World Journal of Computer Application and Technology 3(3): 41-48, 2015 http://www.hrpub.org
DOI: 10.13189/wjcat.2015.030301

A Review on Database Migration Strategies, Techniques and Tools

M. Elamparithi*, V. Anuratha

Department of Master of Computer Applications, Sree Saraswathi Thyagaraja College (Autonomous), India

Abstract In general, Relational Database Migration (RDBM) has always complex, time-consuming, and magnified process due to heterogeneous structures and several data types of RDB. This paper discusses the categorized literature review of existing migration strategies, surveys the translation techniques, and also discusses the technical issues for making the migration process more effective. This review assesses the impact of existing migration strategies and shows how it has been designed according to the recent trends. In software industries, plenty of Database Migration Tools (DBMTs) are available, but finding the effective one is still a risky job. Hence, some of the DBMTs are reviewed and suggested several basic criteria for evaluating migration toolkits. The discussion of these issues will help the researchers and also database practitioner in planning the migration projects.

Keywords Database Migration, Schema Translation, Database Migration Strategy, Migration Techniques and Database Migration Tool

1. Introduction

Migration of Relational Database (RDB) means that the data from source RDB to target RDB, which includes schema translation and also data transformation. Since 1990, several approaches begins with legacy migration, because of growth of information technology many software industries developed their own migration process tool for their essential applications. Earlier Barron C et al., (1974) summarized reasons where the purposes of database migrations are suggested. The core reason for need of migration is upgrading the existing system into developed system according to the industry requirements. The impending problem is redefining of existing database and storage system in terms of complex language. Based on the industry rule, it is common that during the migration, the source and target databases are structurally different or data is inconsistent across multiple data sources. Due to this problem, several researches and development of migration tools are emerged continuously. Joseph R. Hudicka (1998) provided a complete solution of data migration methodology for migration project. In accordance to his view, the entire database migration processes are divided into several phases which are completed one by one. In this methodology, the database migration deals with row counts, columns counts and related statistics to the source database. The disadvantage of this method is that it does not migrated null and numeric values and error has been occurred for key constrain data fields. RDBM has not been subject of broad academic research for past one decade, but due to tremendous advent of Open Source Database Management System (OSDBMS), the database migration workflow in taken into account. Based on the detailed analysis of the existing literature review on migration strategies and techniques, the successful RDBM among heterogeneous databases including Open Source Databases (OSDBs) are still in the initial stage of research. However, this paper examines most of the database migration primitive and recent strategies, approaches, models and toolkits.

The rest of this paper is organized as follows: Section 2 summarizes the impact of primitive strategies such as legacy migration and reverse engineering. Section 3 discusses the different migration strategies and techniques related to Relational Database Migration (RDBM). Section 4 presents the different types of Database Migration Toolkits (DBMTs) and evaluation criteria. Section 5 deals with conclusion and future work.

2. Primitive Strategies

2.1. Legacy Migration

Legacy migration strategy is an important factor during the time of executing an Information System**.** Much of the literature on database migration is somewhat appropriately devoted to legacy migration. Migrating of these systems can

be a time-intensive and extremely expensive task; as a result it is essential for organizations to simplify the migration process and to make it as cost-effective as possible. These systems differ substantially from modern enterprise architectures, since the presentation, business logic, and data access tiers are generally of the same tier. Legacy system migration often includes a great number of research areas like reverse engineering, business reengineering, schema mapping, application development, and translation. A technical report from CMU-SEI (Weiderman et al., 1997) developed an enterprise framework for the legacy migration systems. This acts as a guide for organizations planning software evolution efforts, like migrating legacy systems to more distributed open environments, as shown in Figure 1. A legacy migration life cycle includes the following procedures:

- *Before Migration*: Plan, assess and prepare
  - Assess hardware, software and network readiness and plan for future
  - Clean up by eliminating useless data, consolidating resources, monitoring everything
- *During Migration*: Prototype, pilot and deploy migration
  - Use powerful database modeling to simulate migration, resolving issues before commit
  - Track migration
- *After migration*: Maintain and manage new environment

