any product. This work will be devoted to solving this problem.

The aim of this thesis is the design and development of a method to migrate data from XML in a relational database.

Tasks are:

- Conduct an analysis of the most appropriate methods
- Learn the basic problem of data synchronization
- Develop a data migration method
- Conduct a comparative analysis of the developed method with existing

The novelty of the research conducted in the master's thesis, is determined by the relevance of the chosen theme, and practical challenges facing IT companies. In particular, on the basis of an analysis of existing software products on the market and methods of data migration, the author proposes the use of notation ContextMap in solving the problem of data synchronization.

Practical value. This method can be used to synchronize data in distributed systems using heterogeneous repository (relational or XML). For example, it can be used in systems of interdepartmental information sharing.

The first chapter provides an overview of the investigated subject. The chapter discusses the main directions in research on migration data that covers aspects such as terminology, types of data migration, the General principles of the migration data. The second chapter describes and assesses priority side data synchronization. And several algorithms to find proximity measure differences between old and new data. A model based on the notation ContextMap. The third chapter shows data synchronization method based on diagramming ContextMap, and compare them to identify changes between XML document and relational database in the target table. An analysis of the proposed method with a method full transfer tables. Identified priority part of the developed method such as the relatively rapid implementation data migration and

conservation of resources of your computer or computer nodes. Theoretical and empirical results were formulated in the conclusion.

CHAPTER I. ANALYZE OF PROBLEM AREA

Reliable storage of information is one of the most urgent tasks for every company and organization. Caring for a full protection of your data is a prerequisite for survival and development. While it is often necessary to move the data to another company, new equipment or new virtual Wednesday. And when it comes to this kind of data migration, many IT managers are experiencing great difficulties, because it is very laborious, time-consuming and resource-intensive operation that is associated with the risk of losing information.

Number of relevant information throughout the world increases annually at 60% (according to IDC for 2008 year). In doing so, IT managers must ensure data redundancy in the event of natural disasters, the execution of new regulatory laws. They must also ensure accessibility and mobility data and applications, the consolidation of the old array with the new systems, and perform many other duties. All this makes the approach to strategic data migration, you can even say supporting, IT management.

Every year companies must, in one way or another, move up to a quarter of all operational data (according to Enterprise Strategy Group). Data migration is an endless cycle where work is in full swing. As data center infrastructure develops, IT managers may need an effective strategy that would support the work of the data centers, providing high availability of applications while not increasing, but rather by reducing unnecessary data storage costs. A good strategy is always the key to success.

The need for migration, or more simply, moving data often involves fairly routine works, is a replacement for servers and hardware for data storage, relocation, consolidation, update leasing contracts, etc.

Usually the company carries out data migration when implementing virtualized among data management systems by moving infrequently used data to less expensive storage. The problem is that traditional methods of migration inherently disruptive, especially if the process involves first moving data from one device to another, the SAN, and then-and migrate the applications themselves. The actual data movement is typically performed on weekends or at night time, and the biggest problem is the paralysis of the applications. This is one of the explicit indication that data migration methods have to change.

A list of the major problems faced by IT managers when migrating data, includes: long plain or decrease application performance, loss and data corruption, as well as compatibility issues. According to some studies only every fourth IT specialist raised no problems related to migration data [9].

Data migration is usually a long and expensive process, which is very important for those businesses where it requires constantly. Although the majority of IT professionals believed that data migration is rarely necessary, but about 40% of cases, data migration takes place once a month or more frequently. In most cases, when you migrate the weekend IT professionals try to avoid application downtime and other similar risks. And those tasks is expensive and high overtime payouts for employees and, often, what is even more frustrating, disgruntled staff.

Data migration is a standard task for modern data centers, and IT managers desperately need a strategic plan that would combine the existing computational requirements among them and flexibility for further improvement in the future.

Data transport model

Despite the fact that there are many ways to migrate data to select two basic: moving data online and offline data movement.

Moving data online

This method is that at a time when migration is carried out directly by the data servers and applications online. This mode is typically used for mission-critical applications with high availability, such as Oracle, Exchange, E-Commerce and other databases and high-level billing systems-those whose work requires 24 hours a day. Online application mechanism to move data is embedded in the data stream, with the migration of old data or write new data does not violate and not much slow down applications. The program then data migration creates a new logical unit number (LUN, disk), which brings together old and new data. And finally, the program deletes the data migration old LUN or moves it to the archival access levels and security.

Moving data offline

This is an easier way to migrate is used when applications can be temporarily stopped and movement can occur offline. The main task of this method is to move the data as quickly as possible to the desired position and ensure that displaced data were completely correct. Then you need to restart applications, test their operability, as well as the condition of displaced data. Then, as in the previous method, you must remove the old LUN or move it to archive access levels and security. Despite the fact that data migration is carried out according to one of these two models, there are many ways to direct the implementation of these processes. Here are five of the most popular of these ways:

-host-based (the method that executes on the host) is the most popular method in small and medium business, it's great for a small amount of data migration from one point to another. The advantage of this method is that it is very simple, inexpensive, since mostly based on already built into the OS. However, host-based solutions generally do not feature high performance as forced to simultaneously move packets of data between the host and array first, then between the host and the second array. This solution is also a good scalability. Problems can be alleviated through the use of host-based solutions

from third-party manufacturers, but those solutions do not come cheap, the price often may amount to tens of thousands of dollars;

-storage virtualization-storage virtualization while introducing a host accesses the proxy device (such as a switch or router) that provides host virtual LUN and displays the virtual logical volumes to the physical LUN. Virtualization, as a rule, it is nothing less than a permanent change in the architecture of the servers and then moving the metadata in a virtualized Wednesday. This method of data migration "unloads" Server and usually can support different models and storage systems from different manufacturers. This solution reduces the need for frequent migration moving data and has much more than host-based solutions. Of the shortcomings of the method, you can note the excessive impact on applications, which have to share the same bandwidth. In addition, this solution is more complicated than host-based, and much more expensive-virtualization requires the use of special equipment and software, the value of which can go up to hundreds of thousands of dollars;

-switch-based (based on the switch) is a method of data migration on the switch uses virtual LUNs or ports in the same way as when using the method of storage virtualization, but then returns control to the host without storing the metadata. This allows you to take such a decision open, unlike the private method of virtualization. Based on the switch data migration solution has several advantages.

Firstly, it supports a large number of different system components: hosts, operating systems, applications, and storage.

Secondly, the decision does not require the use of host agents.

Finally, this decision differs in some cases even higher performance and scalability.

The main disadvantages of this method is its high price, some performance impact on the storage system, the opportunity cost. Infrastructure built on switches "Director" class can cost many hundreds of thousands of dollars. This solution is very complex from a technical point of

view, and require highly qualified personnel for both the implementation and subsequent support in working condition. In addition, the use of this data movement solutions can affect application performance and reduce the rate of continuous data availability. And finally, software for implementing such decisions will also cost very expensive;

-router-based (based on the router)-this method of data migration is the same as Switch-based method, and has the same advantages. The router adds additional flexibility in supporting and managing the used protocols and does not affect the performance of the infrastructure, so as not to transmits data through the switch. In addition, using this method reduces the cost of hardware and software to move data. The method is well suited for both large companies and small and medium-sized businesses.

-array-based (based on the storage controller)-this method is private data migration solution, implemented on a particular storage device family, and generally does not support devices from other manufacturers. Migration data from the array to the array over the network, and looks like a direct copy volumes. manufacturers arrays provide the best performance for moving data type array-array (array-to-array). However, these solutions are proprietary, does not support third-party storage devices and often only support moving into an array, but not from the array. In addition, the price only software for such decisions can often be very expensive.

As you can see, the best solution for data migration is a method based on routers. Slightly behind him and a method based on the use of switches. But it should be borne in mind that in any case the data migration method you choose should be based primarily on the specific conditions and offer the most effective strategy in each case [15].

Data migration plan

Go back directly to the building plan data migration. The first task for IT managers when implementing data movement is to plan the process so that its

implementation does not affect system performance, or at least reduce this impact to the minimum. The second challenge is to ensure the integrity and availability of data. Thirdly-to control access to data.

Not all data you can copy or move the same way, and each data type can have a different impact on the application. IT managers must gain a full understanding of the structure and functioning of its database, to predict the impact of completed migration before proceeding directly to moving data.

Traditional data migration plan consists of six tasks.

1. the definition of the objectives of the migration data. As in the execution of any project, you have to implement it, and how you will measure success. You need to start with drawing up objectives: technical, business, budget, etc.

2. Check calculations. Now that you have an outline of the objectives and criteria for evaluating success, you need to check the settings. What is the size of the data? How much has time to move these data? Is there any tools required for implementation of the move?

3. Evaluation of the impact of migration data. The next step is to study the details. What happens when the data will be moved? What will the reaction of applications and users, respectively? How does it affect them? What will happen to data protection systems and access control? How do I notify of these changes?

4. Use one of the patterns of migration data. It managers know that the most effective is the phased solution to the problem. It is necessary to divide the work on phase and run each of the phases in sequence. This will help minimize the negative effects and to work quickly and efficiently.

5. evaluation of the work done. Never underestimate the paragraph plan! After the move, you should carefully check all data protection and access control applications.

6. To reassign or remove? Now it's time to think about the disk drive from which data has been moved. What should be done to him? Whether or not to

remove the protection from it, so that it can be removed from the system? Come the conclusion of the leasing contract and the device should be returned to the owner? Can I use it for other purposes? If the answers to these questions are already known, the data migration process completed successfully.

To switch to a different version of the DBMS manufacturers usually provide a means for implementing data migration. There are also separate programs performing data migration when moving from one RDBMS to another. However, the creation of tools to perform all data migration scenarios mentioned above remains a challenge.

All major data migration issues can be grouped as follows.

- Lack of expertise in data migration
- Weak understanding of data in source systems
- Changing the target system
- Insufficient quality data to be migrated
- Inability to synchronize after moving data

Each of the listed problems is severe enough, however, crucial for the success of migration may have a loyal project methodology is correctly chosen methodology greatly facilitates data migration.

When analyzing the products identified major data migration software products that implement data migration. The first product Migration is intended for planning and Architect to the following projects: creating a data warehouse and data marts; migrating data in the finished application. data migration and applications associated with the transition to a new DBMS; migration of data when you change the database schema. If the source database has a many-to-many relationship, the data migration process will increase dramatically, causing an interruption in the system of risk, which is the main disadvantage of this product. The second product is TRUmigrate manages complex data migration using effective and customizable wizard driven interface coding is not required. In this product, the downside is that if the structure of the XML data (OBD) are nested and difficult can be mapped to relational tables [17].

The first chapter provides an overview of the investigated subject. The chapter discusses the main directions in research on migration data that covers aspects such as terminology, types of data migration, the General principles of the migration data.

1. Basic concepts and definitions

When considering the subject area of the task were introduced the following concepts:

The dependent table is a table that has foreign keys in other tables.

Source database (SDB)-database, the data from which you want to migrate to another database.

The configuration file is an XML file that is passed to the data transfer phase. This file contains the generated sequence diagram bypass SDB and the necessary information to perform the migration mechanism of data (information about the tables and the relationships between them).

Data migration is the process of transferring data from a source database into a target, and their schema may differ.

The model is a set of classes that represent the database schema in memory in a unified way and let you work with its contents.

Independent data row is a row of data from a table that has no foreign keys, or a row from a table with foreign keys, but the values of those foreign keys are not defined for a given string (NULL).

Connection parameters to the database, the user name and password.

Crawl sequence diagrams sequence tables-SDU whose data you want to migrate in TDB that defines how to migrate data.

Table c "loop" is a table that contains 1 or more foreign keys from itself.

Successful data migration-migration of data, concluded without exceptions, such as the inability to migrate because of restrictions on the data in the database schema, etc.

The target database (TDB)-the target database-the database that you want to migrate the data [10].

2. Data migration

Data migration is the process of transferring data from a source database into a target, and their schema may differ.

The reasons that organizations start, data migration projects can be quite a lot: from application updates and the introduction of new enterprise systems to full-scale restructuring as a result of the merger.

Analyzing the experience of a large number of data migration information systems can distinguish typical data migration procedure, which includes:

- analysis of data formats and database structure of the original target, the migration plan and data conversion.
- defining relationships between tables (object hierarchy).
- determination of data transfer in accordance with the hierarchy of dependencies. Sometimes, you can ignore the relationship and just disable all foreign keys before migration and recreate them after all data manipulations. Exception — for example, when a new version of the database is already in use.
- script execution to change objects in the new version of the database.
- direct data transfer with the necessary transformations — "on the fly".
- script execution for the recovery of disabled indexes, additional conversions, etc. after the completion of the procedures for data migration.

The easiest option is to create an interim programme. It must communicate with the source and destination databases and perform the necessary changes (see Figure 1).

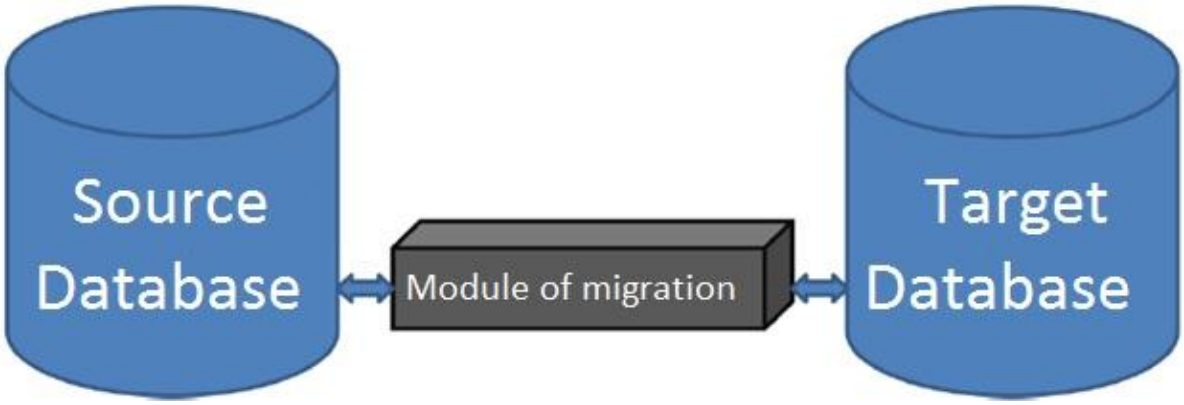


Figure 1. Data migration scenario with an intermediate program

Data migration is another option (this option is preferable when switching to a new database management system (DBMS))-preload old data into temporary tables in the new database. Modern database management systems (for example, Oracle) usually contain special utilities that allow very fast loading external data in various formats. In such a case the migration module is written by means of procedural languages, built-in DBMS (e.g., PL/SQL or java) (see Figure 2).

Figure 2. Option by the DBMS data migration

Although migrating data can be a fairly time-consuming process, the benefits can be worth the cost for those that "live and die" by trends in data. Migrating data to new systems means that older and more cumbersome applications need not be maintained. However, there are many more reasons as to why data migration is needed within any business. This section includes a

discussion on the following topics: Data migration definition, Migrating legacy data decision, and How to migrate data [10].

