An Approach for Reducing Failures using Phased Data Migration Projects

S. Geetha
Research Scholar,
Madurai Kamaraj University, Madurai, India.
geeta_baskar@yahoo.in

Abstract - Software organizations often need to migrating applications from one platform or technology to another for a variety of reasons. As part of an Information Lifecycle Management (ILM) best-practices strategy, the organizations require innovative solutions for migrating data among heterogeneous database systems and environments. The software engineering research community is trying to find out techniques by which such migration projects can be carried out efficiently. This paper covers the unique challenges of data migration in dynamic IT environments and the key business advantages that we have designed to provide a new approach for database migration. The present paper proposes an approach called phased migration is used to reduce failures when migrating databases. This paper attempts to draw on the power of genetic algorithms in addressing complex problems. Through an empirical study, we show that phased methodology performs better than Big Bang of migrated databases.

Index Terms— Databases, phased migration, data migration and genetic algorithm

Introduction

Nowadays Database systems have become an essential part of any computer software, almost every computer system be it personal or corporate do have a database system. Data migration is the process of making an exact copy of an organization's current data from one device to another device—preferably without disrupting or disabling active applications—and then redirecting all input/output (I/O) activity to the new device.

Due to advancement in the technology newer version of secure database have been developed to transfer the data to the newer version or to an upgraded databases. Basically during the migration process, data is extracted from the old system and loaded to the new system.

The present paper presents an attempt of exploiting the power of genetic algorithms in aiding migration of data from one database to another. The rest of the paper is organized as follows: Section 2 reviews the related literature in the area; sections 3 and 4 introduce the general ideas of developing migration methodology and genetic algorithms respectively. The proposed approach is outlined in section 5 and the results of an empirical study with the proposed approach are presented in Section 6. Section 7 concludes the work and suggests directions for planned future work in the area.

Related work

Data migration techniques have always been a topic of research. Many reorganization techniques have already been proposed [2] [3] and the research on many is on the way. Basically, there are two different approaches for data migration. A Big-Bang data migration is where an entire dataset is moved from source to target systems in one operation whereas phased migrations the data will be migrated in smaller increments until there is nothing left to move.

One way to solve the data migration issue is by using the classical data migration technique that uses the locking, insertion and deletion function for processing data migration. The work of Rajesh Yadav et al [1], on recommending improvements to data migrations using big bang approach is directly related to the present work. This methodology sometimes failures in one part of the system can cause problems and failures in others [3]. An alternate approach called phased migration is used to reduce failures when migrating databases.

Proposed System

The Phased Migration policies help to reduce the stress and uncertainty related to file data migrations. Phased Migration policies use a clear, programmatic approach with well-defined phases:

- In the initial phase, users continue to have access to file data while a baseline copy, performs an initial baseline copy of the file data from the source to the destination.
- In the incremental phase, Phased migration policies continue to copy file data from sources to destinations using to the schedule you specify, copying new files and updated files, as well as any files that were previously locked during the initial phase or during a previous incremental phase policy run.

- In the final phase, quickly and programmatically prevents users from connecting to the source, performs a short final sync to copy any new, changed, or previously locked files, and then stops sharing file data on the source.

Usually all the three phases execute in parallel since the data extraction takes time, so while the data is being pulled another transformation process executes, processing the already received data and prepares the data for loading and as soon as there is some data ready to be loaded into the target, the data loading kicks off without waiting for the completion of the previous phases.

Genetic Algorithms

Genetic Algorithms that mimic the natural process of evolution to uncover solutions to complex problems have been successfully applied across multiple domains to solve various types of problems, not just optimization problems. The skeleton of a genetic algorithm is as follows:

Initialize population of solutions at random

Repeat

Evaluate fitness of each solution

Select 2 solutions at random based on fitness

Cross over the solutions to generate a new offspring

Mutate offsprings at random

Change the population by replacing the less fit solutions with the newly generated offsprings.

Until termination-criterion-is-met

The challenges to be handled when applying genetic algorithms to solve problems include the selection of a suitable encoding mechanism for solutions, design of a fitness function that can evaluate the fitness of a solution, selection of the type of cross-over to be applied and mutation mechanism to be employed. Also, to be decided are the number of solutions in a generation and the number of generations.

Proposed methodology

The main objective of the research is to exploit the power of genetic algorithms in aiding database migration. The genetic algorithm is capable of learning complex mappings that exist between two databases.

The data transformation stage applies a series of rules or functions to the extracted data from the source to derive the data for loading into the end target. Some data do not require any transformation at all; this is known as direct move or passes through data in technical terms.

An important function of data transformation is cleansing of data that aims to pass only proper data to the target [6]. When different systems interact with each other, based on how these systems store data, there is a challenge in interfacing with each other. Certain character sets that may be available in one system may not be available in others. These cases must be handled correctly or eventually lead to a number of data quality related issues.