Legacy migration has been classified into well-defined interfaces, applications, and database services. For legacy migration, user and system interfaces are separate modules, at the same time applications and database services are not separable. Legacy migration strategies are easy to apply, fast to implement, and can be widely applied to industry software projects. An important issue of this system is that it is very difficult to incorporate with newer system such as open source operating systems as having of non-extensibility, incompatibility, and less-openness of the underlying hardware and software of the legacy systems (Bisbal et al., 1999). The major disadvantage is to find and separate business logic from presentation and data logic and also extremely hard to manipulate and retrieve data because of the redundancy. Legacy systems and migrations are still in demand because of their distinct characteristics and good pedigree. However, it is possible to either eliminate or integrate the legacy systems by following effective migration strategy and appropriate migration tools.

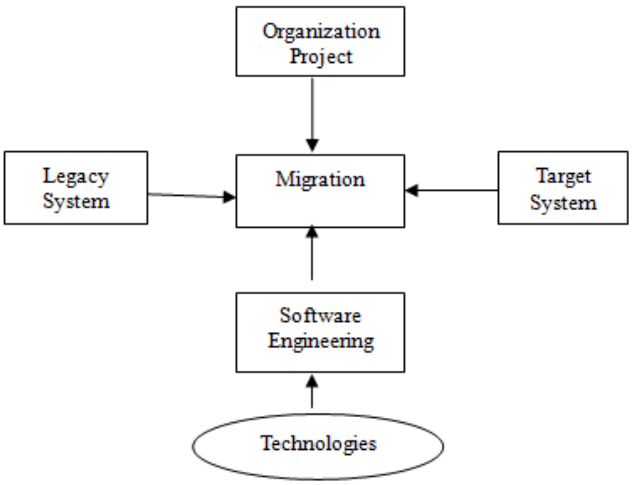

Figure 1. A framework for the legacy migration

2.2. Database Reverse Engineering (DBRE)

Reverse engineering is the primitive technique, which is widely used in software engineering process. It is the first step on migration path through analysis of the source relational database system. It is the process of database code, documentation, and behavior to identify its components and their dependencies to migrate and create the targets system concepts and design information (SEI, 2004). The disadvantage of this process is that the subject system cannot to be modified. In this technique, the source system is considered as the Data Dictionary (Source and Target Databases) and DDL (Data Definition Language). Database Reverse Engineering deal with the migration of source database design specification (Schema information) and tasks of understanding databases.

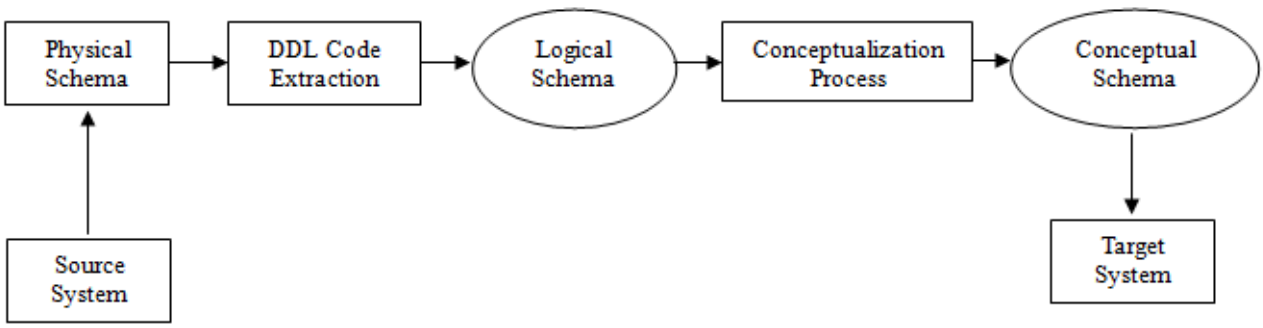

Figure 2. A framework for Database Reverse Engineering (DBRE)

Database Reverse Engineering (DBRE) is often identical with the reverse engineering of the data structure. In software industries, the reverse engineering process is a methodology to analyze all the schema information of the source database system's and achieves a deeper understanding of their internals through the process of DDL Extractions, and converted into logical to conceptual schema for the target database as shown in figure 2. During this process, one has to identify the volume of data to be converted, extract the current data structure (schema) as well as relation between data and system's procedural components, and complete and update documentation fragments. Database reverse engineering provides a valuable analysis toolkit and points useful resources inside and outside the system that have to be examined and questioned. However, reverse engineering data structures is still a complex task. This is the major issue of DBRE. Due to the complexity of data structures, successful migration is still to be examined and for this issue, database analysts are forced to make arbitrary choices, to develop new technique for successful migration.