Some key terms in understanding data migration are:

- ***Legacy data*** is the recorded information that exists in your current storage system, and can include database records, spreadsheets, text files, scanned images and paper documents. All these data formats can be migrated to a new system.
- ***Data migration*** is the process of importing legacy data, from one or more old systems, to a new system or systems. This could involve entering the data manually, moving disk files from one folder (or computer) to another, using database insert queries, developing custom software, or other processes. The specific method used for any particular system depends entirely on the systems involved and the nature and state of the data being migrated.
- ***Data cleansing*** is the process of preparing legacy data for migration to a new system. Because the architecture and storage method of legacy data can be quite different from the new system, the legacy data often does not meet the criteria set by the new system, and must be modified prior to migration. For example, the legacy system may have allowed data to be entered in a way that is incompatible with the new system. Architecture differences, design flaws in the legacy system, or other factors can also render the data unfit for migration in its present state. The data cleansing process manipulates, and cleanses, the legacy data so that it conforms to the new system's requirements.

When deciding on data migration, all factors should be examined before making the assumption that the whole dataset or none of the dataset should be moved over to the new system. The proof is in whether these data records will be used and acted upon when the new system and process is in place. There are key variables to consider when considering data migration. These include data volume and data value.

Data volume is the easiest variable in the decision process. How many data records are we talking about: 1000, 10,000, 100,000, 250,000? How many are expected to come into the new system on a weekly/monthly basis to replenish this supply? Check to see if there are any technical barriers to bringing over a certain amount of data and also if large databases will affect performance of system functions like searching. If not, then 10 records or 100,000 records should not make any difference.

If volume is low, then it may be well worth carrying out a migration so there is some existing database for users and for trend analysis. If volume is high, then it may make sense to examine the age/value of the data and start filtering on certain criteria.

Data value is a much harder variable in the decision process. There are different perceptions concerning what value the existing data provides. If users are not working with older data in the current system, chances are they may not work with older data in the new system even with improved search functionality. If migrating, you may want to look at shorter-term date parameters - why bog down a system's performance with data that is never used?

Criteria, as discussed in the questions above, can be date parameters, but can also include other factors. Extracting the exact data based on some of these factors will depend on the abilities of your current system and database as well as the ability to write the detailed extraction script. Keeping it simple when possible is the best approach. However, there may be circumstances where filtering data may make sense.

Once you have determined which data you want to migrate, determining what parts of the data record are required will also be important [18].

3. Types of data migration

Use the migration data to achieve different purposes led to the formation of certain types of data migration:

1. the number of performed scripts per unit time:

- sequential;
- parallel.

Explanation of these types of migration are described on the example of relational databases. Sequential migration will execute one script at a time. The order of executing scripts based on dependencies between relationships. Parallel data migration is performed a more detailed analysis of the relationships between relationships. Script execution in this case comes at a time when all the necessary relations data already migrated [11].

2. According to the type of data storage:

- between the stores of the same type;
- between different types of stores.

Categories

Data is stored on various media in files or databases, and is generated and consumed by software applications which in turn support business processes. The need to transfer and convert data can be driven by multiple business requirements and the approach taken to the migration depends on those requirements. Four major migration categories are proposed on this basis.

Storage migration

A business may choose to rationalize the physical media to take advantage of more efficient storage technologies. This will result in having to move physical blocks of data from one tape or disk to another, often using virtualization techniques. The data format and content itself will not usually be changed in the process and can normally be achieved with minimal or no impact to the layers above.

Database migration

Similarly, it may be necessary to move from one database vendor to another, or to upgrade the version of database software being used. The latter case is less likely to require a physical data migration, but this can happen with

major upgrades. In these cases a physical transformation process may be required since the underlying data format can change significantly. This may or may not affect behavior in the applications layer, depending largely on whether the data manipulation language or protocol has changed – but modern applications are written to be agnostic to the database technology so that a change from Sybase, MySQL, DB2 or SQL Server to Oracle should only require a testing cycle to be confident that both functional and non-functional performance has not been adversely affected.

Application migration

Changing application vendor – for instance a new CRM or ERP platform – will inevitably involve substantial transformation as almost every application or suite operates on its own specific data model and also interacts with other applications and systems within the enterprise application integration environment. Furthermore, to allow the application to be sold to the widest possible market, commercial off-the-shelf packages are generally configured for each customer using metadata. Application programming interfaces (APIs) may be supplied by vendors to protect the integrity of the data they have to handle.

4. General principles of implementation of data migration

Data migration success depends on many factors. Some sources described the influence of technological, human factors and the factors associated with the data. Examples of technological factors are the amount of data to be migrated and the frequency of their migration. Factors related to data include data quality, the similarity of the source and target data structures, the dependencies between data and others. As any IT projects, data migration projects exposed to risks. Typical risks for projects of this type are in.

One way to reduce the risks of data migration projects is splitting into phases or stages. Thus, the data migration process is proposed, consisting of 7 phases: strategy, analysis, design, build, testing/implementation, validation,

tracking, highlighted three stages — analysis, testing, and implementation. For example, the migration from Oracle to Microsoft SQL Server when using the database migration process, consisting of 7 steps, described the possibility of applying DB Best products at each step and shows the proportion of data migration process assignments in percentage when using the described software products. Such partitioning to improve project management, as well as choose suitable software to perform certain tasks or data migration process as a whole. The literature identifies the characteristics that must have application to migrate data: logging, error notification, cold/hot start, implementation of clean data, index management, automatic detection of errors and other.

An important part of the data migration process is testing. It ensures that the target data store no defects, repository contains all the required data, the data is transferred to the necessary structures, etc. In spite of the importance of this aspect of testing is still unclear when describing the methods and approaches for data migration. In the sources described the testing life cycle data migration, review testing methods and quality control of the cast.

5. Effective data migration strategies

Many organizations migrate data as part of the process of upgrading controls (think Sarbanes-Oxley), systems, or storage. Enterprises need to minimize the business impacts of data migration - downtime, data integrity issues, costs, control problems, and so on. The way to do this is to utilize a robust methodology for migrations, which we discuss in this note. A majority of respondents to data migration surveys report one problem or another with these migrations. Simply scheduling migrations during “off-hours” is not always a sufficient strategy, since:

- it increases migration costs and decreases employee morale due to staff overtime
- migrations which go over schedule can disrupt processing during normal hours

- most enterprises no longer have a significant “off-hour” window for activities like data backups or data migrations due to their global operations or other customer demands for availability

Effective data migration strategies go a step further and can avoid such risks, as we will see [8].

Specific operational goals of effective data migration

There are specific goals associated with implementing an effective data migration strategy. Primarily, data must be migrated from the source platform to the target platform completely and accurately, and according to company and regulatory policies on information controls and security. This means no dropped or incomplete records, and no data fields that fail validation or other quality controls in the target environment. Another goal of data migration is that the process be done quickly, with as short a downtime window as possible (given the enterprise's Maintenance Time Objective targets.) Finally, the cost of data migration must be manageable, in terms of technology and staff requirements.

- There are many metrics that can measure the effectiveness and efficiency of data migrations:
    - Number of online customizations required
    - Percentage of migrated records
    - Percentage of migrated tables
    - Percentage of migrated records by technology
    - Percentage of migrated records by application
    - Percentage of data with quality problems
    - Number of migration errors
    - Migration impact on database size
    - Downtime due to migration
    - Required staging storage / hardware
    - Percentage of reconciliation errors
    - Percentage of cleansed data

Risks of implementing an effective data migration strategy

Migrating data brings many risks and challenges:

- Migration might be done as part of a larger chain of dependencies (operating system upgrades, application upgrades or implementations, database structural changes, etc.) – thereby increasing complexity.
- Data requirements are not clearly defined - data rules for integrity, controls, security, availability and recoverability are often ill-defined. In the absence of such rules, data is migrated incorrectly.
- Migration acceptance criteria may not be defined.
- Data is often too distributed to be migrated easily (think end-user computing).
- Budgets may limit technology options for performing migrations.
- Expertise in data migration and management may not be present.
- Management attention might be insufficient (migrations are typically "routine" operations and not major attention-getting projects).
- There could be poor support from the storage vendor(s), making migrations all the more taxing.

The effective data migration strategy

The business driver for effective data migrations is to strike the optimal balance between data accuracy, migration speed, low or no downtime, and minimum costs. We will see how to formulate such a strategy.

Expectations (Out-of-scope)

To a large extent the enterprise's storage technology, storage management tools, and storage network capabilities define the range of possibilities for data migration strategies. Selection of such is out of scope of this note; however, we will introduce a methodology for data migration strategy that can work under most technology environments.

Analyze phase

The first stage of data migration is data classification. It is important to know how and where data is stored, backed up, and archived. Data structures must be well understood, and there should be visibility into the data's usage, capacity, and growth patterns. Various interfaces and reports utilizing the to-be-migrated data must be considered. It is also important to understand the network connections between data points (especially required bandwidth and controls). Data classification also describes conditions for data access, retention requirements and control & security measures such as encryption. Next, migration requirements are determined. Beyond the "top three" requirements of what must be migrated, how much downtime is acceptable, and how much budget is available, there are other needs that must be considered. These can include quality requirements, new or modified service-level agreements, expectations for the new storage infrastructure, and objectives such as reduced management costs, reduced storage expenditures, greater insight into expenditure, a simplified vendor model or greater technical flexibility or stability. If the organization has standards or policies concerning data, these must be factored in. Finally, analysts must take into account historical data to be migrated (which is often forgotten until it is late in the game).

Then there are technology considerations, such as:

- How old is the operating system(s) under which data is to be migrated? Some migration tools do not support legacy operating systems.
- Which storage tiers and devices are involved?
- Can identical data structures be used for the target data (this will reduce the time and complexity of migration)?
- What staging area requirements are present, given current technologies and data migration requirements?
- Do you need or want the option to recover quickly from the source disk, or to fall back to the original storage device as a fail-over? There are both procedural and technological ways of accomplishing this.

- Is a central console needed to manage data migrations across multiple servers?
- Is there a need to control the data migration from a local server – or a remote server? If remote, which protocols must be supported?
- Is there a requirement to throttle or control data flows between servers?
- Must data integrity be checked during (not just after) migration?
- Which staff or consultants are available to assist with migration analysis, design, and deployment?

Data may be migrated in several ways:

1. All at once
2. In logical sections (by application, operating system, database, business function, etc.)
3. Via a pilot migration or parallel run for "proof of concept" or risk mitigation.
4. In phases (first critical data, then less critical or historical data).

Acceptance Test Considerations

The Analyze Phase is complete when data has been classified, data migration requirements are clear, and technology considerations have been considered. For organizations with an established data classification program (that typically includes government entities, regulated firms, and enterprises complying with quality standards) this can be done in a week or two. If a full data classification must be done as well, that can add another few weeks.

Key analysis milestones

Milestones in the Analysis Phase typically include the following:

- Legacy data and report analysis and profiles has been completed.
- There is an updated data classification document.
- The data migration strategy document has been written.

- Data exception report requirements are clear.
- Audit and reconciliation report requirements are defined.
- Data migration requirements (business and technology) are documented.

Design phase

The design phase involves taking the data classification and requirements defined in the analysis phase and puts definition around how these will be realized:

- The responsibilities, roles and tasks for each individual, department, or external consultant involved in migration are determined.
- Data elements are mapped from source to target.
- A plan for freezing physical data structures during the migration is put together.
- Any tools to be used in the migration (data transformation tools, migration options, data mapping tools, CASE or other data modeling tools, simple spreadsheets, etc.) are identified or acquired.
- Any software to facilitate data extraction or checking is developed.
- Clear acceptance criteria are determined and agreed.

Where economically viable (large volumes of data to migrate, frequent mission-critical migrations, complex data structures, high data quality requirements) data transformation tools can offer the following advantages over simple spreadsheets:

- They provide flexible reporting of data elements and rules.
- They can generate migration code or scripts directly from the mapping rules.
- They can detect data integrity violations.

Acceptance test considerations

The Design Phase is complete when migration staffing has been confirmed, data mapping is complete, a plan for freezing data structures is in place, tools have been identified, and acceptance criteria have been agreed. The

length of the design phase depends mainly on the need to acquire additional migration tools and the need to develop extraction, loading, and quality checking software. If such tools and software are in place and simply need to be customized, the design phase can take place in a few weeks; otherwise, this process can take a few months.

Key design milestones

Milestones in the Design Phase include the following:

- Roles, responsibilities, and task assignments are clear and accepted.
- Data elements are mapped from source to target.
- A plan for freezing physical data structures during the migration is put together.
- Migration tools are identified.
- Migration software is developed.
- A migration strategy is chosen.
- Acceptance criteria are agreed.
- Funding for staff, consultants, and tools has been secured.

Deploy phase

The first step in the design stage is to put together a project plan and structure. As part of this process there should be close analysis of any dependencies in data migrations; where possible, such dependencies and complexities should be reduced to better manage deployment risk. During the deploy phase the following occurs:

- Physical data structures are frozen on source and target.
- Interfaces and processing on source and target are brought down where required.
- Data is staged from the source location.
- Quality reports are run and any data errors or inconsistencies identified.

- Data quality issues are fixed in the staging area.
- A preliminary reconciliation takes place in the staging area, and any reconciling items are investigated and resolved.
- Data is migrated to the target location.
- Reconciliation reports are run.
- Acceptance criteria are check; if reconciliation errors or other criteria are not met, the system is rolled back to the original data source. Otherwise, interfaces and processing from the source is discontinued and then activated on the target.
- The above may be conducted in phases, or as part of a parallel run or pilot depending upon the migration approach chosen.

Acceptance Test Considerations

Testing and implementation are inseparable in a data migration. Physical errors are typically syntactical in nature and can be easily identified and resolved through syntax corrections in the migration scripts. Logical errors are due to problems with the data mapping. Looking into these requires asking questions like:

- How many records did we expect this script to create?
- Did the correct number of records get created? If not, why?
- Has the data been loaded into the correct fields?
- Is the data load complete – or are certain fields missing?
- Has the data been formatted correctly?
- Are any post-migration clean-up tasks in order?

The goal of a successful data migration is to keep the length of the deploy phase(s) to a minimum. What is acceptable depends upon the organization – some will want this phase to last no more than a day (or less), others can tolerate a deployment that lasts a few days if availability is not a major concern.

Key deployment milestones

Milestones in the Deploy Phase include the following:

- Project plan in place and agreed.
- Project team is formed.
- A test plan is written.
- The migration takes place.
- Reconciliation / data checking reports are run.
- Acceptance criteria and performance metrics are evaluated.
- A go / no-go decision takes place.
- Data issues may be resolved post-migration.

Data Migration Challenges

Let us describe the data migration challenges in little more detail. Data migration can be a simple process, however there are challenges one may face in implementation [17].

Storage Migration

Storage migration can be handled in a manner transparent to the application so long as the application uses only general interfaces to access the data. In most systems this is not an issue. However, careful attention is necessary for old applications running on proprietary systems. In many cases, the source code of the application is not available and the application vendor may not be in market anymore. In such cases storage migration is rather tricky and should be properly tested before releasing the solution into production.

Database Migration

Database migration is rather straight forward, assuming the database is used just as storage. It "only" requires moving the data from one database to another. However, even this may be a difficult task. The main issues one may encounter include:

- Unmatched data types (number, date, sub-records)
- Different character sets (encoding)

Different data types can be handled easily by approximating the closest type from the target database to maintain data integrity. If a source database supports complex data formats (e.g. sub-record), but the target database does not, amending the applications using the database is necessary. Similarly, if the source database supports different encoding in each column for a particular table but the target database does not, the applications using the database need to be thoroughly reviewed.

