MARCH, 2019

M.Sc. in Electronics and Computer Engineering

HAWKAR HAMASALIH

HASAN KALYONCU UNIVERSITY

GRADUATE SCHOOL OF

NATURAL & APPLIED SCIENCES

DATABASE MIGRATION PROCESSES AND OPTIMIZATION USING BSMS (BANK STAFF MANAGEMENT SYSTEM)

M. Sc. THESIS

IN

ELECTRONICS AND COMPUTER ENGINEERING

BY

HAWKAR HAMASALIH

MARCH 2019

Database Migration Processes and Optimization Using BSMS (Bank Staff Management System)

M.Sc. Thesis

in

Electronics and Computer Engineering

Hasan Kalyoncu University

Supervisor

Asst. Prof. Dr. Mohammed MADI

by

Hawkar HAMASALIH

March, 2019

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES INSTITUTE
M.Sc. ACCEPTANCE AND APPROVAL FORM

Electronics-Computer Engineering M.Sc. (Master of Science) programme student **Hawkar HAMASALIH** prepared and submitted the thesis titled **"Database Migration Processes and Optimization Using BSMS (Bank Staff Management System)"** defended successfully on the date of 22/03/2019 and accepted by the jury as a M.Sc. thesis.

Position	**Title, Name and Surename** **Department/University**	**Signature:**
M.Sc. Supervisor **Jury Head**	Assoc. Prof. Dr. Mohammed. MADI Computer Engineering Department Hasan Kalyoncu University	
Jury Member	Assoc. Prof. Dr. Ahmet Mete VURAL Electrical and Electronics Engineering Department Gaziantep University	
Jury Member	Assist. Prof. Dr. Saed ALQARALEH Computer Engineering Department Hasan Kalyoncu University	

This thesis is accepted by the jury members selected by the institute management board and approved by the institute management board.

Prof. Dr.Mehmet KARPUZCU

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Hawkar HAMASALIH

ABSTRACT

DATABASE MIGRATION PROCESSES AND OPTIMIZATION USING BSMS (BANK STAFF MANAGEMENT SYSTEM)

HAMASALIH, Hawkar Kakaawla Hamasalih

M.Sc. in Electronic and Computer Engineering

Thesis Supervisor: Assist. Prof. Dr. Mohammed MADI

March, 2019, 59 pages

Databases are essentially a storage technology designed to handle complex data dependent tasks, and to perform these tasks, data-integrity is important. For many companies, their database is literally an electronic representation of the company's business and records and losing any bit of data during migration is unacceptable. There are several business reasons for moving data, some of these are archiving, data-warehousing, moving to new environment, platform, or technology. Database migration is a complex, multiphase process, which usually includes assessment, database schema conversion, data migration, and functional testing. Online Transaction Processing (OLTP) databases are usually much normalised for efficiency by performing tasks like providing data Integrity, eliminating data redundancy and lowering record locking. But this database design system presents us very numerous tables, and each of these tables and its foreign-key constraints must be accounted for at the point of data migration. Also, the Acceptance criterion for a data-movement job unlike conventional tasks is purely 100% because errors are not tolerated in databases and quality is important. This thesis demonstrates the challenges and considerations during the migration of data from a slow, inefficient and obsolete database-platform called Paradox database into a much more advanced database called Oracle that has successfully migrated the data. Indexing technique was used to improve the performance of a query by retrieving the data at a rapid speed without any inconsistency and loss of data.

Keywords: Data Migration, DBMS, Data Integration, Database Migration Tools.

ÖZET

ALMA SİSTEMLERİNE İLİŞKİN BİR KARŞILAŞTIRMA ÇALIŞMASI

HAMASALIH, Hawkar Kakaawla Hamasalih
Yüksek Lisans Tezi, Elektronik ve Bilgisayar Mühendisliği
Tez Yöneticisi: Yrd.Doç. Dr. Mohammed MADI
Mart, 2019, 59 sayfa

Veritabanları temel olarak karmaşık verilere bağlı görevleri yerine getirmek ve bu görevleri gerçekleştirmek için tasarlanmış bir depolama teknolojisidir, veri bütünlüğü önemlidir. Pek çok şirket için, veritabanları kelimenin tam anlamıyla şirketin işinin elektronik bir temsilidir ve göç sırasında herhangi bir veri parçasını kaybeder ve kaybeder kabul edilemez. Verilerin taşınmasının çeşitli ticari nedenleri vardır, bunlardan bazıları arşivleme, veri depolama, yeni ortama, platformlara veya teknolojiye geçmedir. Veri tabanı geçişi, genellikle değerlendirme, veri tabanı şeması dönüşümü, veri geçişi ve işlevsel testi içeren karmaşık, çok fazlı bir işlemdir. Çevrimiçi İşlem İşleme (OLTP) veritabanları genellikle veri bütünlüğü sağlama, veri fazlalığını ortadan kaldırma ve kayıt kilitlemesini azaltma gibi görevleri yerine getirerek verimlilik için çok normalize edilir. Ancak bu veritabanı tasarım sistemi bize çok sayıda tablo sunar ve bu tabloların ve yabancı anahtar kısıtlamalarının her biri veri taşıma noktasında dikkate alınmalıdır. Ayrıca, geleneksel görevlerden farklı olarak veri taşıma işi için Kabul kriteri tamamen% 100'dür, çünkü hatalar veritabanlarında tolere edilmez ve kalite önemlidir. Bu tez, verilerin Paradox veritabanı adı verilen yavaş, verimsiz ve eski bir veritabanı platformundan, verileri başarıyla geçiren Oracle adı verilen çok daha gelişmiş bir veritabanına aktarılması sırasında ortaya çıkan zorlukları ve kaygıları göstermektedir. Herhangi bir tutarsızlık ve veri kaybı olmadan verileri hızlı bir şekilde alarak, bir sorgunun performansını iyileştirmek için indeksleme tekniği kullanılmıştır.

Anahtar Kelimeler: Veri Taşıma, DBMS, Veri Entegrasyonu, Veritabanı Taşıma Araçları.