3. Database Migration Strategies and Techniques

This section introduces different migration strategies and techniques related to database migration. Section 3.1 discusses various strategies to database migration whereas section 3.2 discusses contemporary conversion techniques.

3.1. Database Migration Strategies

To apply progressive database technology, software industries are facing the challenge of migrating data and applications as well. A several database migration strategies exist in the software industries which can present information about database conversion and applications. The following table shows a literature survey of database migration model strategies.

Table 1. Database Migration Model Strategies

Author	Database Migration Model Strategies
Fishman et al., (1987)	Object Oriented Database Management System
Hardwick and Spooner. (1989)	Using Object Technology to Engineering Applications
Wilkinson et al., (1990)	Object Oriented Data Management System
Hainaut, J. (1991)	Database Reverse Engineering Models and Techniques
Crowe. M. K. (1993)	Object System over Relational Databases
Andreas Meier et al., (1994)	Hierarchical to Relational Database
Ian Graham. (1995)	Relational to Object Technology
Monk, S. et al., (1996)	Relational to Object Oriented Database
Andreas Behm et al., (1997)	Relational to Object Oriented System
Yury Bychkov and Jens H. Jahnke (2001)	Legacy Databases to XML
Fong, J. and Cheung (2005)	Relational to XML Database
Abdel Salam Maatuk. Et al., (2008)	Relational to Object Oriented Relational Database(ORDBs)

These approaches are categorized into three basic strategies related to database migrations. First one is for handling data stored in database through OO/XML interfaces. Second one is connecting the existing source relational database to a conceptually different target database system. Third is migrating completely both schema and data in source database to equivalent target database system. The first and second strategies deal only with the schema translation. The advantage of the first approach is that the relational data is still accessible as relational database; the disadvantage is the inefficiency of having translated Data Manipulation Language (DML) commands between the two layers. The second strategy is to implement more of a migration rather than simply to overlay an interface. In this case, relational technology is migrated to objects. The most significant step in this process is to derive an object-oriented scheme from a relational scheme from the existing source system. Due to the substantial investments in many traditional relational databases, part of their data may need to be formatted and implemented in a new different platform. Hence, constructing a gateway interface between the two databases might be preferred. Migrating to a new DBMS might be a good decision in case if the existing system is too expensive to maintain. The impact of the above three basic strategies are summarized in the following subsequent sections.

3.1.1. Strategy 1: Migration through Object Oriented Interfaces

Data may be required to be processed in object/XML form

and stored in relational form based on the concept of object for programs and database for persistence. This process requires object-to/from-database mapping techniques; such mapping is bi-directional on demand of updating a relational database using Object Oriented Interfaces. This is the reverse direction from where object-based schemas are translated into a database schema. While the objects are associated via references, data in the source database tables are linked through the values of primary keys and foreign keys. A single object might be represented with several tuples in quite a lot of tables, and therefore, joining these tables is required for queries. This constant conversion leads to a semantic gap between the two different paradigms. To avoid this, developers have to write large amounts of code to map objects in programs into tuples in a database, which can be very time-consuming to write and execute or to use mapping query systems/middleware, which is a software layer that links OO Programming Languages (OOPLs) concepts to data stored in RDBs through ODBC or JDBC drivers. However, mapping using middleware requires time for schema mapping, on each association that stored data are accessed.