When a database is used not just as data storage, but also to represent business logic in the form of stored procedures and triggers, close attention must be paid when performing a feasibility study of the migration to target database. Again, if the target database does not support some of the features, changes may need to be implemented by applications or by middleware software.

ETL tools are very well suited for the task of migrating data from one database to another i. Using the ETL tools is highly advisable particularly when moving the data between the data stores which do not have any direct connection or interface implemented.

Application Migration

If we take a step back to previous two cases, you may notice that the process is rather straight forward. This however, is extremely uncommon in the case of application migration. The reason is that the applications, even when designed by the same vendor, store data in significantly different formats and structures which make simple data transfer impossible. The full ETL process is a must as the Transformation step is not always straight forward. Of course, application migration can and usually does include storage and database migration as well. The advantage of an ETL tool in this instance is its ready-to-use connectivity to disparate data sources/targets.

Difficulty may occur when migrating data from mainframe systems or applications using proprietary data storage. Mainframe systems use record based formats to store data. Record based formats are easy to handle; however, there

are often optimizations included in the mainframe data storage format which complicate data migration. Typical optimizations include binary coded decimal number storage, non-standard storing of positive/negative number values, or storing the mutually exclusive sub-records within a record. Let us consider a library data warehouse as an example. There are two types of publications - books and articles. The publication can be either a book or an article but not both. There are different kinds of information stored for books and articles. The information stored for a book and an article are mutually exclusive. Hence, when storing a book, the data used has a different sub-record format for a book and an article while occupying the same space. Still the data is stored using rather standard encoding. On the Contrary, proprietary data storage makes the Extract step even more complicated. In both cases, the most efficient way to extract data from the source system is performing the extraction in the source system itself; then converting the data into a printable format which can be parsed later using standard tools.

Business process migration

Business processes operate through a combination of human and application systems actions, often orchestrated by business process management tools. When these change they can require the movement of data from one store, database or application to another to reflect the changes to the organization and information about customers, products and operations. Examples of such migration drivers are mergers and acquisitions, business optimization and reorganization to attack new markets or respond to competitive threat.

The first two categories of migration are usually routine operational activities that the IT department takes care of without the involvement of the rest of the business. The last two categories directly affect the operational users of processes and applications, are necessarily complex, and delivering them without significant business downtime can be challenging. A highly adaptive approach, concurrent synchronization, a business-oriented audit capability and

clear visibility of the migration for stakeholders are likely to be key requirements in such migrations.

Project versus process

There is a difference between data migration and data integration activities. Data migration is a project by means of which data will be moved or copied from one environment to another, and removed or decommissioned in the source. During the migration (which can take place over months or even years), data can flow in multiple directions, and there may be multiple migrations taking place simultaneously. The Extract, Transform, Load actions will be necessary, although the means of achieving these may not be those traditionally associated with the ETL acronym.

Data integration, by contrast, is a permanent part of the IT architecture, and is responsible for the way data flows between the various applications and data stores - and is a process rather than a project activity. Standard ETL technologies designed to supply data from operational systems to data warehouses would fit within the latter category.

Major challenges of data migration

All major data migration issues can be grouped as follows.

Lack of expertise in data migration:

- Organizations very rarely have a centre of competence and quality migration documentation that could be effectively used in the work on the project.

Weak understanding of data in source systems:

- Data are often scattered by outdated source systems in a variety of formats and do not meet the required level of quality and accuracy. Documentation is usually missing or already out of date. To understand this data is capable of only a specialist, who knows the source systems and business functions, and such people in the State, usually a bit.

Changing the target system:

- During the migration project, the target system, in which data is moved, often located refinement. And any changes you made to it may significantly alter the data migration requirements.

Insufficient quality data to be migrated:

- Data, it is important not to just move, but make them working in the target system. They must comply with the

The inability to sync after moving data:

- Often the old system for a long time, leave to work in parallel with the new ones. During the transition period without synchronizing data between these systems. In addition, you must be able to demonstrate, when regulators upon project completion reports for correctness and dates of data movement.

Each of the listed problems is severe enough, however, crucial for the success of migration may have a loyal project methodology is correctly chosen methodology greatly facilitates data migration.

Subtasks of data migration

Known approaches and methods data migration explicitly or implicitly perform analysis of data storage and data transfer.

The analysis compares data stores or comparison of the structures of the source and destination data stores. Under the mapping schema (schema matching) understand the task definition of semantic correspondences between elements in the schema. Mapping between schemas is traditionally considered AI-complete task, and this limits the variants of its solution. Classification of approaches to mapping schemes can be found in the reviews. In Russian literature found use of the term "schema comparison 2" as a synonym for schema mapping.

The idea of the proposed methods and approaches for mapping schemes based on comparing graphs, probability theory, machine learning and other

areas. The most important characteristics of techniques and approaches to mapping schemas are precise result and time consuming. Since AI-completeness of schema mapping necessitates adjustments results matching man and plus sizes schemes increase run time comparison, much attention to laborious the proposed algorithms. For example, if you are using Graph matching, it should be borne in mind that in General, the task of comparing graphs not attributed either to class P nor to the class of NP-complete problems, and isomorphism subgraph (one type of graph matching) is NP-complete. Thus, if necessary, schema mapping selection method for its implementation should be carried out, in accordance with the valid maximum accuracy and complexity.

When migrating data between different types of data stores mapping structures is complicated by using different data models. For example, migrate relational databases in object-oriented or XML database. The difficulty lies in the fact that often there is no ready-made and/or unambiguous rules compare elements of different data models. There are several possible approaches to mapping elements from different data models. One such approach is to define comparison rules for each possible data model. Another approach is based on using an intermediate conceptual representation, which will be submitted to the source and target data structure. So, as an intermediate representation offered Earl, the relational model, semantic models, XML files. Disadvantages of this approach are to possible loss of semantics when going to an intermediate representation, as well as in increasing labour input. It is obvious that migration of data between data stores of the same type are not faced with these problems, but it limits the scope of its application. However, even in the case of data migration between data stores of the same type there are difficulty to perform the comparison of data structures. They are associated with differences in specific implementations of data stores. For example, one data store the length of the data type string is limited, and the other is not; in relational DBMS can use different standards of the SQL language.

Various representations of objects modeled world and the relationships between them also complicate the comparison of data structures of the source and destination data stores. Speaking terms ER-Chen, the same phenomenon modeled world can be represented as an attribute, as many entities and how many links. Thus, there is a problem of matching objects and their relationships in the possibility of using different views. This problem also applies to the use of equivalent structures of data models and different points of view in the objects view of the simulated world, for example, the use of different names (synonyms, giperonimov), integrity constraints, data types. Classification of possible differences between the views of objects and their relations are.

Implementation data migration involves several issues. One of them is to define data migration. Solution to the issue depends on the existence of relationships between objects modeled world represented in the data store. In the case where the data warehouse is represented by only one type of object simulated world without links (for example, the relational model, this means that there is only one attitude that has no foreign keys), difficulties in defining the order no data migration. When the data store provides several types of objects without links, also brings no special difficulties: migrated for each type of object in any order. Difficult to determine the order of data migration are situation with the existence of relationships between objects. Considering the migration data in relational databases, proposed an approach based on facts. Fact is a combination of semantics and one or more atomic values. There are two types of facts is independent and dependent. Migration facts consists of two subprocesses: independent migration facts and long-range dependent facts. Similar ideas are incorporated into other methods and approaches, for example, methods and systems for data migration, described. However, the proposed methods do not solve the problem of occurrence of cyclical relationships between objects of different or same types.

Important is to move data and definition of values unique object identifiers of the simulated world. In the case of transfer in non-empty data store

possible conflicts of unique identifiers, and when transferring to another type of data store or a different implementation or in the data warehouse structure differs from the structure of the source repository, you may experience problems such as an inability to convert one data type unique identifiers in another and so on way out of a problematic situation in the form of a unique identifier values change entails changing the values of these unique identifiers in the relationship between the objects modeled world that also makes it more difficult to migrate the data.

When migrating data, it is often necessary to perform the conversion. The complexity of admissible transformations in specific methods and approaches that would define their applicability to a particular data migration project. For example, assume that in the source data does not have unnecessary information and data are correct, and does not consider not expressing the semantics of keys and complex data transformations. Generate scripts for predefined conversion templates provided in relational databases. Systematic coverage of issues related to the conversion of data in database schema changes.

Thus, the data migration is a complex task that involves different kinds of errors. This chapter covered the basic directions in research on migration data. Matters will assess the applicability of methods and approaches for specific data migration projects, as well as to determine the direction of development of existing and new methods and approaches.

6. Analysis of data migration products on the market

On the market there are a sufficient number of software that implement data migration from one Wednesday to another. Of these products, to analyze today's trends were taken 2 most popular.

The first product called Migration Architect, manufacturer which Evoke Software (http://www.information-management.com), is intended for planning and implementation of the following projects: creating a data warehouse and data marts; migrating data in the finished application. data migration and

applications associated with the transition to a new DBMS; migration of data when you change the database schema.

Main features:

- analysis of not only the metadata, but actual data to more accurately define the scope of attribute values and checking for primary keys.
- the definition of relations between attributes (functional dependence), and relationships between tables (identification data redundancy).
- support for simple models to save and define mappings between the source and target attributes database schema.

The second product is TRUmigrate (manufacturer: Valiance Partners, http://www.valiancepartners.com), which did not go unnoticed among the users due to its flexibility. On manages complex data migration using effective and customizable wizard driven interface coding is not required. It is compatible with more than thirty of the most commonly used applications in regulated business Wednesday, including Documentum, SharePoint, Lotus Notes, MasterControl, TrackWise, and QUMAS. This allows the TRUmigrate quickly, offer accurate data migration functionality.

Main features:

- Setting the rules. Display are specified by means of a graphical user interface and are stored in an XML file.
- export. Content extraction and meta information from one or multiple data stores and transfer received data to an intermediate store.
- Addition and concatenation. Add data from auxiliary sources and merging with earlier findings.
- conversion. Application of rules of transformation for the received data.
- import. Download content objects in the target application and initialize settings.

Shredding method

This method is commonly known as grinding or decomposition of XML documents. The concept of XML shredding is illustrated in Figure 3. In this example, the customer name, address and phone information in an XML document are mapped to two relational tables. Documents can contain multiple phone numbers, because the relationship between clients and one-to-many phones. Therefore, the numbers are recorded in a separate table. Each repeating element, such as phone numbers, leads to the creation of additional tables in a relational target schema. Suppose customer information can also contain multiple e-mail accounts, list of the most recent customer orders several products in the order and other recurring items. The number of tables in a relational schema in the target can grow very quickly. Shredding XML into a large number of tables can lead to complex and unnatural fragmentation of logical business objects that makes developing applications difficult and contributes to errors. Request crushed data or build the original documents may require sophisticated Multilink.

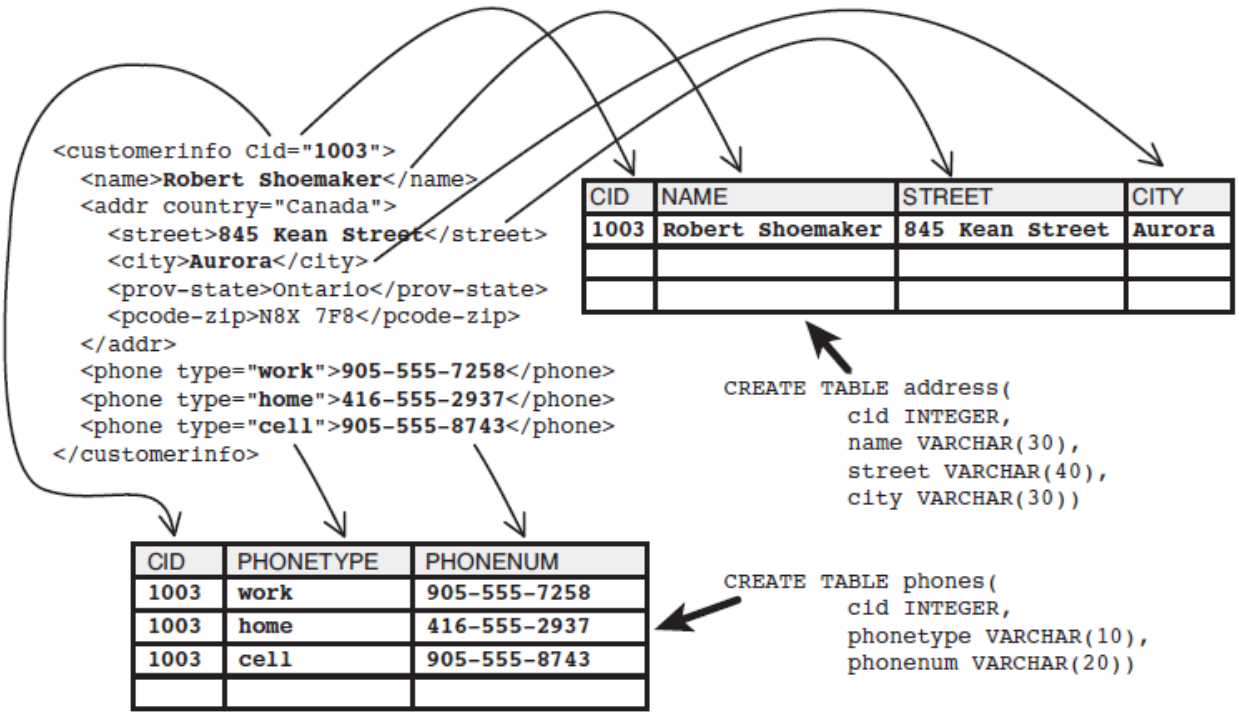


Figure 3. Shredding method of XML document

Depending on the complexity, variability, and assign your grinding method of XML document may or may not be a good choice. Table 1 lists the pros and cons of shredding XML data into relational tables [6].

Table 1. Advantages and disadvantages of the shredding method

Shredding method can be useful when	**Shredding method is not the best choice when**
All fields of the incoming XML document exactly matched the relevant attributes of the relational DB	XML data are complex, nested and it is difficult to compare with a relational schema.
XML documents do not represent business logic objects that must be preserved.	Mapping XML to relational schema document leads to the creation of a large number of tables.
The main purpose is to allow existing relational applications to access XML data.	The XML schema is changeable or tends to vary over time.
XML data structure is such that it can easily be mapped to relational tables.	It is often necessary to reconstruct shredded documents or parts thereof.
An XML document is relatively stable and rarely suffers changes.	Write XML data to a database with a high speed is an important factor for the application.
Query or update data with SQL is more important than write performance.	

Synchronization problem after moving data is not solved in any product. This work will be devoted to solving this problem.

The aim of this work is the design and development of a data migration method from XML into a relational database.

Conclusion

Based on the review of the subject area and analysis of existing products, following objectives were set:

- To learn the basics of the data synchronization problem
- To do an analysis of methods that best suited to the task
- To develop a new method of data migration
- To conduct testing method

CHAPTER II. MODEL DESIGN

In computer science, synchronization refers to one of two distinct but related concepts: synchronization of processes, and synchronization of data. Process synchronization refers to the idea that multiple processes are to join up or handshake at a certain point, in order to reach an agreement or commit to a certain sequence of action. Data synchronization refers to the idea of keeping multiple copies of a dataset in coherence with one another, or to maintain data integrity. Process synchronization primitives are commonly used to implement data synchronization.