To My Parents

ACKNOWLEDGEMENT

My deepest gratitude goes to Asst. Prof. Dr. Mohammed Madi for his constant encouragement and guidance. He has walked me through all the stages of the writing of my thesis. Without his consistent and illuminating instructions, this thesis could not have reached its present form. I would like to thank all the lecturers in electronics and computer engineering department in Hasan Kalyoncu University.

I would also like to acknowledge Mr. Sirwan the second reader of this thesis, and I am gratefully indebted to his valuable comments on this thesis. Also, I appreciate my both dear brothers Mr. Hoshang and Mr. Hersh who were too tired and supported me in every step.

Finally, I would like to thank my family and friends to support, encouragement and contribution they have made for my research to be successful.

TABLE OF CONTENTS

LIST OF FIURES

LIST OF ABBRIVIATIONS

BSMS	Bank Staff Management System
CSV	Comma Separated Value
DB	Database
DBMS	Database Management System
DDL	Data Definition Language
ETL	Extract Transform and Load
NHS	National Health Service
ODBC	Open Database Connectivity
PAL	Paradox Application Language
RDBMS	Relational Database Management System
SQL	Structured Query Language
SSIS	Server Integration Service

CHAPTER 1

INTRODUCTION

1.1 Introduction

Moving data which is sometimes called ETL (Extract Transform and Load) is a core area of database administration and the backbone of data-warehouses [1]. In this thesis, we will try to work with an existing commercial database system originally designed by Key-IT systems for its hospital clients. The software tool called BSMS (Bank Staff Management System) was developed by Key IT System Ltd and delivered to NHS in order to manage the availability of temporary nursing staff in cases of emergency [2]. The original database designed for this commercial software is called Paradox, the task is to transfer the hospital staff data which is stored in the form of Paradox tables from existing BSMS database to a new instance of Oracle database without loss of data. The details of the Database are attached to this project as an Appendix.

One of the Primary reasons for data movement in Databases is Business Intelligence and this is because most Organizations have both a development database and a production database that constantly need to exchange data. Like several Database vendors, Oracle has many utilities that facilitate the movement of data between environments such as data-pump and SQL-Loader. The main function of an ETL system such as SQL-Loader is to help insert a CSV (comma separated value) file into an RDBMS and catch bugs during data insertion.

The three steps in ETL process are designed to complement each other, in the extract step, we are pulling the data from the file, then during the Transform Step, the system is filtering rows to possibly discard unwanted rows, the transform stage can also attach or bind data-types, bind variables and even rename columns. The “L” in ETL stands for load; this is the final step where we insert the rows into the destination Oracle table

1.2 Problem Statement

Due to the heavy operational, performance and maintenance costs involved in maintaining the Paradox databases and their inflexibility to support new business creativities, the client wanted to migrate the data from these various mainframes to a common consolidated repository under Oracle Database [2]. The existing client applications then will start communicating from Oracle instead of Paradox. This thesis seeks to employ empirical epistemology in the evaluation of the claims made to the public by Oracle Corporation that its own SQL-Loader is the best tool to be used for data migration. It has decided to employ the dataset of an English-Hospital that houses its dataset in a commercial software system. The software uses Paradox as its DBMS, but in today's time using this DBMS causes such problems of data inconsistency and inaccuracy. Furthermore, the huge cost of maintaining these legacy-systems keeps on growing [3]. Given the immense challenges of migrating over to a newer technology, the custodians of this software want to migrate to a new database-platform which is efficient, consistent and reliable.

Given the challenges inherent in the dataset, the objective of this thesis is to conduct a detailed data-migration experiment using the various publicly available data-migration tools, then advice ETL professionals on the benefits and sacrifices to be made in choosing one tool over the other.

1.3. Objectives of the Project

Over the years, various ETL vendors have incorporated certain common-place processes into the system of data-migration. But in this thesis, focus on sequential migration, this technique is the most commonly used technique for data migration in the retail business [4].

The objectives of this thesis can be broken into three parts.

1. Move an entire database system from a desktop RDBMS (Paradox) to a Client/Server RDBMS (Oracle).
2. Research and perform testing on the various methods of moving the database.
3. Make recommendations on the best way of carrying out this task by evaluating various data-quality criteria.

After the project was completed, the following are the project questions that were answered:

1. What is the requirement for organizations to migrate their databases?
2. What are end profits of data migration for the owners?
3. What can be done to make all data is transferred during data migration?
4. What was the performance improvement of an application after the data migration?
5. What tools and solutions are available to do the migration process?

1.4. Significant of the Thesis

[5] Mentioned that most IT projects today worry some kind of corrective measure at the platform level, they all involve moving data from database to database or application to application due to migrating data and applications from old to modern platforms. In this thesis different tools and platforms has been tasted to do the migration after the data has been moved from the source to the destination some technics implemented to improve the performance of the data.

1.5. Organization of the Thesis

This thesis is divided into five Chapters. The First Chapter is the introductory Chapter which presents the problem of stamens and objectives of the project. The second chapter is about the background information on the Data Migration process and it shows the essential steps to success the migration and explains the different methodology's which exist. Also the explanation of the tools and technology which used in this project has been written.

Chapter three has discussed the research methodology which starts with the logical attitudes that guided the research. It describes the research design, scenarios of the study and different experiments have been tested.

Chapter four presents result testing and performance tuning, after the migration process takes place two tools are used to see the consistency and correctness of the data and we made use of Indexing technique in order to improve the performance of a query. Chapter Five focuses on the summary of the findings, conclusion and recommendations of the research. It presents the implications of the study in accordance to theory, policy, management practice, methodology and limitations of the study.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

Planning of migration methodology and implementing smooth migration for the whole project is the main task of the project. Nowadays technology is emerging rapidly and new database applications available in a more customized and organized manner. Because of the quickly improving technology for computers and changes in business requires the normal life cycle for a database application is less than 10 years [6]. After this time the ability of applications to manage data will not match the needs of the organizations or the technology on which it operates will have become out-of-date. Therefore the organizations will be required to implement a new database application which leads to data migration from legacy systems to modern applications.