3.1.2. Strategy 2: Database Integration

A connection can be established between source and other target databases which allow the applications built on top of a new DBMS to access both relational and object/XML DBMSs, giving the impression that all the data are stored in one database. This represents a simple level of database integration (Christine Parent and Stefano Spaccapietra, 2000) between systems. This is achieved using a special type of software called *gateways*, which supports connectivity between DBMSs and do not involve the user in SQL and database schema. Hence, queries and operations are converted into SQL and the results are translated into target objects. Many applications use two or more underlying databases. On retrieving data from both systems, the unification of their two schemas is necessary by providing two-way mapping. During integration, systems cooperate autonomously by creating a unified and consistent data view for several databases, hiding heterogeneities and query languages. Most commercial DBMSs such as Oracle, MySQL and SQLServer provide flexibility of mapping and gateways construction among heterogeneous databases.

3.1.3. Strategy 3: Database Migration

Migration of a relational database into its equivalents is usually accomplished between two databases according to the literature (Alhajj and Polat, 2001). The first database is a relational database, called the *source*, and the second, called the *target*, which represents the result of the migration process. In addition, the process is carried out with or without the help of an intermediate conceptual representation, e.g., an ER model as a stage of enrichment. The input source schema is enriched semantically and translated into a target schema. Data stored in the source database are converted into target database based on the target schema. Generally, relations and attributes are translated into equivalent target objects. Foreign keys may be replaced by another domain or relationship attributes. Weak entity relations may be mapped into component classes, multi-valued or composite attributes inside their parent class/entity. Other relationships, such as associations and inheritance, can also be extracted by analyzing data dependencies or database instances. In data conversion (Abdelsalm Amaraga Maatuk, 2009), attributes that are not foreign keys become literal attribute values of objects, elements or sets of elements. Foreign keys realize relationships among tuples, which are converted into value-based or object references in a target database. The challenge in this process is that the data of one relation may be converted into a collection of literal/references rather than into one corresponding type. This is because of the heterogeneity of concepts and structures in the source and target data models.

3.2. Translation Techniques

Existing techniques can be classified into two types: (i) Source-to-Target (S2T), including flat, clustering and nesting translation techniques, and (ii) Source-to-Conceptual-to-Target (SCT) translation. In some of these techniques, data might be converted based on the resulting target schema.

3.2.1. Source-to-Target (S2T) Technique

This type of technique translates a physical schema source code directly into an equivalent target. However, as the target schema is generated using one-step mapping with no ICR for enrichment, this technique usually results in an ill-designed database as some of the data semantics are ignored. This approach could take the following forms:

- *Flat Technique*: This technique converts each relation into object class/XML element in the target database (Wang C et al.,, 2006). FKs are mapped into references to connect objects. However, the flattened form of RDBs is preserved in the generated database, with which object-based model features and the hierarchical form of XML model are not exploited. This means that the target database is semantically weaker and of a poorer quality than the source. Moreover, creating too many references lead to degraded performance during the data retrieval.
- *Clustering Technique*: This technique is performed recursively by grouping entities and relationships together starting from atomic to construct more complex entities until the desired level of abstraction is achieved (Sousa et al., 2002). A strong entity is wrapped with all of its direct weak entities, forming a complex cluster labeled with the strong entity name. This technique works well when the aim is to produce hierarchical forms with one root. This technique may reduce search time by avoiding join operations, and

thus speeding up query processing, however, it may lead to complex structures and is prone to errors in translation. In addition, materializing component entities within their parent/whole entities may cause data redundancy, the loss of semantics and the breaking of relationships among objects.

- *Nesting Technique*: This technique uses the iterated mechanism of a nest operator to generate a nested target structure from tuples of an input relation. The target is extracted from the best possible nesting outcome. However, the technique has some limitations, e.g., mapping each table separately and ignoring integrity constraints. Besides, the process is quite expensive, as it needs all tuples of a table to be scanned repeatedly to get the best possible nesting.

3.2.2. SCT Technique

This type of technique enriches a source schema by semantics that might not have been clearly expressed in it and their inter-relationships also (Alajj et al., 2003). Then, the schema is translated from logical into conceptual through recovering the domain semantics (e.g., primary keys, foreign keys, cardinalities, etc.) and making them explicit. Finally, the results are represented as a conceptual schema using DBRE, which can be translated into the target effectively. In this way, the technique results in a good well-designed target database.