Thread synchronization is defined as a mechanism which ensures that two or more concurrent processes or threads do not simultaneously execute some particular program segment known as critical section. When one thread starts

executing the critical section (serialized segment of the program) the other thread should wait until the first thread finishes. If proper synchronization techniques are not applied, it may cause a race condition where, the values of variables may be unpredictable and vary depending on the timings of context switches of the processes or threads.

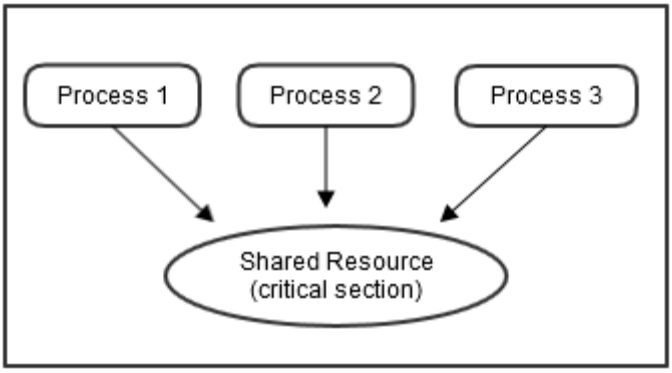


Figure 4. Three processes accessing a shared resource (critical section) simultaneously

For example, suppose that there are three processes namely, 1, 2 and 3. All three of them are concurrently executing and then need to share a common resource (critical section) as shown in Figure 4. Synchronization should be used here to avoid any conflicts for accessing this shared resource. Hence, when Process 1 and 2 both try to access that resource it should be assigned to only one process at a time. If it is assigned to Process 1, the other process (Process 2) needs to wait until Process 1 frees that resource (as shown in Figure 5).

Another synchronization requirement which needs to be considered is the order in which particular processes or threads should be executed. For example, we cannot board a plane until we buy the required ticket. Similarly, we cannot check emails before validating our credentials (i.e., user name and password). In the same way, an ATM will not provide any service until we provide it with a correct PIN.

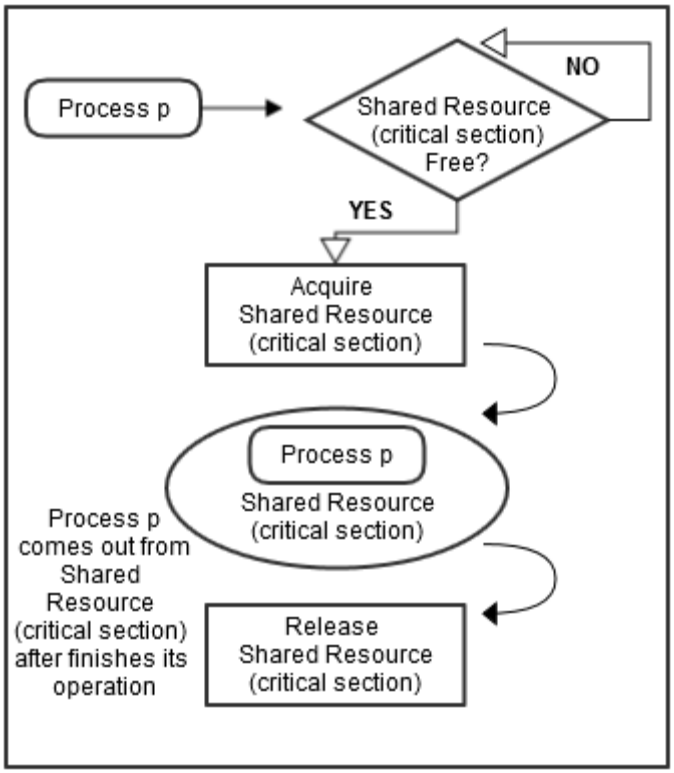


Figure 5. A process accessing a shared resource if available, based on some synchronization technique

Other than mutual exclusion, synchronization also deals with the following:

- deadlock, which occurs when many processes are waiting for a shared resource (critical section) which is being held by some other process. In this case the processes just keep waiting and execute no further;
- starvation, which occurs when a process is waiting to enter the critical section but other processes monopolize the critical section and the first process is forced to wait indefinitely;
- priority inversion, which occurs when a high priority process is in the critical section, it may be interrupted by a medium priority process. This violation of priority rules can happen under certain circumstances and may lead to serious consequences in real-time systems;
- busy waiting, which occurs when a process frequently polls to determine if it has access to a critical section. This frequent polling robs processing time from other processes.

Processes access to critical section is controlled by using synchronization techniques. This may apply to a number of domains.

Classic problems of synchronization

The following are some classic problems of synchronization:

- The Producer–Consumer Problem (also called the The Bounded Buffer Problem);
- The Readers–Writers Problem;
- The Dining Philosophers Problem.

These problems are used to test nearly every newly proposed synchronization scheme or primitive.

Hardware synchronization

Many systems provide hardware support for critical section code.

A single processor or uniprocessor system could disable interrupts by executing currently running code without preemption, which is very inefficient on multiprocessor systems. "The key ability we require to implement synchronization in a multiprocessor is a set of hardware primitives with the ability to atomically read and modify a memory location. Without such a capability, the cost of building basic synchronization primitives will be too high and will increase as the processor count increases. There are a number of alternative formulations of the basic hardware primitives, all of which provide the ability to atomically read and modify a location, together with some way to tell if the read and write were performed atomically. These hardware primitives are the basic building blocks that are used to build a wide variety of user-level synchronization operations, including things such as locks and barriers. In general, architects do not expect users to employ the basic hardware primitives, but instead expect that the primitives will be used by system programmers to build a synchronization library, a process that is often complex and tricky." Many modern hardware provides special atomic hardware instructions by either

test-and-set the memory word or compare-and-swap contents of two memory words.

Synchronization strategies in programming languages

In Java, to prevent thread interference and memory consistency errors, blocks of code are wrapped into synchronized (lock_object) sections. This forces any thread to acquire the said lock object before it can execute the block. The lock is automatically released when thread leaves the block or enter the waiting state within the block. Any variable updates, made by the thread in synchronized block, become visible to other threads whenever those other threads similarly acquires the lock.

In addition to mutual exclusion and memory consistency, Java synchronized blocks enable signaling, sending events from those threads, which have acquired the lock and execute the code block to those which are waiting for the lock within the block. This means that Java synchronized sections combine functionality of mutexes and events. Such primitive is known as synchronization monitor.

Any object is fine to be used as a lock/monitor in Java. The declaring object is implicitly implied as lock object when the whole method is marked with synchronized.

The .NET framework has synchronization primitives. "Synchronization is designed to be cooperative, demanding that every thread or process follow the synchronization mechanism before accessing protected resources (critical section) for consistent results." In .NET, locking, signaling, lightweight synchronization types, spinwait and interlocked operations are some of mechanisms related to synchronization.

Data synchronization

A distinctly different (but related) concept is that of data synchronization. This refers to the need to keep multiple copies of a set of data coherent with one another or to maintain data integrity, Figure 6. For example, database replication

is used to keep multiple copies of data synchronized with database servers that store data in different locations.

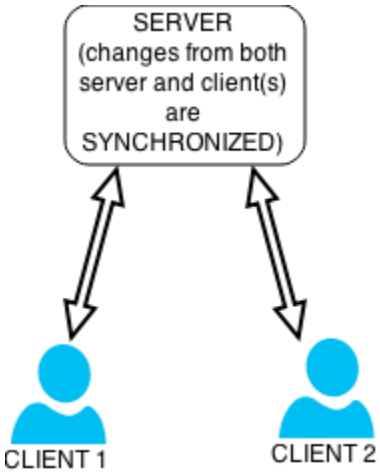


Figure 6. Changes from both server and client(s) are synchronized

Examples include:

- file synchronization, such as syncing a hand-held MP3 player to a desktop computer;
- cluster file systems, which are file systems that maintain data or indexes in a coherent fashion across a whole computing cluster;
- cache coherency, maintaining multiple copies of data in sync across multiple caches;
- RAID, where data is written in a redundant fashion across multiple disks, so that the loss of any one disk does not lead to a loss of data;
- database replication, where copies of data on a database are kept in sync, despite possible large geographical separation;
- journaling, a technique used by many modern file systems to make sure that file metadata are updated on a disk in a coherent, consistent manner.

Challenges in data synchronization

Some of the challenges which user may face in data synchronization:

- data formats complexity;
- real-timeliness;

- data security;
- data quality;
- performance.

Data formats complexity

When we start doing something, the data we have usually is in a very simple format. It varies with time as the organization grows and evolves and results not only in building a simple interface between the two applications (source and target), but also in a need to transform the data while passing them to the target application. ETL (extraction transformation loading) tools can be very helpful at this stage for managing data format complexities.

Real-timeliness

This is an era of real-time systems. Customers want to see the current status of their order in e-shop, the current status of a parcel delivery—a real time parcel tracking—, the current balance on their account, etc. This shows the need of a real-time system, which is being updated as well to enable smooth manufacturing process in real-time, e.g., ordering material when enterprise is running out stock, synchronizing customer orders with manufacturing process, etc. From real life, there exist so many examples where real-time processing gives successful and competitive advantage.

Data security

There are no fixed rules and policies to enforce data security. It may vary depending on the system which you are using. Even though the security is maintained correctly in the source system which captures the data, the security and information access privileges must be enforced on the target systems as well to prevent any potential misuse of the information. This is a serious issue and particularly when it comes for handling secret, confidential and personal information. So because of the sensitivity and confidentiality, data transfer and all in-between information must be encrypted.

Data quality

Data quality is another serious constraint. For better management and to maintain good quality of data, the common practice is to store the data at one location and share with different people and different systems and/or applications from different locations. It helps in preventing inconsistencies in the data.

Performance

There are five different phases involved in the data synchronization process:

- data extraction from the source (or master, or main) system;
- data transfer;
- data transformation;
- data load to the target system.

Each of these steps is very critical. In case of large amounts of data, the synchronization process needs to be carefully planned and executed to avoid any negative impact on performance [15].

1. Synchronization methods

Replication system worked correctly, you need a properly running the algorithm for determining data to be passed between the two nodes. This algorithm is one of the key features of replication systems. Consider the different algorithms of detection (or capture) changes. The algorithms are considered here, which as a result of its work is guaranteed to synchronize two databases (subject to the normal operation of the system and the lack of changes during a data exchange).

The full transfer of tables is a degenerate case of change detection, when really there is no capture changes, and the table is passed in its entirety during the replication session. Hypothetically, this approach may be applicable only in cases when the nodes are linked very fast data transfer channel. In practice, it is

almost never used because it was too inefficient use of resources. The full transfer of the entire table is meaningful only at the initial stage of the interaction between nodes, i.e. at the stage of creation of remote copies. Figure 7 shows the basic scheme of the algorithm.

Advantages: ease of technical implementation.

Disadvantages: expensive resources that do not allow the use of this method in commercial systems.

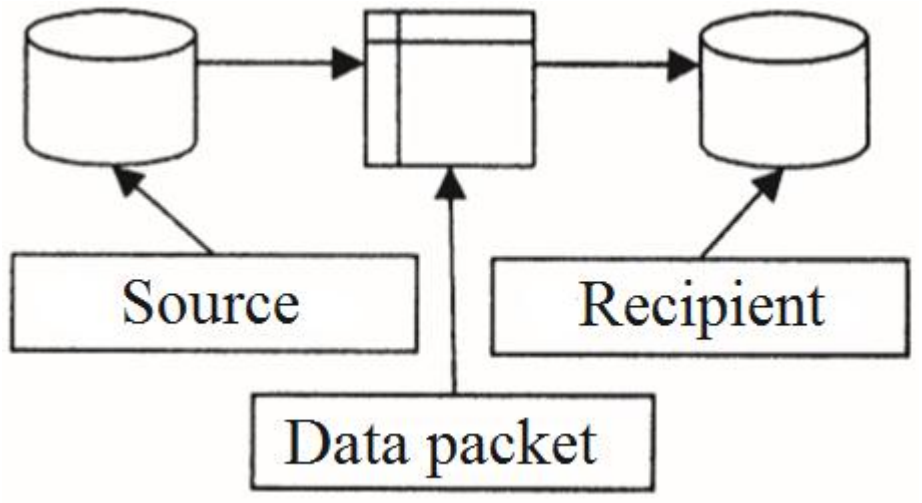


Figure 7. Method of full transfer tables

Change detection using double copying of tables is that after each session replication for each table, you create a copy (either in the same database, or on a different host). This copy does not change until the next replication session. Thus, any changes made to a table, you can get by comparing the source table and a copy of it. An example of such an algorithm is shown in Figure 8.

Advantages:

1. No need to change an existing programming code and database structure.

2. Such an approach will be found all changes, regardless of how they were made.

Disadvantages:

1. For each tracked table kept her complete copy that generally doubles the amount of data.

2. In the event of a failure on the primary node recovery should occur not quite justified algorithm-full forwarding table with subsequent analysis of data that really need to be updated.

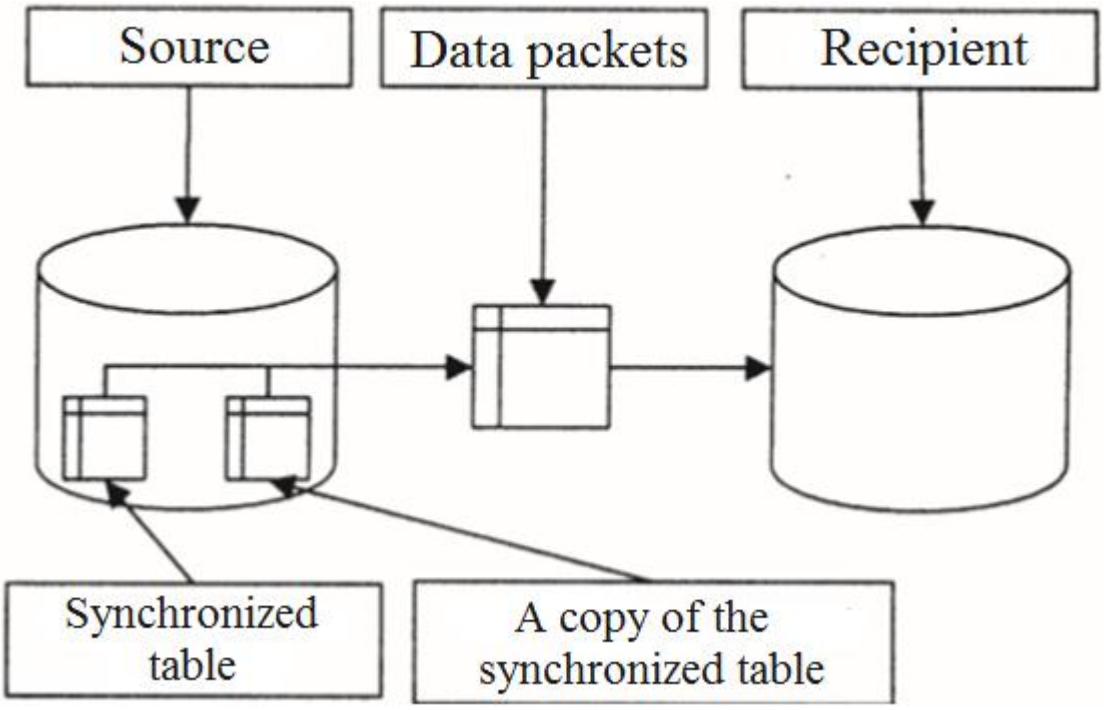


Figure 8. Double copying tables.