The direct impact of a poor data migration procedure is usually very poor data-quality. A Survey conducted by [7], indicates that many managers and business owners are oblivious to the fact that the data in which their business decisions are based on are flawed and inaccurate. Poor data quality could be catastrophic and possibly lead to expensive lawsuits and according to the [7]. Redman; there has been an increase in operational cost in organisations with poor quality data because a lot of time and other resources are devoted to detecting and correcting errors. He states that in the business world, “Decisions are no better than the Information in which they are based”. Database correctness has to do with the integrity and consistency of stored data, and Integrity is usually expressed in terms of constraints, and these constraints are consistency rules that the database is not permitted to violate. [8] Shows that error rate of up to five percent are not unusual, with a projected immediate loss of about ten percent in revenue. Every facet of business is negatively affected by poor data

quality. And the trend of poor data quality is worsened in huge organisational databases with several sources of data inputs.

The importance of quality to database design is very important and [9] states that the wrong Information impedes deductive and inductive thoughts. Over the years, experience has shown that even the best decisions involve some degree of uncertainty; so, it is obvious that decisions based on the most relevant, complete, accurate and timely data have a better chance of advancing the enterprise's goal. According to the journal by [7], Shipping and mailing Industries are some of the hardest hit businesses because of poor data quality, and this often lends itself to poor job satisfaction among the employees of these businesses. In a survey conducted in the United States, [7] concluded that a data inaccuracy of about five percent can increase the operational cost by twelve percent, increase organisational mistrust, cause gridlocks in decision making and make implementation of a data-warehouse very difficult.

2.2. Background of the Problem

The process of transferring data from one system to another is called Data migration and it varies from data movement. [5] Mentioned that due to the complexity and similarity to related techniques, data migration suffers a few myths and misconceptions, for example, Data migration is not a matter of copying data because the term migration misleads some people to think they can simply copy data from system A to system B. But the systems usually have different data models, so mapping and copying data without transforming it is rare. Moving data from one system, or domain, to another without compromising security or losing any of the data is known as domain migration, it happens when servers are upgraded and the data (including any authentication and authorization information) must be moved to a new system.

2.3. Literature Related to the Problem

2.3.1. Migration Strategies

According to [10], the Chicken Little Methodology is One of the earliest migration approaches It is a gateway-based eleven-step approach which allows both the legacy and target system operate in parallel during the migration operations. The approach is

also incremental as the target system though small at the onset, continues to grow as the migration progresses until it replaces the legacy system.

Another approach challenged the iterative and approach of the Chicken Little which is The Butterfly Methodology. This required the definition of a five-phase approach which involves the following steps:

Step 1: Determination of the semantics of the candidate legacy system and development of the target schema;
Step 2: Construction of a sample data store in the target system based upon target sample data;
Step 3: Migration of all the information system components while leaving out the data;
Step 4: Migration of the legacy data to the target system and the training of users
Step 5: Decommissioning of the legacy system and switch over to the new system
Each of these steps is further broken down into sub-steps and specific activities [4].
[11] Mentioned that generally there are two principal types of migration: big bang migrations and trickle migrations. Depending on the project requirements and available processing window, any Organization wants to migrate their data should consider which style of migration is most suitable for their needs. Big bang migrations involve completing the entire migration in a small, defined processing window. In the case of a systems migration, the system interruption while the data is extracted from the source system(s), processed, and loaded to the target, followed by the transferring of processing completed to the new environment.

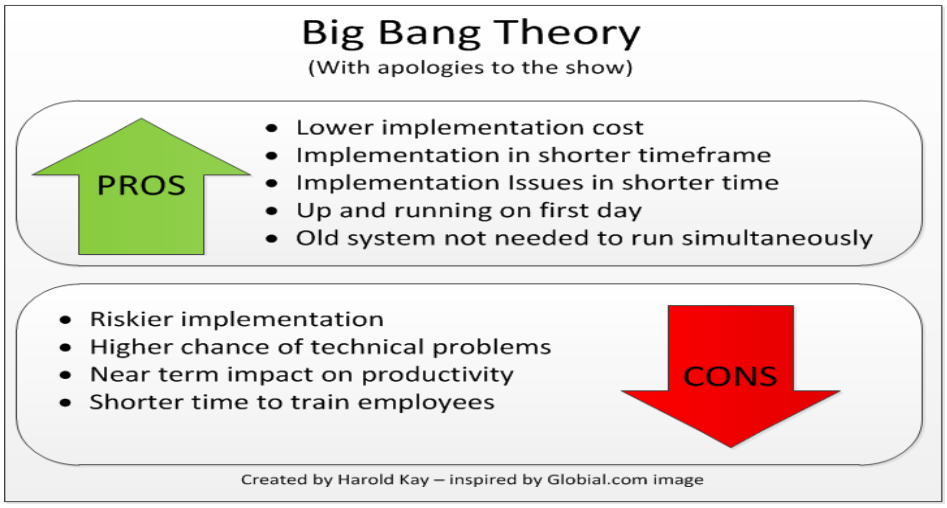


Figure 2. 1 [12] : Big Bang Migration Principle

This method is more attractive it can complete the migration in the shortest possible time with carrying several risks, it is hard for organizations to wait to do the migration and see the core system's being unavailable for long. Trickle migrations take an additional approach to migrating data instead of completing the whole process in a short period window. In this approach, the old and new system is running in parallel and the data will be migrated in parts. The system with this method never stops working and it will still be working 24/7.