4. Database Migration Tools (DBMTs)

A number of prototypes and tools have been developed to facilitate the migration of relational databases into target databases. DBMTs presented a system, called the Knowledge Extraction System (KES), for generating an EER model from RDBs. KES has been developed to extract domain semantics by analyzing the RDB schema and data instances. However, various semantic constraints, schema-mapping constructs and data migration techniques were not addressed adequately in this work. In later years, plenty of tools arrived for database migrations. In software industries, one of the major problems is ensuring quality database administration, the tasks connected to a migration workflow is diverse and complicated. Executing all these processes manually requires plenty of time and a highly experienced migration team in the source as well as the target system is essential. As both factors are not available in most situations, migration tools may come handy and should be considered to ease the migration workload. Table 2 presents some examples of database migration tools (Jutta Hortsmann, 2005).

Senior researchers Bin Wei, and Tennyson X. Chen (2012), developed Data Migration Tool (DMT) with five criteria for US National Oceanic and Atmospheric Administration (NOAA) that need to be considered when evaluating a DMT.

The five criteria's are

1. Types of database the DMT supports, If does not support the database from or to which users need to perform the migration.
2. How the database transfer is configured through the DMT's interface. This configuration will determine whether the data transfer can be executed repeatedly.
3. DMT should check database integrity before executing a data transfer. Sometimes database migration operation fails because of database integrity violation among the data.
4. How well the DMT incorporates customized data transfer requirement.
5. Ensuring the correctness of a completed database migration operation.

While the criteria outlined above are adequate for the complex project that developing for DMTs. The complexity of a general extract, transform, and load (ETL) system may go beyond what these criteria can evaluate. Most of the DMTs are not adequate enough to handle with complex data like image, audio, and video files. Still the investigations are needed on dealing with complex files. In addition, some other criteria, such as data migration performance and cost, can be important to project managers and DMT developers.

Table 2. Database Migration Toolkits

S.No	Name	Company	Source	From	To	Operating System
1	OSDM Toolkit	Apptility	Open	Oracle, SyBase, Informix, DB2, MS Access, MS SQL	PostgreSQL& MySQL	Windows, Linux, Unix & Mac OS
2	DB Migration	Akcess	Closed	Oracle & MS SQL	PostgreSQL& MySQL	Windows
3	Mssql2 Pgsql	OS Project	Open	MS SQL	PostgreSQL	Windows
4	MySQL Migration Toolkit	MySQL AB	Open	MS Access & Oracle	MySQL	Wndows
5	MySQL Migration Toolkit	Intelligent Convertors	Closed	MS Access, MS SQL, Dbase & Oracle	MySQL	Windows
6	Open DBcopy	Puzzle ITC	Open	Any RDB*	Any RDB*	OS Independent
7	Progression DB	Versora	Open	MS SQL	PostgreSQL, MySQL & Ingres	Linux & Windows
8	Shift2Ingres	OS Project	Open	Oracle & DB2	Ingres	OS Independent
9	SQLPorter	Real Soft Studio	Closed	Oracle, MS SQL, DB2 & Sybase	MySQL	Linux, Mac OS & Windows
10	SQLWays	Ispirer	Closed	All Relational Databases	PostgreSQL & MySQL	Windows
11	SwisSQL Data Migration Tool	AdventNet	Closed	Oracle, DB2, MS SQL, Sybase & MaxDB	MySQL	Windows
12	SwisSQL SQLOne Console	AdventNet	Closed	Oracle, MSSQL, DB2, Informix & Sybase	PostgreSQL & MySQL	Windows
13	MapForce	Altova	Closed	SQL Server, DB2, MS Access, MySQL & PostgreSQL	SQL Server, DB2, MS Access & Oracle	Windows, Linux & Mac OS
14	Centerprise Data Integrator	Astera	Closed	SQL Server, DB2, MS Access, MySQL & PostgreSQL	SQL Server, DB2, MS Access, MySQL & PostgreSQL	Windows
15	DBConvert	DB Convert	Closed	Oracle, DB2, SQLite, MySQL, PostgreSQL, MS Access & Foxpro	Oracle, DB2, SQLite, MySQL, PostgreSQL, MS Access & Foxpro	Windows