An intermediate level of access to the database is an additional software layer between the level of logic and database access layer. Thus, along with the record of changes in the table being tracked this intermediate writes, and in turn changes. In Figure 9 shows one possible implementation of this method.

Advantages:

1. No need to make changes to an existing database schema, and such conditions are often assigned to the developers.

Disadvantages:

1. Could be considerable costs when introducing an intermediate level of access to the database, even if the original application was designed correctly.

2. Not tracked changes made programmatically, do not use this intermediate level of access. This occurs when, for example, the background changes from both the host and from a variety of support tools.

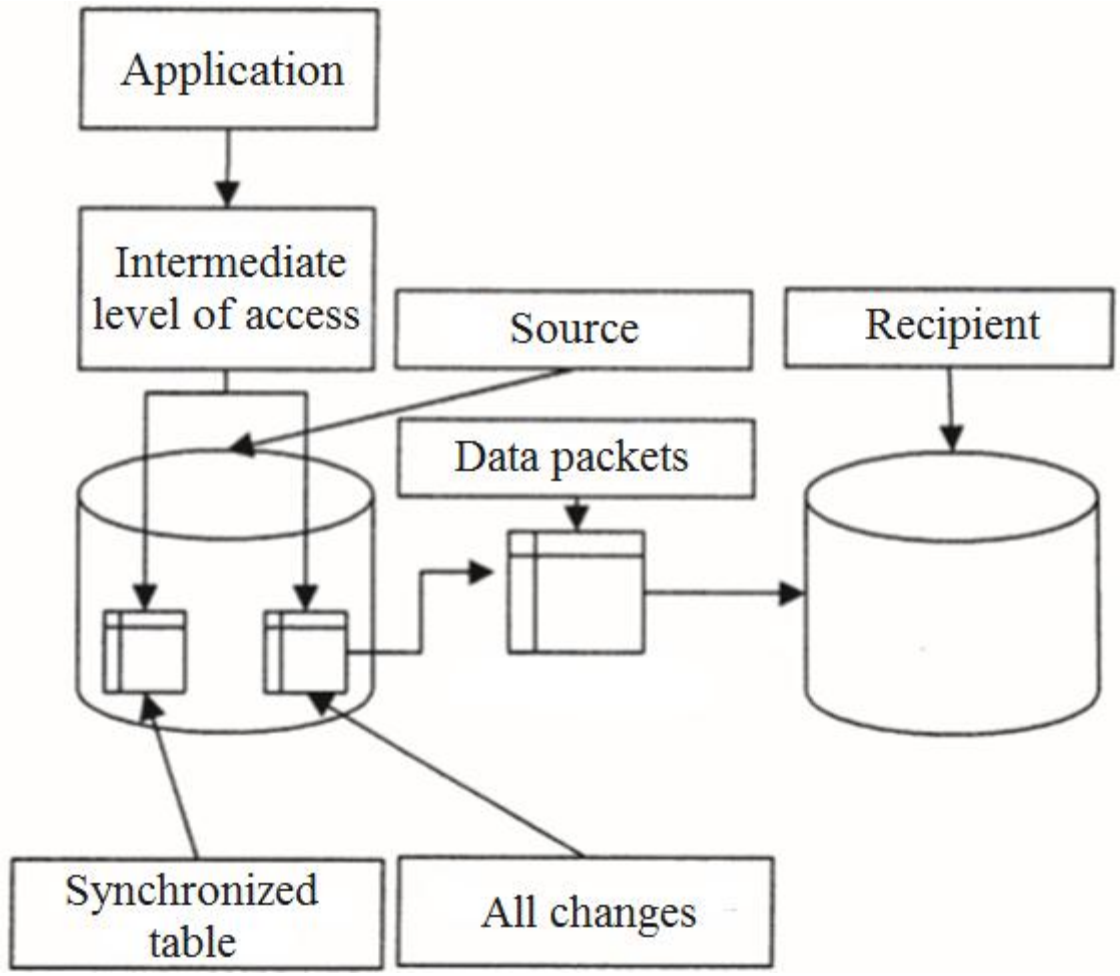


Figure 9. Principle of operation of the system with an intermediate level of access to the database

Change detection using the transaction log. Replication tracks changes to the database transaction log and either creates a new queue changes or uses the transaction log itself to generate the list of changes. In Figure 10 shows how this algorithm works.

Advantages:

1. No need to make changes to an existing database schema.

2. There is no dependence on a software tool used for editing.

Disadvantages: Different transaction log architecture. The architecture of this journal is different not only in different DBMS vendors, but also different versions of the DBMS from a single supplier.

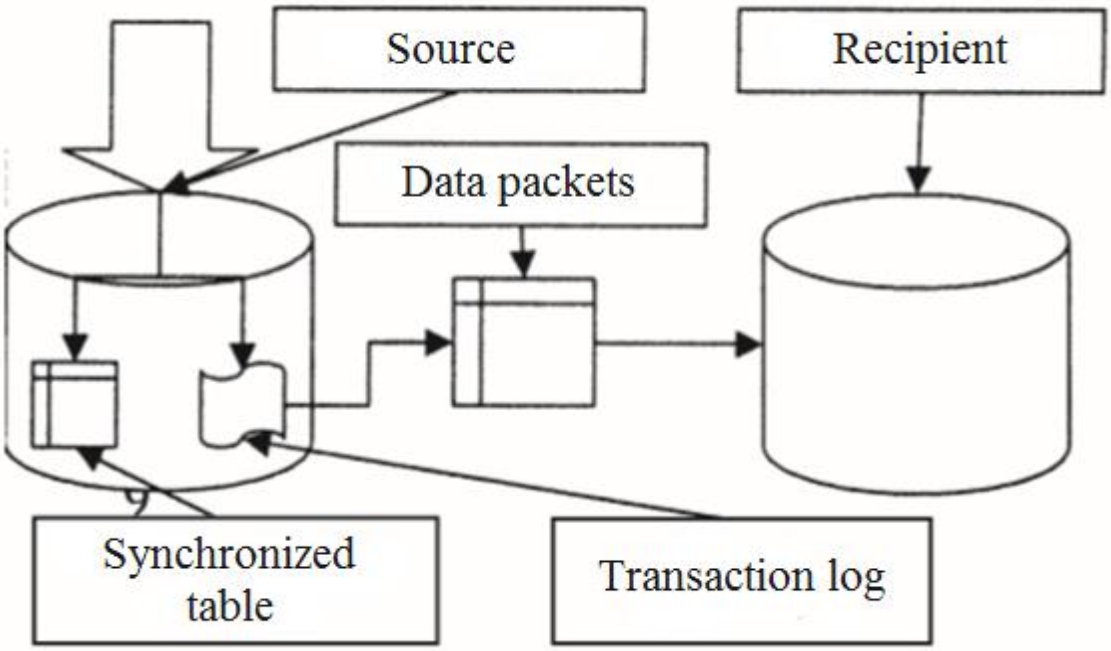


Figure 10. Change detection using a transaction log

Change detection using events or triggers. The system tracks every change that occurs in the database by using events that are tied to each kind of commands (insert, edit, delete). In Figure 11 shows an example implementation of this method.

Advantages:

1. Work with third-party applications. This means that the change made in a modifiable table any software tool will be incorporated into the Protocol changes.

Disadvantages:

1. You support mechanism triggers (events) that it is not always implemented by the DBMS for personal use.

2. Require modification of an existing database structure, which is especially difficult if the trigger mechanism has already been used for other purposes (have to combine the old with the new triggers code).

3. Cannot be auto complete data synchronization (transfer tables entirely if your synchronized tables on node is empty).

4. Possible strong expansion of an existing database by storing additional data.

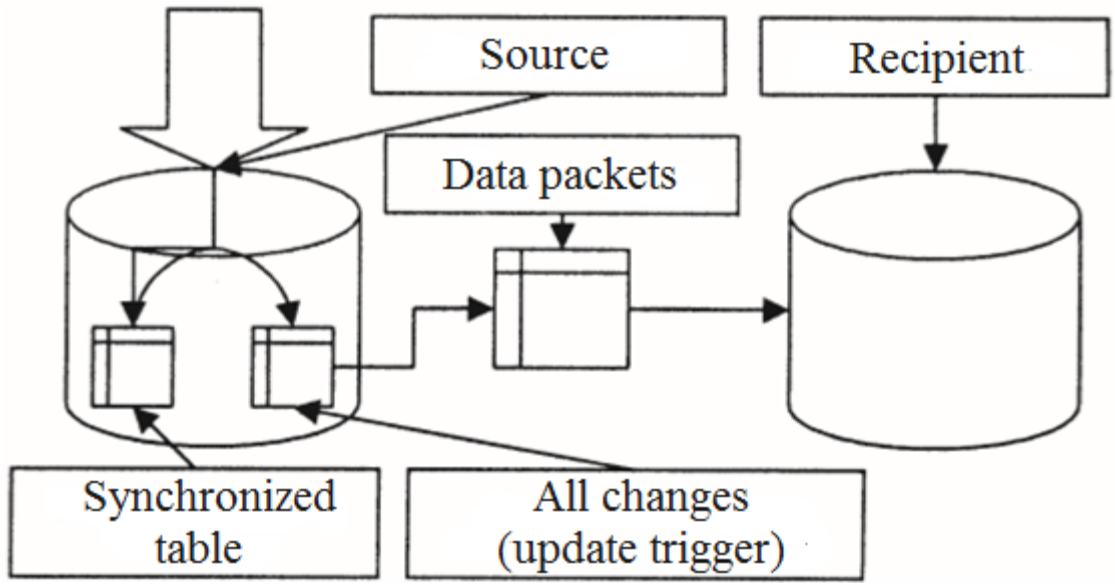


Figure 11. The method based on triggers

Change detection using control tables at first glance looks like a dual method of copying tables. For each replicated table start supplement (control) table for each record in the table indicates some control information, which includes time data changes, as well as the update source. Time changes for each record is automatically updated by triggers. Absolutely necessary for this operation is to trigger the update. Triggers on the creation and deletion are not mandatory, but their presence will prevent the use of additional tools to clean the holdout tables of information about deleted records, and will provide an opportunity to obtain more accurate data on time changes. This method is actually an improvement method triggers or events. It has almost the same advantages and disadvantages as the triggers. Advantage of this method is that it does not create unlimited growing event queue [12].

2. Using ContextMap notation as a solution

The methodology is needed for both relational (structured) and XML (semi-structured database schema), as well as direct business analysts and database administrators throughout the design process.

ContextMap originally called jMap (connected on the map), was first introduced by Dr. WM Jaworski. The technology was originally developed to

recover and recycle knowledge from legacy systems. Using the concept of the structure of the table, it is possible to describe and process information concept. Various information can be easily integrated into a single consistent map using notation ContextMap, thus it can be considered to be a kind of technology of high level designations. Using the notation ContextMap allows effective recovery and modeling of common schemes. This technology can be applied in many areas such as modeling, mining, evaluation contexts, methods, processes, designs, artefacts, databases, Web sites, information systems, models of knowledge with common templates, domain experts and proprietary designations technology.

Methodology ContextMap has formal, flexible, Extensible syntax and notation that can effectively recover and model common process diagrams, objects and views in these systems.

The main elements of the ContextMap tuple context that is shared by many members of the Association, cast in roles. ContextMap can consist of an unlimited number of contextual tuples.

Although the ContextMap can be implemented, deployed and used in a number of different methods, computer programs, databases, modelling and methodology, etc., for the sake of a simplified rendering is usually represented as an extended table [4].

3. ContextMap paradigm and technology

Site www.gen-strategies.com built by Dr. m. Wojciech Jaworski contains much useful information about maps, including maps evolution Context context. Historically, this technology was originally developed as a means of extracting and processing traditional knowledge system. In the late 1970 and early 1980 's, he was named as an array-oriented language based on conceptual graphs, imposed by f. Sova. In the late 1980 's, it was renamed the ABL/W4 (W4 specifies the value, what, when, where and what). In the early 1990 's, given the existing notations and methodologies, Professor Jaworski first introduced the

notion of ContextMap and this technology as ContextMap, namely articulated map. Still, context Maps cannot represent assets of knowledge in 3 p-capable format (process-able, the plugin can and paintings-able).

The context Cards introduces the concept of creating a style sheet for the knowledge-based management of access to information and navigation. It represents the relationships between the various hubs in vertical table columns. In a technical sense, context Maps describe the information specified formally declaring the theme and associating relevant pieces of information, as set out in relevant topics [4].

In connection with a particular context property maps, possible and efficiently describe and handle the conceptual information, using the popular notion of the structure of a table. And by applying the logic of the query tool with the structure of the table, specific information on the application to expect to find the map you can quickly acquire. So context is a collection of maps of various information connected together in matching aid and its technology is very powerful.

In practice, using this new technology, source code can be expressed clearly and read the context table Maps notation. In this report, some typical designs will be showcased.

4. ContextMap syntax

The syntax ContextMap is based on the relationship-oriented paradigm defined with respect to sets (concepts) and select users. Shortcut maps, relationships are represented by kTuples (i.e., vertical bars on the map). The kTuple consist of a set, a member of the Set, and the role of tuples. This design is the fundamental structure is defined by the concept and cases associated with the roles.

Regarding the mechanism is implemented through the allocation of roles in the scheme and sets their copy to install the components on the map. Compared with charts, maps are very compact and offer a rich context within the limited

space of the computer screen. Maps are created or edited in an organized electronic sheet (MS Excel spreadsheet) that ensures the effective manipulation of relationships (columns) and heavy reuse of components (rows).

Figure 12 is a sample that demonstrates how to represent state machine "Inception" phase in the context of the Unified Process Maps. At this stage, at each stage can be transferred to subsequent stage after achieving all tasks associated with this stage. Left figure illustrates context Maps Displaying document flow in the early stages, and the right side shows graph Photo document management in the early stages.