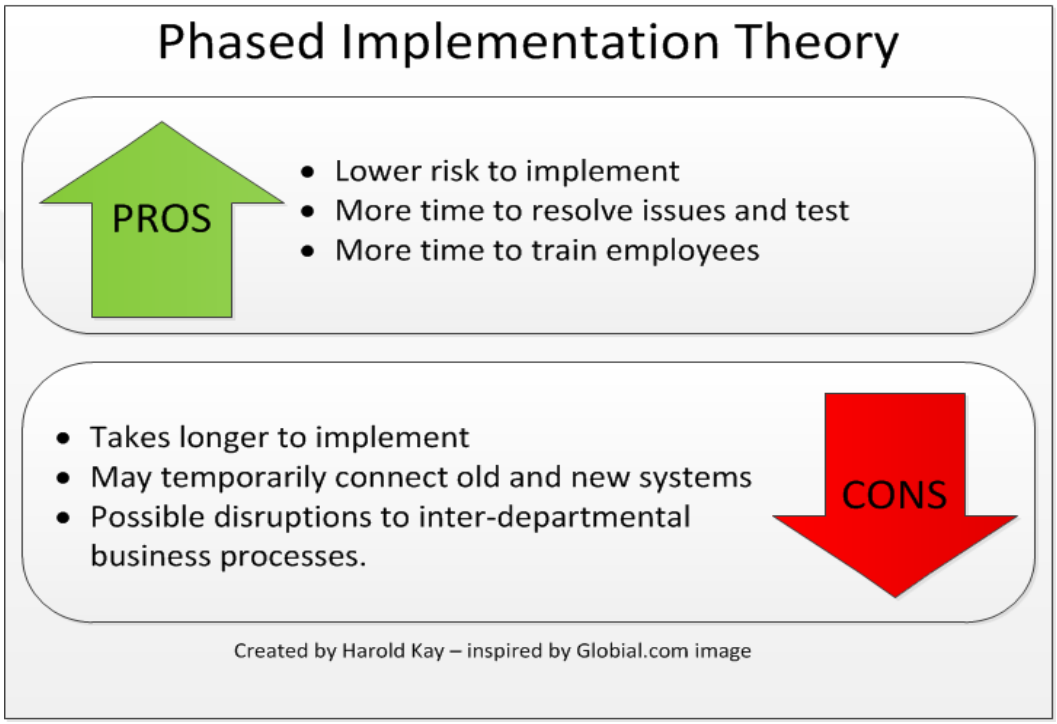


Figure 2. 2 [12]: Phased Implementation

To move data with real-time processes a trickle migration can be implemented, to maintain the data these processes can also be used by passing upcoming changes to the target system.

2.3.2. Essential Steps to Success

[13] has written some essential steps to success the migration process, in the article six different phases have been mentioned which they are Planning, Understanding the Data, Designing, and Building, Executing, Testing, Follow-Up and Maintenance. After the business has given the data migration project a broad migration scope, in the planning phase the first stage is to define what is viable in terms of what the data sources will support and what is reasonable.

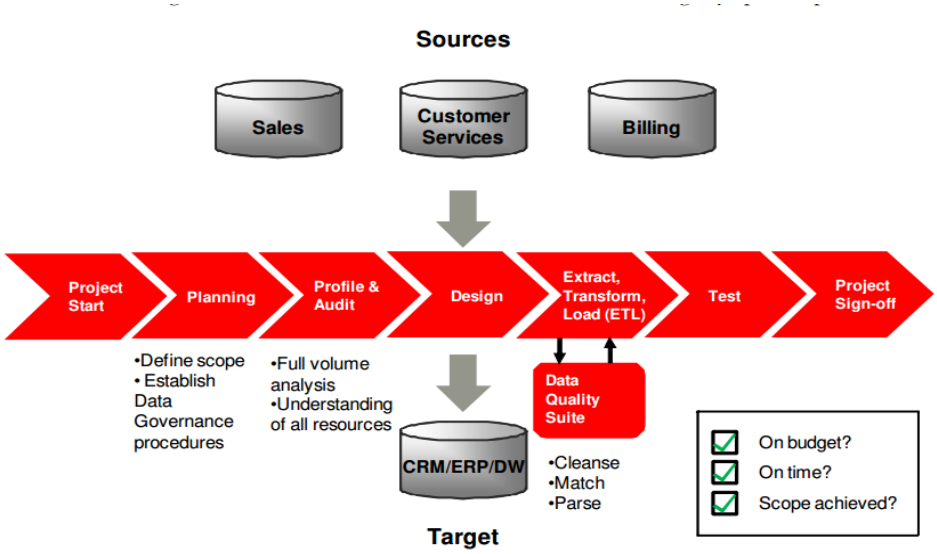


Figure 2. 3 [13]: An effective data migration project

Data migration projects are not an easy process it is too complex, the scope should be defined clearly and a timeline, resource plan, and budget can be put in place. Arrange scoping decisions in line with the importance to the organization by applying a top-down, target-driven rationale and using criteria such as region, the line of business, and product type. Times and effort could be saved by refining data outward from the core. In the third phase which is Designing and Building phase, the planning specifications and supporting documentation should be clearly agreed and signed off by the business, before writing any code. After segmenting into increments Data migration projects run more professionally, in staggered phases the source data could be audited, mapped, tested, and transferred. The budget and deadline are reduced and it will deliver better results. After converting the mapping specifications into migration code the rules could be verified independently, it helps to identify errors in the test environments and making key decisions for going live with migrated data. In the executing phase data from the source is extracted and then the process of transformation and cleansing on data are done, after that, it loaded into the target system. Before the conversion can be signed off by the business a collection application tests essential to be done, this approach benefits avoid storing up issues until too late in the development cycle. The data in the source systems is changing while the migration is in development, this could be a major risk.at the end phase of data migration data, the review can be done at any time to check whether the project is on track and still within its range.

2.4. ETL

These are processes that perform extract, transform and load operations on data which is called ETL in data-warehousing [14]. Recall that a Data warehouse is a repository of integrated enterprise data collected from different source systems within and possibly outside the enterprise. It consists of three main steps as indicated by its name. To extract means to pull data from a Source, to transform refers to manipulating the data as it moves from the source to the destination. In the transform step, changes are applied that make the data fit the schema of the destination database [15]. The transformation can be as simple as copying columns from one table to another or it can be more involved like removing duplicates, substituting values from look-up tables or changing the casing of strings as well as other forms of data cleansing. Loading means inserting the data into a destination store [16]. While the purpose and tasks of the ETL may sound to be well-understood and easy, the opposite is, in fact, the case. In virtually any data migration project, most of the time is spent on getting the ETL right. A lot of time is needed to understand when, where, and how to extract data from the different sources and to understand what the data from different sources mean and how the data can be successfully integrated.

2.5. SSIS

SSIS which stands for SQL-Server Integration Service is one of the members of the Microsoft Business Intelligence package. It helps us move data from one place to another changing its shape along the way if desired.

"Microsoft Integration Services is a platform for building enterprise-level data integration and data transformations solutions. You use Integration Services to solve complex business problems by copying or downloading files, sending e-mail messages in response to events, updating data warehouses, cleaning and mining data, and managing SQL Server objects and data. The packages can work alone or in concert with other packages to address complex business needs. Integration Services can extract and transform data from a wide variety of sources such as XML data files, flat files, and relational data sources, and then load the data into one or more destinations" [17].