- Selecting only certain columns to load: (or selecting null columns not to load).
- Translating coded values: (*e.g.*, if the source system stores 1 for male and 2 for female
- Encoding free-form values: (*e.g.*, mapping "Male" to "M")
- Sorting: Order the data based on a list of columns to improve searching

The task is obviously a non-trivial one as is illustrated by the simple example shown above. The automated translation of a database with millions of such statements is a daunting task. Many existing tools for the task including provide considerable help but still are not completely free of errors and defects [2].

Even in straightforward data integration projects, data mapping is never as simple as connecting the dots. Most mappings include data transformations, because source and target data models are always different. Transformations of legacy data tend to be complex, because some source fields contain multiple values that must be teased out and mapped separately [5]. Furthermore, legacy data will have missing values that can be filled in from other sources. As a result, data mappings are always complex, consisting of many components that must be designed and tested individually before being strung together in a data flow.

The proposed solution for the problem is utilizing *genetic algorithms* for the task. First the system needs to be presented with projects that are available in both the databases. Such projects can be typically obtained as most organizations would have already carried out similar migration tasks in the past. Using these projects the system is trained to learn the mapping. The procedure is very similar to the one proposed by Rajesh Yadav et.al. [1]

Results and Discussion

The proposed approach was applied to sets of 20, 35, and 50 projects. Each of the projects was available in 2 versions – relational and the migrated object-oriented version. The percentages of correct mappings learned by the top 3 fit solutions with 50 projects are tabulated below. To demonstrate the superiority of the proposed approach, the results obtained using the approach proposed by Rajesh Yadav et.al.[1] involving the construction of API Transformation Graphs (ATG's) are also shown.

Comparative Analysis Diagram for Big Bang Vs Phased

The table below presents day estimates for each approach based on a data migration involving 40 data objects.

Table 1-Percentage of correct mappings learned by the 3 approaches

Data Migration Approaches	Big-bang	Generic ETL	Phased Db
Percentage	75.47%	76.62%	78.32%

Table 2 – Comparison of execution times of the proposed method and Big bang method.

Number of Transformations Considered	Execution Time (in seconds) with GA	Execution Time (in seconds) with Big-Bang approach
8	6	7
20	8	8
40	13	22
70	18	30

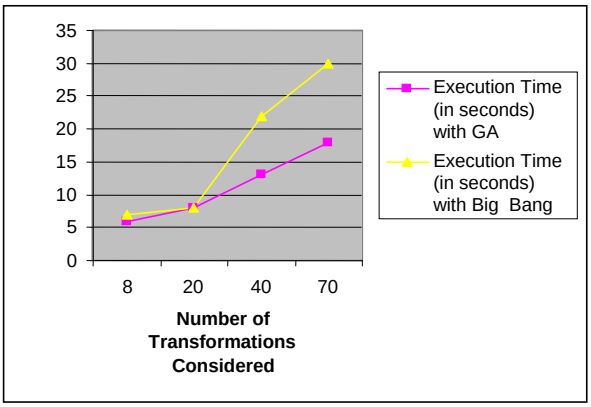

Figure 2 – Comparison of Execution Time with Big Bang

Several observations can me made from the results. First, the performance of the proposed solution is sensitive to the number of projects available for testing the effectiveness of solutions. Second, the proposed approach outperforms the existing approach proposed by Rajesh Yadav et al[1]. This is in line with expectations as one of the arenas in which GA's have been most successful is the learning of complex mappings and the problem fits in this domain. As can be observed, the speed improvement brought about by the proposed approach tends to be more visible with increasing number of transformations considered. This is a positive aspect of the proposed method as in real time the number of transformations to be considered will be huge.

Conclusions and Future Work

A GA based system was proposed for the problem of migration of data from one database to another. Given the rapid technologial changes many more migration projects are likely to arise and hence the development of an effective system for such migration is crucial. The developed system addressed the problem of mapping from relational to object-oriented. More work needs to be done for migration between different paradigms (as from hiererachical to relational).

As a part of future work, the proposed system can be applied to open source applications to improve the credibility of the stated results. Other advancements in Computer Science discipline like neural networks and

natural computation can be applied as it is necesarry that no stone should be left unturned in effors to improve the efficiency of migration tasks.

References

[1] Rajesh Yadav, Prabhakar Patil,Incremental Data Migration in Multi-database Systems Using ETL Algorithm.
[2] Lixian Xing, Yanhong Li, "Design and application of data migration system in heterogeneous database" in 2010.
[3] Data Migration Methodology, Web Qualify, Satyam Computers Services Limited.
[4] Andreas Meier, Rolf Dippolod ,Hierarchical to Relational Database, IEEE Software 1994, vol-IV.
[5] Andreas Behm, Andreas Geppert, Klaus R. Dittrich2, on the migration of relational schemas and data to object-oriented database systems.
[6] Lin, Chang-Yang, Migrating to Relational Systems: Problems, Methods, and Strategies, Vol. 4, No. 4, December 2008.
[7] Data migration best practices et App Global Services, January 2006.
[8] Successful Database Migration by Alex Polishchuk.
[9] Ieeexplore.com: Database migration: A new architecture for transaction processing in broadband networks.
[10] Migrating from generic databases from MYSQL workbench.