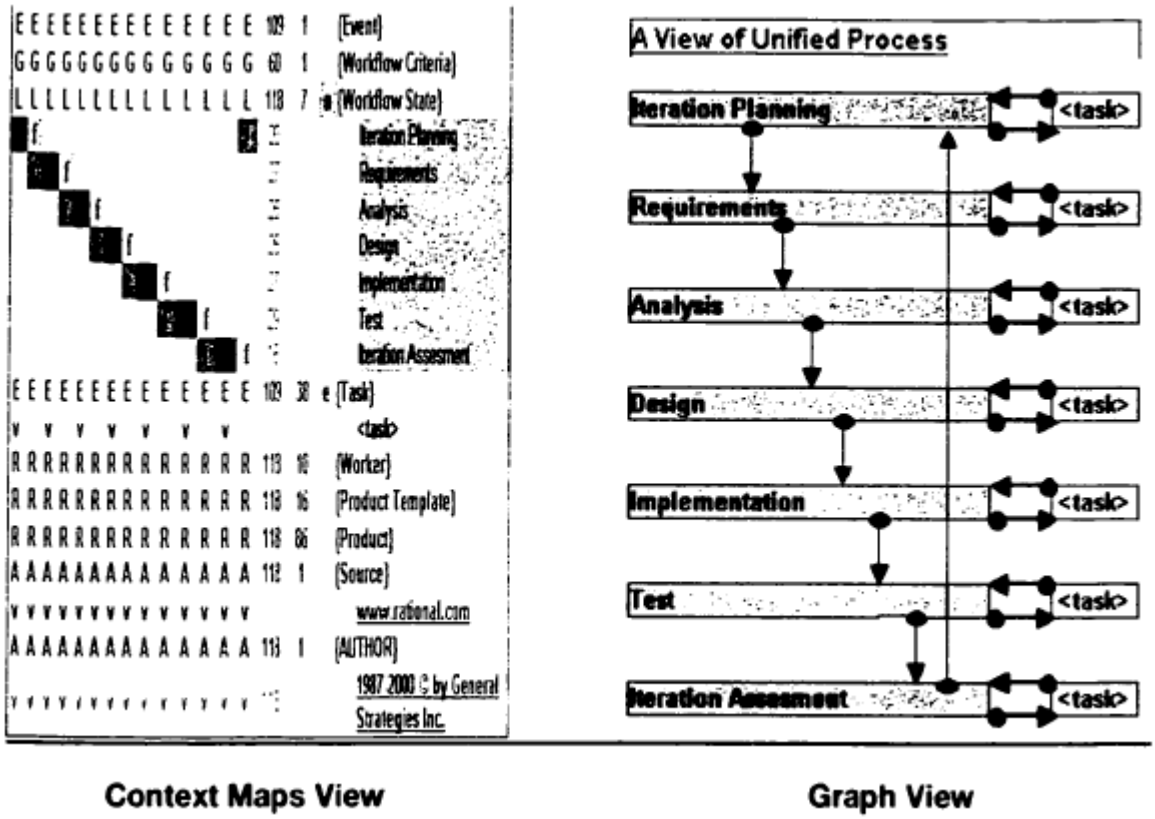


Figure 12. Transition of ContextMap schema to the graph diagrams

Compared to the context of Maps displaying Graph Show in Figure 9, sets (iteration planning, requirements analysis, design, implementation, testing, estimation of iteration) can be represented as "specified users" in the right column under the name Set {procedure of State}, and arrow keys can be represented as a set of "role" column Member 1-14 tagged with lowercase "e" t

"," l ". terminology and symbol from the context Cards are entered in the Card Context 2.3 rules and constrains and Context Maps 2.5 notation in detail.

In the context of the map view, the view layer diagram provides information about the Context of the map structure and size. This card contains multiple sets, namely {Event}, {Workflow Criteria}, {public Procedure} and {} tasks. There are some set Role is "e", "G" and "L" and the four roles of participants, namely, ' V ', ' l ', ' F ' and 't '. Select the role of the "e" was selected in {Event} and {} task. {} Task Clustering allows columns with members the role of "l", "F" and "t" highlights underneath. Select "G" role was allocated to {} Workflow Criteria [5].

If users need to develop great model Context maps, they can hide unwanted rows and columns, editing the visible cells and inserting new columns and rows of new.

In General, appear to the right of the card between the bold braces. They can be thought of as a table or column header row. Roles can be actual content shown in the table. Each column should be read vertically using the syntax. For each Value in the table, the user can read or down columns, and in relation to the right part of the map to find out what Role and a set of "value" is referring to. You can also obtain the schema from the context Cards hiding set members and irrelevant columns and can get useful information by applying a query tool.

5. Conditions of the notation ContextMap

Context conditions Maps used in the present report include definitions, acronyms and abbreviations.

Appendix 1 will be used to explain the terminology Context Maps.

- Context: Maps represent the relationships between the various information kits and provide array functionality, graphics, tables, relational, etc.
- JMaps: abbreviation of articulated map. Previous context name Maps.

- + CONTEXT: a set of tools that was designed to handle context Maps.
- Context tuple: General Association many members of the cast in the roles. In the extended table column roles and associated members set define the context of a tuple.
- KTuple: an abbreviation of a tuple of knowledge. It consists of a set, member recruitment and role of a tuple, has its own schema and contains the ID, type, and handles.
- Schematic: Work framework context Maps. The context Cards integrate concepts and concept instance with abstract architecture.
- Set Between bold {} in column 17, e.g. {Event}, {Procedure State} and {} task.
- Install status: Members below bold {} in the most right column, such as "iteration planning", "requirements" and "analysis".
- Place the cast: capital letters in the cells of a table, such as the letter "e", "G" and "L".
- The role of recruitment of participants: capital letters, or Digitals in cells of a table, such as "e", and "t" and "L".
- Power roles: Column counts the number of blank 15 roles in each row
- The cardinality of the Party: 16 Column counts the number of set members within each Set.
- Atom: All in the right column: specify a name value in {SET}.
- Table: electronic page in MS Excel, this is used for storing and processing data in the context of Maps.

Designation of the ContextMap notation

The context Cards graphic technologies to determine common schema visualization and modeling of information systems. Designation Card context is

an important element in this technology. It can be widely used in various fields such as:

- Architecture of information system
- Restoration and reuse templates system
- Development of information systems
- Assessment and software update
- Automated design systems
- Modeling Web sites and centres of excellence
- System workstations

Context symbol cards can be illustrated by the following map:

In practice, some characters can be given meaning and necessity, different users can identify themselves. Certain rules are flexible, but easy understanding is more important [5].

6. Finding the differences between the old and new data

Cluster analysis is a way of grouping of multidimensional objects based on a view of the results of the individual observations points suitable geometric space with subsequent allocation groups as "bunches" of these points (clusters of taxa). Cluster (cluster) in English means "clot", "bunch of grapes", "cluster of stars, and so this method of study has evolved in recent years, with the possibility of computer processing of large databases.

Cluster analysis involves the selection of compact, distant groups of objects, searches for "natural" together, splitting the field accumulation of objects. It is used when the source data is represented as matrices of proximity or distance between objects or as points in a multi-dimensional space. The most common second type of data for which a cluster analysis is focused on highlighting some geometrically remote groups within which objects are close.

Select the distance between the objects is the nodal point of the study, it largely depends on the final version of the split objects in classes with this algorithm.

There are a large number of algorithms of cluster analysis, you can divide them by way of the clusters are built on 2 types: standard and benchmark. In the procedures of the reference type on the set of objects you specify multiple source zones, from which starts the algorithm. References can be an initial split into classes, the Centre of gravity of the class, etc. After setting the standards of the algorithm classifies, sometimes changing standards in a certain way.

Clustering algorithms running on different principle include the hierarchical cluster analysis algorithms, the procedure of cutting, etc [13].

The objective of cluster analysis

Let set I = {I1, I2, ..., In} denotes the n objects. The result of the i-th measurement characteristics denote a symbol object Ij xij and vector Xj = [xij] responds to each set of measurements (for the j-th object). Thus, for many I objects researcher has many measurement vectors X = {x 1, x 2, ..., Xn}, which describe many i. lots of X can be represented as n points in p-dimensional Euclidean space Ep.

Suppose m is an integer less than n. Objective of cluster analysis is to based on the data contained in the set x, break many objects I m clusters (subsets) π1, π2, ..., πm so that each object Ij belonged to one and only one subset of the split and to objects belonging to different clusters were heterogeneous (different).

The objective of cluster analysis is to break that satisfies some condition of optimality. This criterion may constitute some functionality that expresses the desirability of various levels of splits and factions. This functionality is often referred to as the target function. Objective of cluster analysis is to optimize, i.e. finding a minimum target function with a given set of constraints. An example of the target function can serve, in particular, the sum of the squares of the deviations of intra-group on all clusters [13].

Consider the basic ways to determine distances between objects.

Metrics for the quantitative scale (distance).

a) linear distance

$$d(X_j,X_i)=\sum_{k=1}^{N}| x_{ki} - x_{kj} |;$$

b) Euclidean distance

$$d(X_j,X_i)=\left[\sum_{k=1}^{N}(x_{ki} - x_{kj})^2\right]^{1/2};$$

c) generalized Minkowski distance (metric)

$$d(X_j,X_i)=\left[\sum_{k=1}^{N}(x_{ki} - x_{kj})^p\right]^{1/p}.$$

Note that the Euclidean distance (and its square) is calculated according to the source, rather than standardized data. This is a common technique when it is evaluated, which has some advantages (for example, the distance between two objects does not change with the introduction of the new object in the analysis, which could be the release). However, the distance can dramatically impact the differences between the axles, which coordinates these distances. For example, if one of the axes of measured in centimeters, and you then translate it in millimeters (multiplying the value by 10), final Euclidean distance (or the square of the Euclidean distance), calculated using the change much, and, as a consequence, the results of cluster analysis may be very different from previous ones.

Conclusion

The advantages of data synchronization were presented and evaluated. As well as several other measures were discussed to find differences in proximity between old and new data. On the basis of the deficiencies cited data synchronization techniques, the model was proposed based on the notation ContextMap. The advantage of using the ContextMap notation is a representation in a lightweight form of the data stored in XML documents.

CHAPTER III. DATA SYNCHRONIZATION ALGORITHM AND EXPERIMENTAL RESULTS

Data synchronization method steps when migrating data:

1) transforming XML document into the ContextMap scheme;
2) transforming table from the target relational database into the ContextMap scheme;
3) comparing two ContextMap schemes and when change is detected between schemes, particular action (INSERT, DELETE, UPDATE) occurs in the table of the target relational database;

1. Algorithm of converting an XML document into the ContextMap scheme

The algorithm of extracting data from XML and loads metadata and their values in the ContextMap scheme. Suppose X is the incoming XML file and (C) the individual tags in the XML file.

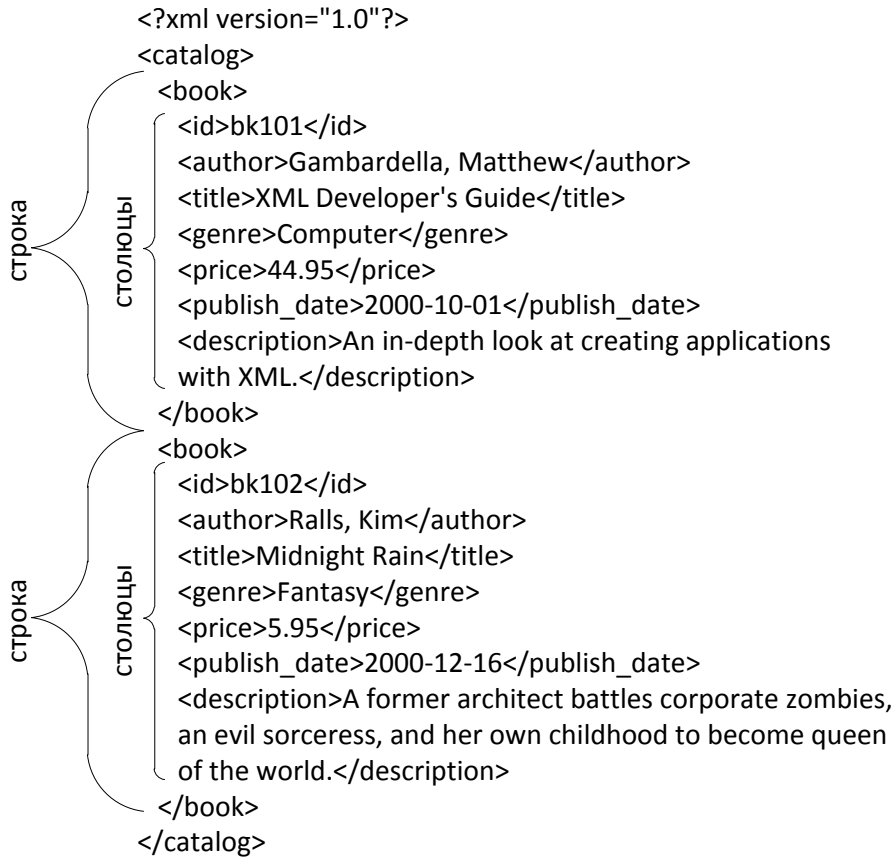


Figure 12. XML document

This is part of the XML file as an example to show the columns and rows.

Pseudocode:

Input: X, C/* XML file, individual elements */

Output: ContextMapXML file

In this file

Insert entity tag under the "Entities" and the number of lines

Cycle from 1 to N where N is the number of columns

Insert name (field) CN tag under "Data" and a symbol of the relationship between the entity, as well as insert number of characters

Cycle from 1 to M where M is the number of rows

Insert tag values and relationship between tag (field), as well as insert number of characters

The end

Description of the algorithm

To convert the XML schema to the schema ContextMap algorithm, we must complete all the steps in detail:

Let's say we have already exists ContextMap scheme and it contains "Entities", "Fields" as well as "F", "t" set of roles and relationships. A set of roles and relationships are listed in the annex.

In the beginning, load <Catalog> parent tag name as the name of the entity in the organizational unit "Entities" and label the symbol "v" to link the fields between entity. Number of characters so much how many lines in the XML file and is specified in a special designated cell.

Then, perform a cycle is equal to the number of unique columns. In the above example, the number of unique columns total 7 (id, author, title, genre, price, publish_date, description).

Inside this loop, insert a title tag (columns) and denoted by the symbol "F" to link the fields between the data in special reserved seats under the Division called "Data".

As well, within that there is another urban cycle which should run as long as total rows. The number of rows is determined by the number of tags <book>. In this case, the number of rows only 2.

Inside the loop, inserting the value tag (id, ...) and label with the symbol "t" to indicate membership in the value fields. This algorithm ends its work.

The result of the algorithm is the ContextMap diagram below.

Table 2. The result of the algorithm of converting an XML document into the ContextMap scheme

			ENTITIES
v	v	2	Catalog
			DATA
F	F	2	**{id}**
t		1	bk101
	t	1	bk102

F	F	2	**{author}**
t		1	Gambardella, Matthew
	t	1	Ralls, Kim
F	F	2	**{title}**
t		1	XML Developer's Guide
	t	1	Midnight Rain
F	F	2	**{genre}**
t		1	Computer
	t	1	Fantasy
F	F	2	**{price}**
t		1	44,95
	t	1	5,95
F	F	2	**{publish_date}**
t		1	01.10.2000
	t	1	16.12.2000
F	F	2	**{description}**
t		1	An in-depth look at creating applications with XML.
	t	1	A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

2. Algorithm of converting a table from the target relational database into the ContextMap scheme

The algorithm of extracting data from the target table of a relational database and loads data and attributes in schema ContextMap.

Table 3. The target table of a relational DB

id	author	title	genre	price	publish_date	description
bk103	Corets,	Maeve	Fantasy	5,95	17.11.2000	After the

	Eva	Ascendant				collapse of a nanotechnology society in England
bk104	O'Brien, Tim	Oberon's Legacy	Horror	36,95	10.03.2001	A deep sea diver finds true love twenty thousand leagues beneath the sea.

Pseudocode:

Input: A, D/* set of attributes and data */

Output: ContextMapRDB file

In this file

Insert the name of the table under the "Entities" and the number of lines

Cycle from 1 to N where N is the number of columns

Insert name (field), AN attribute of "Data" and a relationship between an entity, as well as insert number of characters

Cycle from 1 to M where M is the number of rows

Insert attribute value of DM and symbol of the relationship between an attribute (field), as well as insert number of characters

The end

The end

The end

The algorithm is similar to the first algorithm, but serves other input values. Following is the result of the algorithm.