According to [14], SSIS is an enterprise-grade tool for the extraction, transformation, and loading of data. It is designed to handle the migration and manipulation of data throughout the life-cycle of a Business Intelligence operation. It also has usefulness

outside the Microsoft environment because it could move data between other software applications; it supports almost every popular data-storage-source and destinations including Oracle, Excel etc. SSIS is Microsoft's tool for ETL, ETL stands for extraction, transformation, and loading. SSIS solutions typically consist of one or more packages. A typical SSIS package consists of several components, the control flow, and some number of data-flows, connection managers and configuration elements [17]. SSIS provides various drivers and configurations for many types of data sources. So almost irrespective of where the data is stored, we could use SSIS to pull it out, so we could work with data sources, data source views and connection managers to connect to those sources and pull data in various ways and we also have a lot of flexibility in the way we could configure that process. A typical SSIS-package can simultaneously extract data from various soft-wares like Oracle, excel etc.

SSIS supports a wide variety of destinations; this means we can output our data into spreadsheets, databases, flat-files etc. Although we can use SSIS to load data from external sources into a relational store, a data warehouse or an Analysis cube, we are not restricted to that intention; we can also use SSIS to move data to any type of database software [14].

2.6. BSMS Utilities

These were the tools for understanding the design of the Paradox database, and understanding the database to help make a smooth database migration project. The tools include the following

- P20: This tool has great functionalities with a powerful graphical user interface. This tool can also be used to generate a '.CSV' file.
- SQL Launch: This tool is also used to analyze the Paradox database tables. With the help of this tool, it is possible to write SQL queries and the knowledge obtained from the queries, it helps to reverse engineer the database in the Oracle interface

2.6.1. DATA Migration Tools and Solution Overview

These were the tools employed in the migration of data from Paradox into Oracle. The processes that perform extract, transform and load operations on data is called ETL in Data-Migration. The transformation can be as simple as copying columns

from one table to another or it can be more involved like removing duplicates, substituting values from look-up tables or changing the casing of strings as well as other forms of data cleansing. Loading means inserting the data into a destination store [16]. The following tools played a pivotal role in our migration process.

- **MS Access:** This tool was used as a middleware to transform the data from Paradox table to Oracle database.
- **SQL-Server Integrated Services:** This is one of the members of the Microsoft Business Intelligence package. It helps us move data from one place to another changing its shape along the way if desired (Guy and Langit 2011).
- **SQL-Loader:** This is Oracle's main tool for migrating data from flat-file sources [18].
- **Full Convert:** Full Convert will quickly and easily copy the data from source to the target database. It will create all the tables, copy all of the data, and then create indexes and foreign keys (Spectral Core 2018).

2.7. Data

[19] Mentioned that data in computing is information that can be changed into a form that has the ability to move and process, all information is converted into a binary digital system. Data represents as binary values on the computer; only two numbers 1 and 0 are used as a pattern to store video, sound, text, and images. The smallest unit of measuring data is a bit that is only a single value, the bigger unit which is eight binary digits called byte, usually, computer storages and memory are measured in megabytes and gigabytes. In mainframe systems, data can be stored in file formats but in better specialization developed a database, database management system and the relational database technology arose to organize information.

2.8. Databases and Database Management System (DBMS)

A database is a collection of information organized to provide efficient retrieval. The collected information could be in any number of formats (electronic, printed, graphics, audio, statistical, combinations). The relational database management system is a set of technologies designed to store soft copies of information in a structured and specific format. With many companies or organizations, the database holds the structure and data of the entire business. Users of the system are given facilities to perform several kinds of operations on such a system for either

manipulation of the data in the database or the management of the database structure itself [20]. Some examples of well-known database software are

- IBM DB2
- Microsoft Access
- Microsoft Excel
- Microsoft SQL Server
- MySQL
- Oracle RDBMS
- Paradox Database
- QuickBase

[21] Explains the main objectives which are used in DBMS as the following:

- Data availability: cost performance and the data updating are the responsibility of the data availability. Availability functions make the database available to users helps in defining and creating a database and getting the data in and out of a database.
- Data integrity-The data integrity provides protection for the existence of the database and maintaining the quality of the database.
- Data independence- DBMS provides two types of data independence. First is a physical data independence program, which remains unaffected from the changes in the storage structure or access method, and the second is the logical data independence program, which remains unaffected from the changes in the schema.

2.8.1. Database Management System components

The main five components of DBMS are hardware, software, data, procedures and database access language. The first component is hardware which includes the computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory. And the second component is software, it is the program for controlling everything and it will be responsible for easy-to-use interface to store, access and update data. The third component is data, DBMS was mainly created to store and manage it. Another component is procedures, it is including all instructions to use a database management system such as setup and install a DBMS, login and log out of DBMS software, manage databases, take backups, generating reports

etc... The last component of DBMS is database access language, it is a language designed to write commands to access, insert, and update and delete data stored in any database [22], [20].

2.9. Paradox Database

The paradox was an early desktop relational database management system (RDBMS) that was first released by Ansa Software in 1985. It was originally written in C, but later ported to C++, and was initially offered for Microsoft's DOS operating system. Ansa Software was purchased by Borland in 1987, and today Paradox is one of the smaller RDBMS offerings, although one with a dedicated group of users and supporters.

2.9.1. Characteristics of Paradox

[21] Said that the Characteristics of Paradox are:

- Connectivity- Paradox is connected virtually transparently with database and enables different ODBC databases, you can change Paradox tables in any relational database management system supported database, such as Oracle, SQL, and Microsoft.
- Data Converting- Paradox fully supports data converting, but conduction is another database which supports to RDBMS. Paradox offers great compatibility with other database applications. You can open a Paradox table in any ODBC-compliant application.
- Database Size- Paradox provides great flexibility; with the size of its database you can create up to two billion records per table and up to 255 fields per record.
- Query Expert- SQL is a Structured Query Language, which stores and manipulates information in relational databases. Paradox creates four different methods (Select, Insert, Update and Delete). The Query Expert is easiest for new database users. It takes you through a step-by-step process to create commonly used query models.
- PAL Object-PAL (Paradox Application Language Object) is an object-based event-driven and visual programming language, which can be used to completely customize applications with entirely new buttons, menus, dialog boxes, prompts, warnings, and online help. Object PAL can be

used to extend the regular Paradox functions or to create non-database applications.