5. Conclusions and Future Work

In this paper, a deep study is presented on the existing database migration strategies since from 1987. In addition, investigation on primitive migration like legacy migration and database reverse engineering are also carried out. The major limitations of each strategy are pointed out. Each migration strategies have some rules to enable the process, which might be a point of disadvantage of migration process. It must be observed that the schema translation are done using S2T technique, and SCT techniques are employed during the process of migration to XML , also the major focus on generating a DTD schema and data. Due to focusing on schema rather than data, strategies either data loading or enabling on the structure of target databases and data remain stored in relational databases. Moreover, there are still shortcomings in implementations of schema translation and data conversion mechanisms. By using middleware technologies, it may lead to slow performance, making the process too expensive at run-time because of dynamic mapping between source and target databases. Most of the database migration techniques generate a database that to either flat relational or has a deep level of clustering or nesting. The existing technique does not provide a solution for more than one target database or for either schema or data conversion. Besides, none of the existing techniques can be considered as a method of time and space complexity of migration between source and target databases.

Some examples of migration toolkits were also presented. Database Migration Tool (DBMT) kit could prove to be very useful in minimizing the risk, stay on budget, keep downtime to minimum, and, in case of failure of migration schema and data be able to remain in source database. Unfortunately, there is not much choice with respect to migration tools for open source databases. This might lead to the migration team having to perform most of the steps “manually”. However, database migration is a common task that most database administrators need to tackle. Knowing how to evaluate and choose the right DMT can be vital to the fate of a software project that might directly contribute to the success of a business operation. Yet there are few guidelines in how to evaluate the usefulness and effectiveness of a general DBMT. It is concluded with five basic criteria that can serve as standards for current and future Database Migration Tools.

And this review paper expose that developers will serve users well in evaluating DBMT products and for future research.

REFERENCES

[1] Andreas Meier et al., (1994). 'Hierarchical to Relational Database Migration', *IEEE Software*, Vol. 11 Issue 3, pp 21-27

[2] Abdelsalam Amaraga Maatuk, (2009) Migrating Relational Databases into object-based and XML databases. Published PhD thesis, Nortambria University, Newcastle, United Kingdom.

[3] Andreas Meier. (1995) 'Providing Database Migration Tools – A Practitioner's Approach', in VLDB'95; *Proceedings of the 21st International Conference on Very Large Data Bases VLDB*, Zurich, Switzerland, pp 635-641.

[4] Andreas Behm, Andreas Geppert and Klaus R. Dittrich. (1997). 'On the Migration of Relational Schemas and Data to Object-Oriented Database Systems', *Proceedings of 5th International Conference on Re-Technologies for Information Systems*, Klagenfurt, Austria, pp 13-33.

[5] Barron C. Housel, Vincent Y. Lum, Nan Shu (1974), 'Architecture to an Interactive Migration System' in *SIGFIDET 1974: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, Access and Control,* New York, USA, pp. 157-169.

[6] Bin Wei and Tennyson X. Chen (2012), '*Criterial for Evaluating General Database Migration Tools'*, [online] Research Report, RTI Press Publication No. OP-0009-1210. Research Triangle Park, NC:RTI Press. http://www.rti.org/pubs/op-0009-1210-chen.pdf (Accessed 28 August 2015)

[7] Chiang, R.H.L., Barron, T.M. & Storey, V. C. (1994) 'Reverse Engineering of Relational Databases: Extraction of an EER model from a Relational Database', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, pp 107-42

[8] Christine Parent and Stefano Spaccapietra. (2000) 'Database Integration: The Key to Data Interoperability', in M. P. Papazoglou. et al (Eds.), *Advances in Object-Oriented Data Modeling*, The MIT Press, pp. 221-254

[9] Crowe, M. K. (1993). 'Object systems over relational databases', *Information and Software Technology* Vol. 35, pp 449-61

[10] Fishman et al., (1987) 'Iris: An object oriented Database Management System', *ACM Transactions on Office Information Systems* Vol. 5, pp 48-69

[11] Fong. J. and Cheung, S. K. (2005) 'Translating relational schema into XML schema definition with data semantic preservation and XSD graph', *Information and Software Technology*, Vol. 47 Issue 7, pp 437-462