Table 4. The result of the algorithm of converting table from the target relational database into the ContextMap scheme

			ENTITIES
v	v	2	Catalog
			DATA
F	F	2	**{id}**
t		1	bk103
	t	1	bk104
F	F	2	**{author}**
t		1	Corets, Eva
	t	1	O'Brien, Tim
F	F	2	**{title}**
t		1	Maeve Ascendant
	t	1	Oberon's Legacy
F	F	2	**{genre}**
t		1	Fantasy
	t	1	Horror
F	F	2	**{price}**
t		1	5,95
	t	1	36,95
F	F	2	**{publish_date}**
t		1	17.11.2000
	t	1	10.03.2001
F	F	2	**{description}**
t		1	After the collapse of a nanotechnology society in England
	t	1	A deep sea diver finds true love twenty thousand leagues beneath the sea.

3. Comparison of schemas and change detection

The comparison is also one of the key in data synchronization method, because this step is determined by the modified, udelannye and added the data in an XML document.

Pseudocode:

Entry: XML, RDB/* an XML schema XML file ContextMap RDB-target table schema ContextMap relational database */

Output: the SQL file with possible queries, INSERT, UPDATE, DELETE to change the target relational database tables

Cycle for each Ri in R, where R is the sum of columns from the RDB, i in [1 .. n]

Cycle for each Dj to D, where D is the set of strings (from the RDB) in the current column j in [1 .. m]

For each cycle Xl in X, where X is the set of strings (of XML) in the current column, l [1 ... k]

If Dj == Xl

Check drawing diagrams and make appropriate changes to the target table of a relational database

End of condition

End of cycle

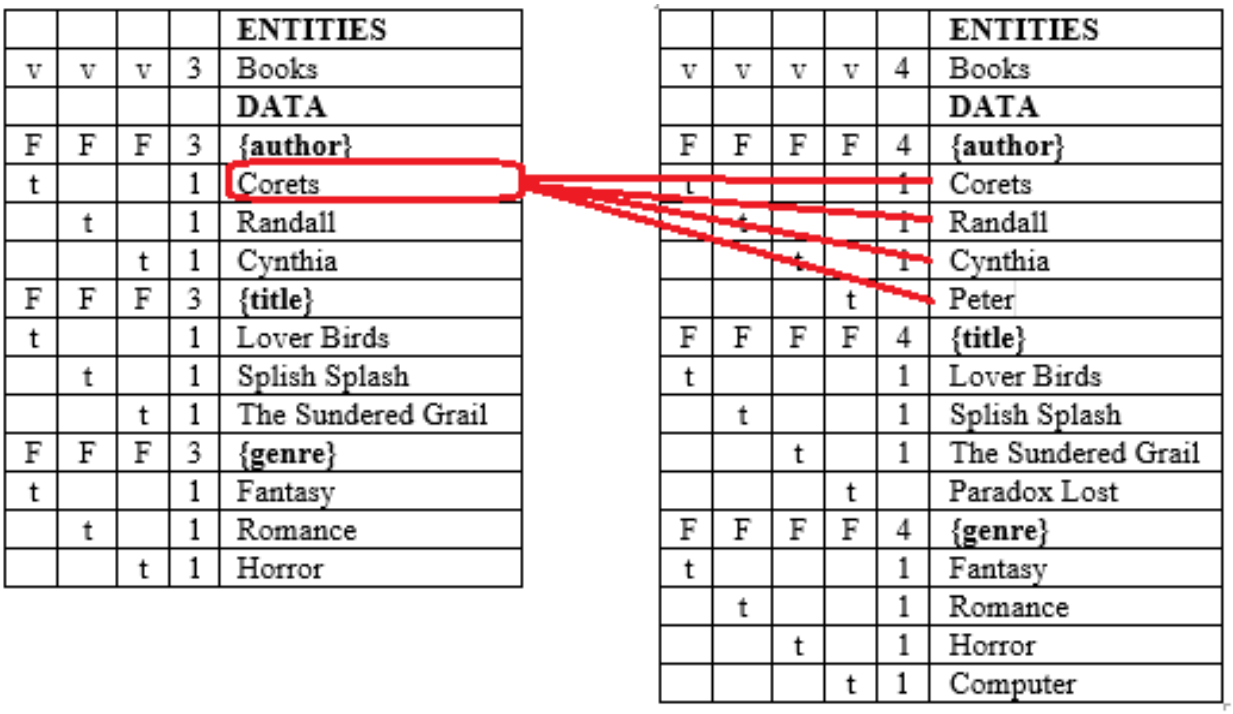
End of cycle

End of cycle

Description of the algorithm

Each row of the RDB is compared with each line of XML in this column must match with each other.

For starters, the algorithm reads the number of columns of the papers (depends on the XML schema of the document conforms to the schema of the target table). In each column, each row of the RDB is compared with each line of XML.

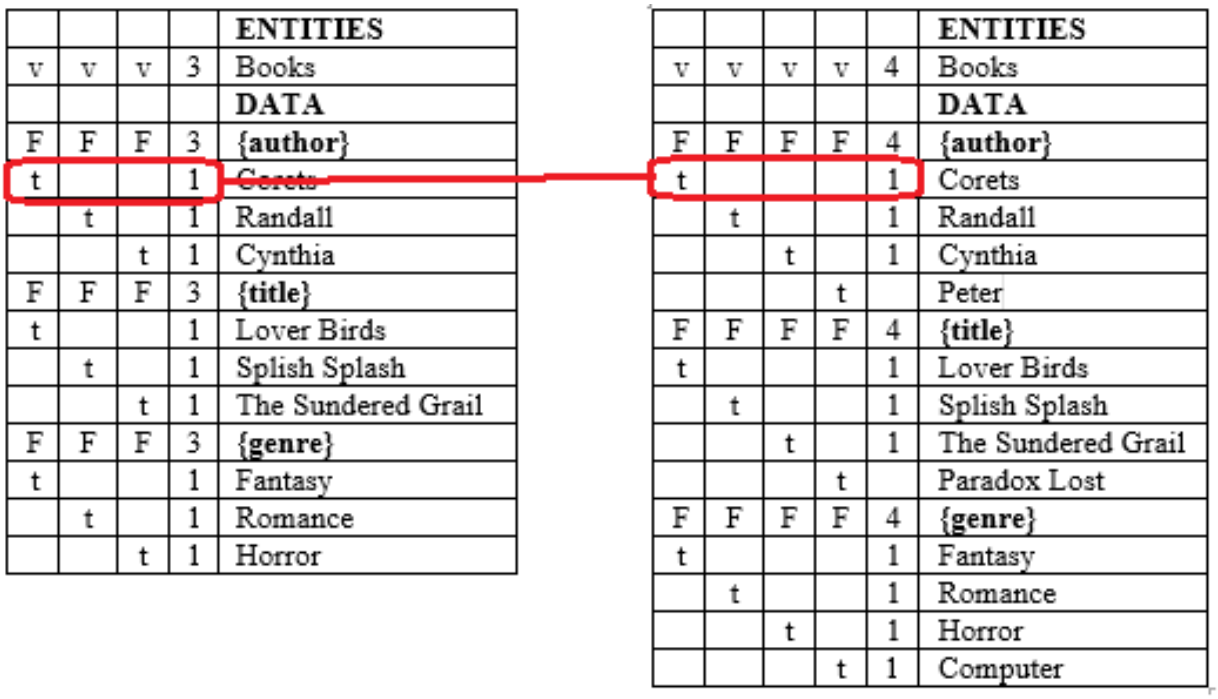


				ENTITIES
v	v	v	3	Books
				DATA
F	F	F	3	{author}
t			1	Corets
	t		1	Randall
		t	1	Cynthia
F	F	F	3	{title}
t			1	Lover Birds
	t		1	Splish Splash
		t	1	The Sundered Grail
F	F	F	3	{genre}
t			1	Fantasy
	t		1	Romance
		t	1	Horror

					ENTITIES
v	v	v	v	4	Books
					DATA
F	F	F	F	4	{author}
t				1	Corets
	t			1	Randall
		t		1	Cynthia
			t		Peter
F	F	F	F	4	{title}
t				1	Lover Birds
	t			1	Splish Splash
		t		1	The Sundered Grail
			t		Paradox Lost
F	F	F	F	4	{genre}
t				1	Fantasy
	t			1	Romance
		t		1	Horror
			t	1	Computer

Figure 13. Comparing columns of ContextMap schemes

When it finds a string of XML that contains the same data from the RDB, the algorithm starts to compare the schema itself ContextMap model.



				ENTITIES
v	v	v	3	Books
				DATA
F	F	F	3	{author}
t			1	Corets
	t		1	Randall
		t	1	Cynthia
F	F	F	3	{title}
t			1	Lover Birds
	t		1	Splish Splash
		t	1	The Sundered Grail
F	F	F	3	{genre}
t			1	Fantasy
	t		1	Romance
		t	1	Horror

					ENTITIES
v	v	v	v	4	Books
					DATA
F	F	F	F	4	{author}
t				1	Corets
	t			1	Randall
		t		1	Cynthia
			t		Peter
F	F	F	F	4	{title}
t				1	Lover Birds
	t			1	Splish Splash
		t		1	The Sundered Grail
			t		Paradox Lost
F	F	F	F	4	{genre}
t				1	Fantasy
	t			1	Romance
		t		1	Horror
			t	1	Computer

Figure 14. Picture comparison of ContextMap schemes

When you compare schemas, the algorithm detects the necessary SQL command from a list of possible commands (UPDATE, DELETE, INSERT) to change the target relational database tables. The result of the algorithm is the SQL file.

4. Comparative analysis of the method

There are two possible cases when migrating data: the first cases of this when XML file has only the new data; second cases is when the XML file has an updated or new data and old data that have already been translated into the target table of a relational database.

To compare the full transfer method was selected tables. For a description of this method is shown in the second chapter.

The first type, there is a large difference in migration in both methods. As in the XML file are only new data (that is, without the old data), the comparison method between XML document and relational database target table is not required. Therefore, build schema ContextMap also makes no sense. Therefore, the developed method migrates all data from the XML document (without checking) to the target table of a relational database with the preservation of existing data in the destination table.

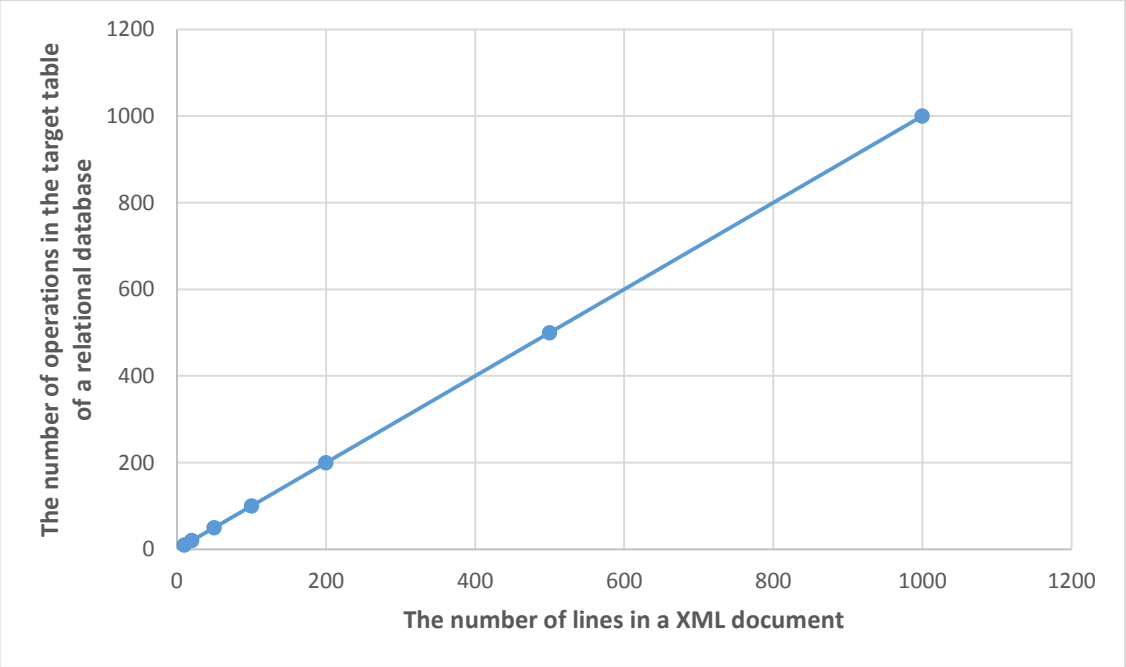


Figure 15. Graphic of a complete data transmission method depending on the number of lines and the number of operations in the target database

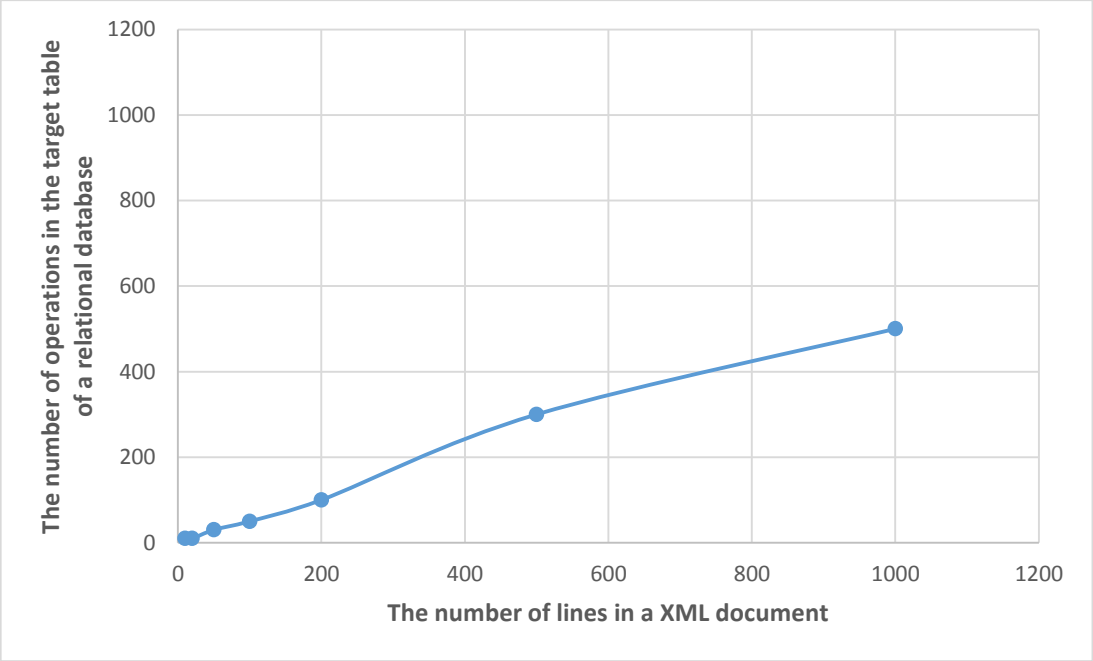


Figure 15. Graphic of the developed method depending on the number of lines and the number of operations in the target database

Method of full transfer tables works on the same principle, that is, without checking the data between the source and the recipient, but without saving the data in the target table, relational database. That is, this method replaces all data that are in the target table, the data that are in the XML document. Delete the existing data from the target table may end disastrously for some corporations or businesses.