2.10. Oracle database server

Oracle database server is a relational DBMS, the data collection is treated as a unit, it used to store and retrieve related data, it uses a server to manage its information, it can manage a huge amount of data in a connected environment which a number of users can use the same data concurrently with high-performance delivery, the server avoids unauthorized access and provides well-organized solutions for failure recovery. The first database was designed for enterprise grid computing is Oracle Database. The database has logical structures and physical structures. The physical storage of data can be managed without affecting the access to logical storage structures due to the separation of the physical and logical [23].

CHAPTER 3

METHOD OF EVALUATION

3.1. Data Migration

In this chapter, the intended migration path and the rationale behind the choices made have been explained.

The intention of this thesis is to take a qualitative assessment of the three different data migration techniques and ascertain which is better in a production environment.

3.1.1. Scenario 1:

The first migration is more of a manual process with a lot of human interactions and interventions in its actualization. It will involve a procedure of accessing the paradox database-tables, and after pulling the data into an Excel environment, Excel-functions are leveraged to clean the data until it conforms to desired destination constraints.

3.1.2. Scenario 2:

The second migration is also a manual process, but more sophisticated in approach than the first technique. In this technique, we leveraged an industry standard data-connector called ODBC data-connection systems to connect Access database to Oracle database.

3.1.3. Scenario 3:

The third technique is more of a data-migration automated process. In this approach, we leverage SSIS data-connectors both to pull and drop data to desired data-locations.

3.2. Set-up of Experiment

For this experiment, an Oracle Enterprise Class database has been installed in a test Computer. This software obtained by registering with the Oracle website and getting download permission from Oracle. Then a user account named exp1 (of elevated

privilege) created and granted the inbuilt HR account SYSDBA privilege for testing purposes. Also, SQL-Developer from the Oracle Website has been downloaded, SQL-developer is a free utility tool for administering the Oracle 11g Instance.

Microsoft office 2007 already installed and running on the test PC, and finally Microsoft SQL-Server 2008 RDBMS and the BSMS software installed as well.

All tables are migrated in the Paradox database to Oracle successfully but we would be focusing on the three key tables in the Paradox database to avoid the repeating process in this thesis writing, these are:

- Person Table
- Availability Table
- Requirement Table

3.3. Scenarios and Experiment

The primary objective of this phase of the project is to move the entire data and RDBMS configuration from PARADOX files into ORACLE without compromising data quality and avoiding data loss during the migration. And we would be exploring three different ways of doing data movement.

These are the four methods that decided to try-out and observe which of them produces an effective solution for future use.

1. Migrating from paradox to Excel, then proper cleaning and saving as CSV file format, and finally move data to Oracle with the help of SQL-LOADER.
2. Migrating from Paradox to Access and finally to Oracle via ODBC connection between Access and Oracle, then using SQL Statements in Oracle for data cleansing and assisting error-fixes.
3. Opening the various database tables with Excel, and then using Microsoft SSIS to import data, make data-transformation and finally push the data into Oracle 11g.
4. Using a Graphical User Interface tool called Full Converter to move the database tables from the source to the destination.

3.3.1. Preparing for Primary-Key Maintenance after Data-Movement

We need to enforce the row-level uniqueness of the primary-key after the data movement into Oracle. Unlike most RDBMS, Oracle does not automatically generate

primary key integers, but Oracle provides us with the ability to generate our new primary-key Integers with the sequence command. Sequences are schema objects, a single sequence would never repeat an already dispensed value, and this makes it perfect for primary-keys.

The sequence created below would be applied to the database table after it has been moved.

```
CREATE SEQUENCE
person_seq
      START WITH 2205
INCREMENT BY 1
NOMAXVALUE
NOCACHE;
```

```
CREATE SEQUENCE
avail_seq
START WITH 21330
INCREMENT BY 1
NOMAXVALUE
NOCACHE;
```

```
CREATE SEQUENCE
req_seq
START WITH 15233
INCREMENT BY 1
NOMAXVALUE
NOCACHE;
```

Figure 3.1: Database Table Moving Process.

3.3.2. Experement 1 – Moving Data with SQL-Loader

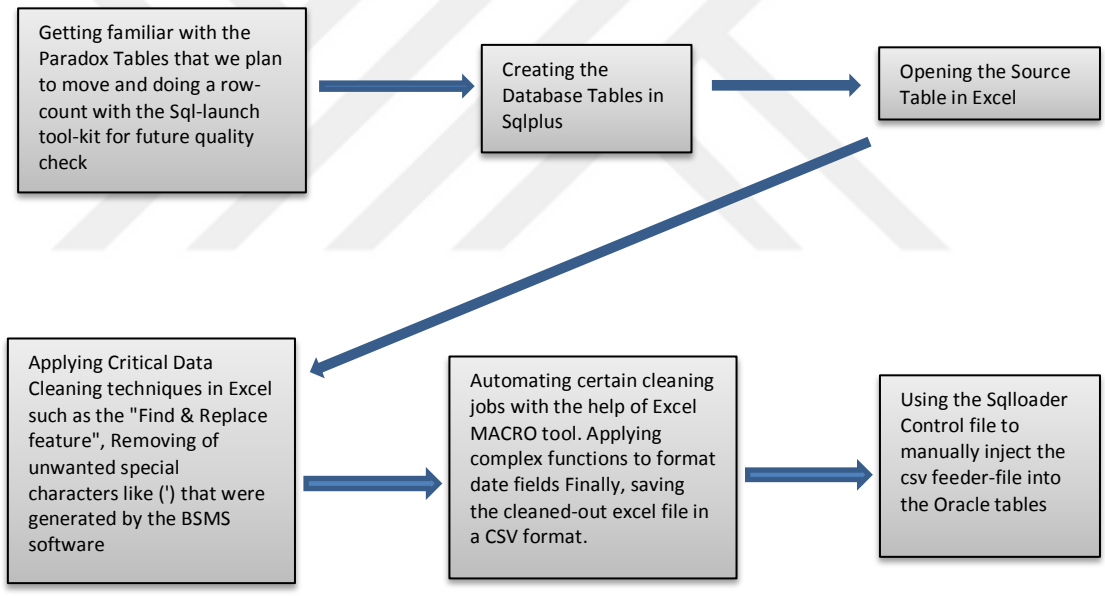


Figure 3.2: Moving Data with SQL-Loader

SQL-LOADER can be run from the Oracle Enterprise Manager or from the command line. The SQL-LOADER is our entry level bulk load utility. What this means is that we are taking data from an external source which is usually called the Feeder file. We are taking the Feeder-file and bringing its data content into an Oracle table. SQL-LOADER has two modes of operation which are known as paths, these are the conventional path and the direct path. The conventional path is basically a faster way of doing multiple insertion statements into a table. SQL-LOADER will