[12] Hainaut, J.(1991) 'Database Reverse Engineering, models, techniques, and strategies' in ER'91; *Proceedings of the 10th International Conference on Entity-Relationship Approach*. San Mateo, California, USA, pp 729-41

[13] Hardwick, M. & Spooner, L. (1989). 'The ROSE data manager: using object technology to support interactive engineering applications'. *IEEE Transactions on Knowledge and Data Engineering*, Volume 1, Issue 2, pp 285-289

[14] Ian S. Graham and Aan Graham. (1995) *Migrating to Object Technology*, 1st ed., Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

[15] Jutta Horstmann. (2005) *Migration to Open Source Databases.* Diploma Thesis, Technical University Berlin, Computation and Information Structures (CIS), Berlin, Germany.

[16] Kathi Hogshead Davis and Peter H. Aiken (2000), 'Data Reverse Engineering: A Historical Survey', in WCRE'00; *Proceedings of the Seventh Working Conference on Reverse Engineering*, Brisbane, QId, pp 70-78.

[17] Maatuk, Abdelsalam, Ali, Akhtar and Rossiter, Nick (2008), 'An Integrated Approach to Relational Database Migration' in IC-ICT 2008; *Proceedings of International Conference on Information and Communication Technologies*, Bannu, Pakistan.

[18] Maatuk, Abdelsalam, Ali, Akhtar and Rossiter, Nick (2010), 'Converting relational databases into object relational databases' *Journal of Object Technology*, Vol. 9 Issue 2 pp 145-161.

[19] Mansaf Alam and Siri Krishnan Wasan. (2006) 'Migration from relational to object-oriented database', *Journal of Computer Science*, Vol. 2 No. 10, pp. 781-784.

[20] Michael L. Brodie, Michael Stonebraker (1994) *Migrating Legacy Systems: Gateways, Interfaces & The Incremental Approach*", San Francisco, California, USA, Morgan Kaufmann Publishers, Inc.

[21] Monk, S., Mariani, J., Elgalai, b., & comphell, H. (1996). 'Migration from relational to Object-Oriented Databases', *Information and Software Technology* Vol. 38, Issue 7, pp 467-75

[22] Moriarty, T. & Hellwege, S. (1998) 'Data Migration', *Database Programming & Design Magazine issue of 98*, pp. 11 – 14.

[23] Peter McBrien and Alexandra Poulovassilis, 'Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach, in CAiSE 2002; *Proceedings of 14thInternational Conference on Advanced Information Systems Engineering*, Toronto, Canada, pp 484-499

[24] SEI (2004), *Software Engineering Institute Glossary*. [online] http://www.sei.cmu.edu/str/indexes/glossary/(Accessed 5th December 2005).

[25] *The Complete Data Migration Methodology*. [online] http://www.dulcian.com/(Accessed 2 , 2004)

[26] Wang C, Lo A., Alhajj R., Barker, K., 'Novel approach for reengineering relational databases into XML, in ICDE'06; *Proceedings of the International Conference on Data Engineering,* pp. 1284-1289.

[27] Weiderman, Nelson H. Bergey, John K. Smith, Dennis B. Tilley, Scott R (1997) *Approaches to Legacy System Evolution*,[online] Technical Report CMU/SEI-97-TR-014 ESC-TR-97-014, Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA 15213.

http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA336213.pdf (Accessed 23 July 2014)

[28] Wilknison, K., Lyngboek, P. & Hasan, W. (1990). 'The IRIS architecture and implementation', *IEEE Transactions on Knowledge and Data Engineering,* Vol. 2, pp 63-75

[29] Wire Ming Lim and John Harrison (1996), 'An Integrated Database Reengineering Architecture-A Generic Approach.', *proceedings of Australian Software Engineering Conference*, 1996, Australia, pp 146-154

[30] Yury Bychkov and Jens H.Jahnke, (2001) 'Interactive Migration of Legacy Databases to Net-Centric Technologies', in WCRE'01; *Proceedings of Eighth Working Conference on Reverse Engineering*, Stuttgart, Germany, pp 328-334