In the second type, the efficiency of the proposed method is much higher than than the full transfer method tables. In this type, the method works on algorithm, i.e. creates a ContextMap schemes and compares them to detect modified data in an XML document. By comparison we avoid full data transfer. With the full data transfer takes a lot of time, and the extra cost of computing resources, especially when large quantities of data. Because the method of complete transfer tables on this principle, the proposed method is passed only

the data that has been modified or added, thereby reducing run time data migration and saving computer resources.

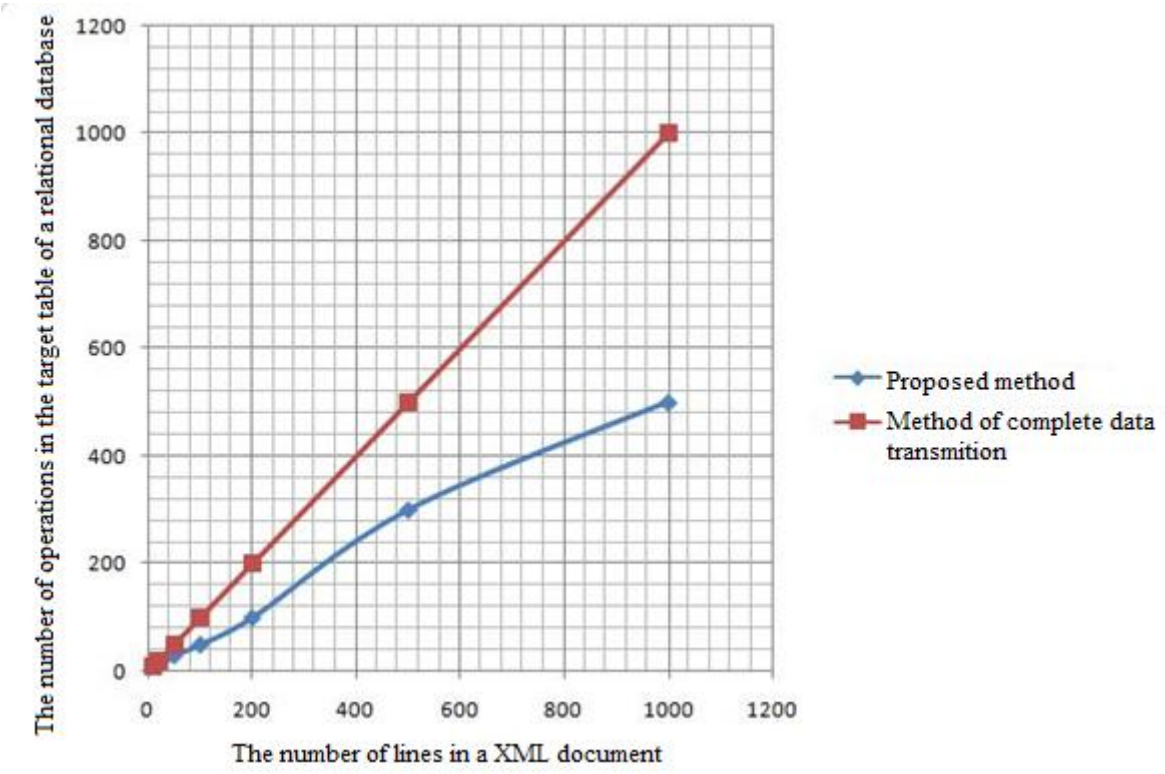


Figure 16. Graphic comparing two methods

In the graphs where the x axis represents the number of rows in the XML document, and the y axis is the number of operations that must be executed on the destination table of a relational database. If at the beginning of the XML document contains 10 rows records, let's say all these 10 rows don't have records in the target table. In this case, the method will detect that these 10 rows not in the table, and writes them to the destination database. Full table transfer method also writes all the data. The next time if in the XML document only 20 rows, then the method writes the 10 new rows of them in a relational database, when as a method of full transfer tables writes all lines, there are 20 rows.

Conclusion

A data synchronization method was developed which is based on building ContextMap schemas, and comparing them to identify changes between XML document and target table in the relational database. A comparative analysis is

proposed with a method of full transfer tables. Advantages of the developed method are identified, such as relatively fast execution of data migration by keeping resources of a computer or computer nodes.

FINAL CONCLUSION

The work is dedicated to the development of a method for data migration from XML into a relational database. The following problems were identified on the basis of the conducted analysis of software products on the market and data migration techniques:

1. Lack of expertise in data migration
2. Weak understanding of data in source systems
3. Changing target system
4. Insufficient quality of migrated data
5. Data synchronization

It was revealed that the problem of data synchronization occurs in front of IT companies almost every day. In particular, this task can arise when you automate a company with territorially remote branches or in case when some

regulatory governmental organization maintains a set of references, each of which must be present in the database of companies reporting to the governmental organization. As a result, it was determined that existing methods of data migration are not effective enough to solve data synchronization problems.

As a solution, it was proposed to use the ContextMap notation for converting XML document and relational database to the middleware format, i.e. ContextMap format. The proposed method consists of 3 steps, for each of which has developed its own algorithm. The first two conversions from XML and relational database to the ContextMap schema.

A comparative analysis was made with the method of full data transfer within two possible cases:

- XML document contains only new data
- XML document contains old and new data

In the first case, the proposed method outperformed in terms of speed, as first full data transfer method removes all data from a target database and records all possible data from XML document into a target database, whereas the developed method records only new data to a target database by storing old information.

In the second case, the performance of the proposed method far exceeds epy performance of the method of full data transfer. In this case, the developed method works within an algorithm, i.e. creates a ContextMap schemes and compares them to detect modified data in an XML document. Due to the comparison, full data transfer is avoided. With a full data transfer, it is spent a lot of time, and also it occupies enormous computer resources especially with large data.

The experiment proved that the developed method based on the ContextMap notation meets stated objectives and more effectively migrates data compared to the analogue of this method, and the developed method also recommended to use in the developing enterprise systems.

REFERENCES

1. Постановления Президента Республики Узбекистан от 21 марта 2012 года «О мерах по дальнейшему внедрению и развитию современных информационно-коммуникационных технологий»
2. Постановление Кабинета Министров Республики Узбекистан «О мерах по организации деятельности Центра развития системы «Электронное правительство» и Центра обеспечения информационной безопасности при Государственном комитете связи, информатизации и телекоммуникационных технологий Республики Узбекистан»
3. Постановление №ПП-2158 Президентом Республики Узбекистан 3 апреля 2014 года «О мерах по дальнейшему внедрению информационно-коммуникационных технологий в реальном секторе экономики»
4. Sanaz Rahmati (2006). “Converting relational database to XML schema and vice versa using ContextMap”. Concordia University, Montreal, Quebec, Canada.
5. Kang Zhou (2002). “CONTEXT+: Development environment for 3P-able context maps”. Concordia University, Montreal, Quebec, Canada.
6. Converting XML to Relational Data. DB2 pureXML Cookbook. Copyright 2010 by International Business Machines Corporation.
7. Andrew Charles Smith. Translating between XML and Relational Databases using XML Schema and Automed. University of London and for the Diploma of Imperial College of Science, Technology and Medicine, September 2004.
8. Константин Баканович (2011). “Миграция данных: что проще?”. “Storage News” № 4 (48), 2011, www.storagenews.ru.
9. Важдаева Ксения Сергеевна. Исследование проблемы миграции данных, хранимых в реляционных СУБД, и реализация сопутствующего программного продукта. Томский государственный университет, 2010.
10. Яковлева Екатерина Петровна. Механизм миграции данных, хранимых в реляционных СУБД. Томский государственный университет, 2010.
11. Важдаева Ксения Сергеевна. Приложение для миграции данных, хранимых в реляционных базах данных. Томский государственный университет, 2012.

12. Н.С. Рябков (2006). “Аналитический обзор методов репликации и синхронизации БД”. Информационные технологии в менеджменте качества и инновационном менеджменте №4.
13. Д.Ю. Кузнецов, Т.Л. Трошина. Кластерный анализ и его применение.
14. Кластерный анализ. Stat Soft - электронный учебник по статистике. http://www.statsoft.ru/home/textbook/modules/stcluan.html
15. Synchronization (computer science) – Wikipedia: https://en.wikipedia.org/wiki/Synchronization_(computer_science)
16. Data migration strategies – Wikibon: http://wikibon.org/wiki/v/Data_migration_strategies
17. Data migration – Data Integration Info: http://www.dataintegration.info/data-migration
18. Data migration – Halogence: http://www.halogence.co.uk/whatIs.html

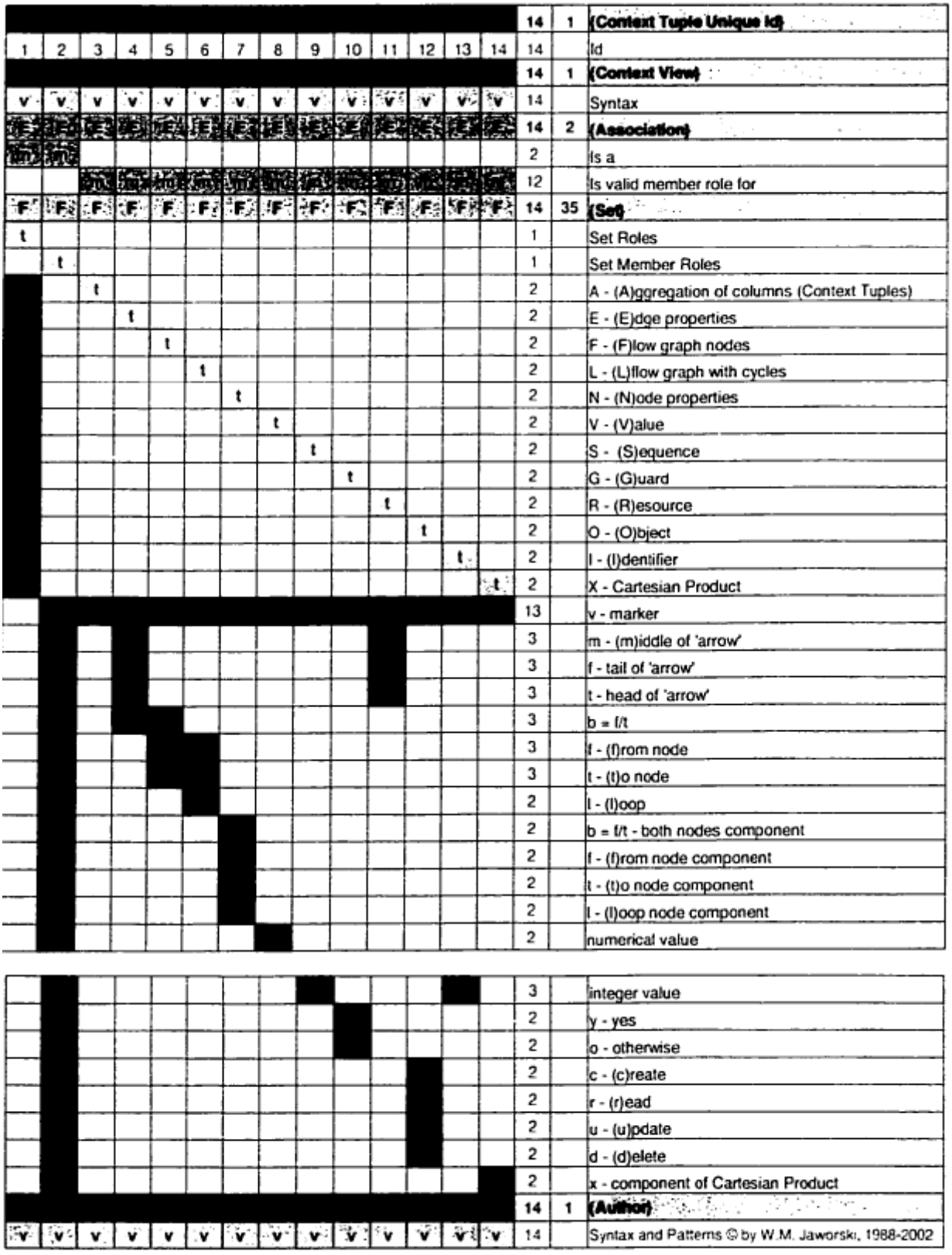
APPENDICES

Appendix 1: The ContextMap notation roles

Category	*Name*	*Description*
Notation of Set Roles	A	(A)ggregation of columns - context tuples
	E	- (E)dge properties
	F	- (F)low graph nodes
	L	- (L)flow graph with cycles
	N	- (N)ode properties
	V	- (V)alue
	S	- (S)equence
	G	- (G)uard
	R	- (R)esource
	O	- (O)bject
	I	- (I)dentifier
	X	- Cartesian Product
	?	- unknown
Notation of Set Member Roles	v	- marker
	?	- unknown
	m	- (m)iddle of 'arrow'
	f	- tail of 'arrow'
	t	- head of 'arrow'
	b	- both f/t
	F	- (f)rom node

Category	*Name*	*Description*
	t	- (t)o node
	l	- (l)oop
	b	- f/t - both nodes component
	f	- (f)rom node component
	t	- (t)o node component
	l	- (l)oop node component
	y	– yes
	o	- otherwise
	r	- (r)ead
	u	- (u)pdate
	d	- (d)elete
	x	- component of Cartesian Product
	c	- concurrence
	j	- (J)oin from fork
	k	- (K)ey

Appendix 2: The ContextMap notation schema



														14	1	(Context Tuple Unique Id)
1	2	3	4	5	6	7	8	9	10	11	12	13	14	14		Id
														14	1	(Context View)
v	v	v	v	v	v	v	v	v	v	v	v	v	v	14		Syntax
														14	2	(Association)
														2		Is a
														12		Is valid member role for
F	F	F	F	F	F	F	F	F	F	F	F	F	F	14	35	(Set)
t														1		Set Roles
	t													1		Set Member Roles
		t												2		A - (A)ggregation of columns (Context Tuples)
			t											2		E - (E)dge properties
				t										2		F - (F)low graph nodes
					t									2		L - (L)flow graph with cycles
						t								2		N - (N)ode properties
							t							2		V - (V)alue
								t						2		S - (S)equence
									t					2		G - (G)uard
										t				2		R - (R)esource
											t			2		O - (O)bject
												t		2		I - (I)dentifier
													t	2		X - Cartesian Product
														13		v - marker
														3		m - (m)iddle of 'arrow'
														3		f - tail of 'arrow'
														3		t - head of 'arrow'
														3		b = f/t
														3		f - (f)rom node
														3		t - (t)o node
														2		l - (l)oop
														2		b = f/t - both nodes component
														2		f - (f)rom node component
														2		t - (t)o node component
														2		l - (l)oop node component
														2		numerical value
														3		integer value
														2		y - yes
														2		o - otherwise
														2		c - (c)reate
														2		r - (r)ead
														2		u - (u)pdate
														2		d - (d)elete
														2		x - component of Cartesian Product
														14	1	(Author)
v	v	v	v	v	v	v	v	v	v	v	v	v	v	14		Syntax and Patterns © by W.M. Jaworski, 1988-2002