construct an insert statement and use a bind variable to read the source data and insert each row with a separate insert statement.

The Conventional path is the more stable way of achieving data insertion, but its major disadvantage is that it is much slower than the other paths. The direct path is like an injection into the source data files. When using the direct path, SQL-LOADER by-passes the database buffer cache. SQL-LOADER requires three files to succeed in its data movement, these are

a) INPUT DATA FILE: This is a standard delimited text file that contains our stored data. This is the feeder-file and is always usually a CSV file.
b) CONTROL FILE: This contains the loading instructions and sequence for the Oracle-engine. The control file is what we would likely be spending most of our time on. Because getting this right is critical because we can use the control files for subsequent SQL-LOADER jobs with similar data-set
c) LOG FILES: These are produced because of issuing the SQL-LOADER statement or running a bulk import Job. Log files might include a bad file which is a file where source data from the feeder-file that have violated some predefined constraints and the target table will be stored somewhere for debugging reasons. Log files also contain reject files which contain rows from the feeder-file that are out of scope.

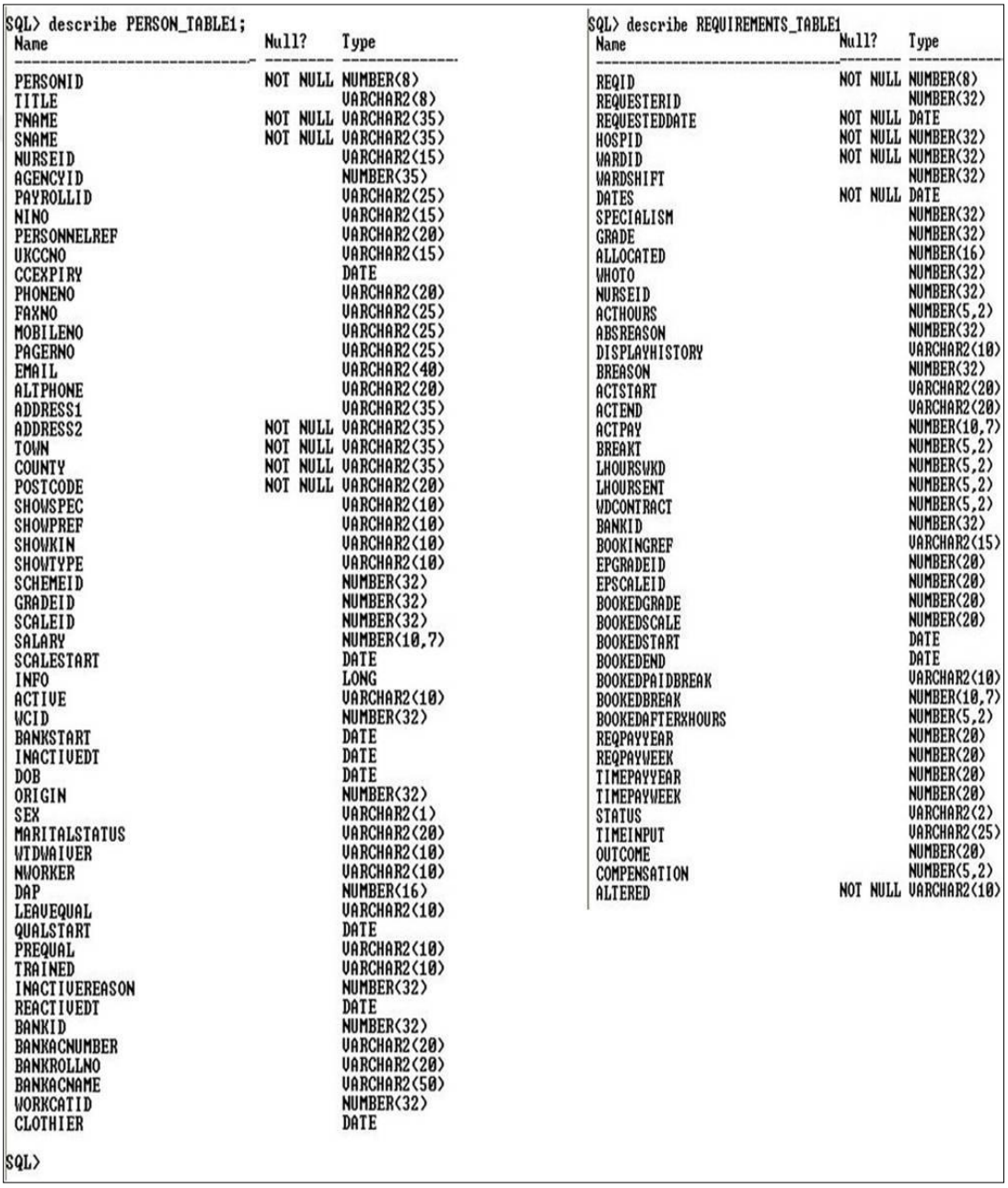
There are some features or advantages of using SQL-LOADER which listed below:

1. SQL-LOADER is a data loading utility that loads records from an external text file into Oracle database tables at very high speed.
2. During the one load session, it can load the data into multiple database tables.
3. Different SQL functions can manipulate the data fields before inserting the data into database tables.
4. The user can fix the data errors quickly because it generates a report of errors during the data load.
5. It has the capability to perform the filtering operation and it can accept a different kind of input data formats during the load.
6. It can generate unique sequential key values for the specified columns such as a primary key.

7. It supports the client-server architecture and it can perform the data load across the network.

3.3.2.1. Creating Oracle Tables To Hold Our Proposed Tables

The diagram shown below are 2 different screen captures showing the describe statements on the Person table and the requirement tables. These statements were issued after table creation. This image also shows the enforced column constraints and data-type chosen to conform with the data-type previously been used in Paradox



```
SQL> describe PERSON_TABLE1;
 Name                           Null?    Type
 ------------------------------ -------- ----------------
 PERSONID                       NOT NULL NUMBER(8)
 TITLE                                   VARCHAR2(8)
 FNAME                          NOT NULL VARCHAR2(35)
 SNAME                          NOT NULL VARCHAR2(35)
 NURSEID                                 VARCHAR2(15)
 AGENCYID                                NUMBER(35)
 PAYROLLID                               VARCHAR2(25)
 NINO                                    VARCHAR2(15)
 PERSONNELREF                            VARCHAR2(20)
 UKCCNO                                  VARCHAR2(15)
 CCEXPIRY                                DATE
 PHONENO                                 VARCHAR2(20)
 FAXNO                                   VARCHAR2(25)
 MOBILENO                                VARCHAR2(25)
 PAGERNO                                 VARCHAR2(25)
 EMAIL                                   VARCHAR2(40)
 ALTPHONE                                VARCHAR2(20)
 ADDRESS1                                VARCHAR2(35)
 ADDRESS2                       NOT NULL VARCHAR2(35)
 TOWN                           NOT NULL VARCHAR2(35)
 COUNTY                         NOT NULL VARCHAR2(35)
 POSTCODE                       NOT NULL VARCHAR2(20)
 SHOWSPEC                                VARCHAR2(10)
 SHOWPREF                                VARCHAR2(10)
 SHOWKIN                                 VARCHAR2(10)
 SHOWTYPE                                VARCHAR2(10)
 SCHEMEID                                NUMBER(32)
 GRADEID                                 NUMBER(32)
 SCALEID                                 NUMBER(32)
 SALARY                                  NUMBER(10,7)
 SCALESTART                              DATE
 INFO                                    LONG
 ACTIVE                                  VARCHAR2(10)
 WCID                                    NUMBER(32)
 BANKSTART                               DATE
 INACTIVEDT                              DATE
 DOB                                     DATE
 ORIGIN                                  NUMBER(32)
 SEX                                     VARCHAR2(1)
 MARITALSTATUS                           VARCHAR2(20)
 WTDWAIVER                               VARCHAR2(10)
 NWORKER                                 VARCHAR2(10)
 DAP                                     NUMBER(16)
 LEAVEQUAL                               VARCHAR2(10)
 QUALSTART                               DATE
 PREQUAL                                 VARCHAR2(10)
 TRAINED                                 VARCHAR2(10)
 INACTIVEREASON                          NUMBER(32)
 REACTIVEDT                              DATE
 BANKID                                  NUMBER(32)
 BANKACNUMBER                            VARCHAR2(20)
 BANKROLLNO                              VARCHAR2(20)
 BANKACNAME                              VARCHAR2(50)
 WORKCATID                               NUMBER(32)
 CLOTHIER                                DATE

SQL>
```

```
SQL> describe REQUIREMENTS_TABLE1
 Name                           Null?    Type
 ------------------------------ -------- ----------------
 REQID                          NOT NULL NUMBER(8)
 REQUESTERID                             NUMBER(32)
 REQUESTEDDATE                  NOT NULL DATE
 HOSPID                         NOT NULL NUMBER(32)
 WARDID                         NOT NULL NUMBER(32)
 WARDSHIFT                               NUMBER(32)
 DATES                          NOT NULL DATE
 SPECIALISM                              NUMBER(32)
 GRADE                                   NUMBER(32)
 ALLOCATED                               NUMBER(16)
 WHOTO                                   NUMBER(32)
 NURSEID                                 NUMBER(32)
 ACTHOURS                                NUMBER(5,2)
 ABSREASON                               NUMBER(32)
 DISPLAYHISTORY                          VARCHAR2(10)
 BREASON                                 NUMBER(32)
 ACTSTART                                VARCHAR2(20)
 ACTEND                                  VARCHAR2(20)
 ACTPAY                                  NUMBER(10,7)
 BREAKT                                  NUMBER(5,2)
 LHOURSWKD                               NUMBER(5,2)
 LHOURSENT                               NUMBER(5,2)
 WDCONTRACT                              NUMBER(5,2)
 BANKID                                  NUMBER(32)
 BOOKINGREF                              VARCHAR2(15)
 EPGRADEID                               NUMBER(20)
 EPSCALEID                               NUMBER(20)
 BOOKEDGRADE                             NUMBER(20)
 BOOKEDSCALE                             NUMBER(20)
 BOOKEDSTART                             DATE
 BOOKEDEND                               DATE
 BOOKEDPAIDBREAK                         VARCHAR2(10)
 BOOKEDBREAK                             NUMBER(10,7)
 BOOKEDAFTERXHOURS                       NUMBER(5,2)
 REQPAYYEAR                              NUMBER(20)
 REQPAYWEEK                              NUMBER(20)
 TIMEPAYYEAR                             NUMBER(20)
 TIMEPAYWEEK                             NUMBER(20)
 STATUS                                  VARCHAR2(2)
 TIMEINPUT                               VARCHAR2(25)
 OUTCOME                                 NUMBER(20)
 COMPENSATION                            NUMBER(5,2)
 ALTERED                        NOT NULL VARCHAR2(10)
```

Figure 3.3 : Creating Oracle Table.

3.3.2.2. Preparing the CSV Files

These were the steps taken to produce an error-free CSV file from PARADOX database. This process was done with the help of Microsoft-excel 2007. Much of excel inbuilt functions were leveraged in making data cleansing possible such as "FIND & REPLACE", and I also wrote custom functions to put the date columns in the Oracle accepted format. For a continuous repetitive task of the data-cleansing same structure of paradox tables, an excel macro could be set up to automate some of the routine tasks. A macro is a recorded set of activities that can be replayed on demand.

3.3.2.3. Coding the Control File

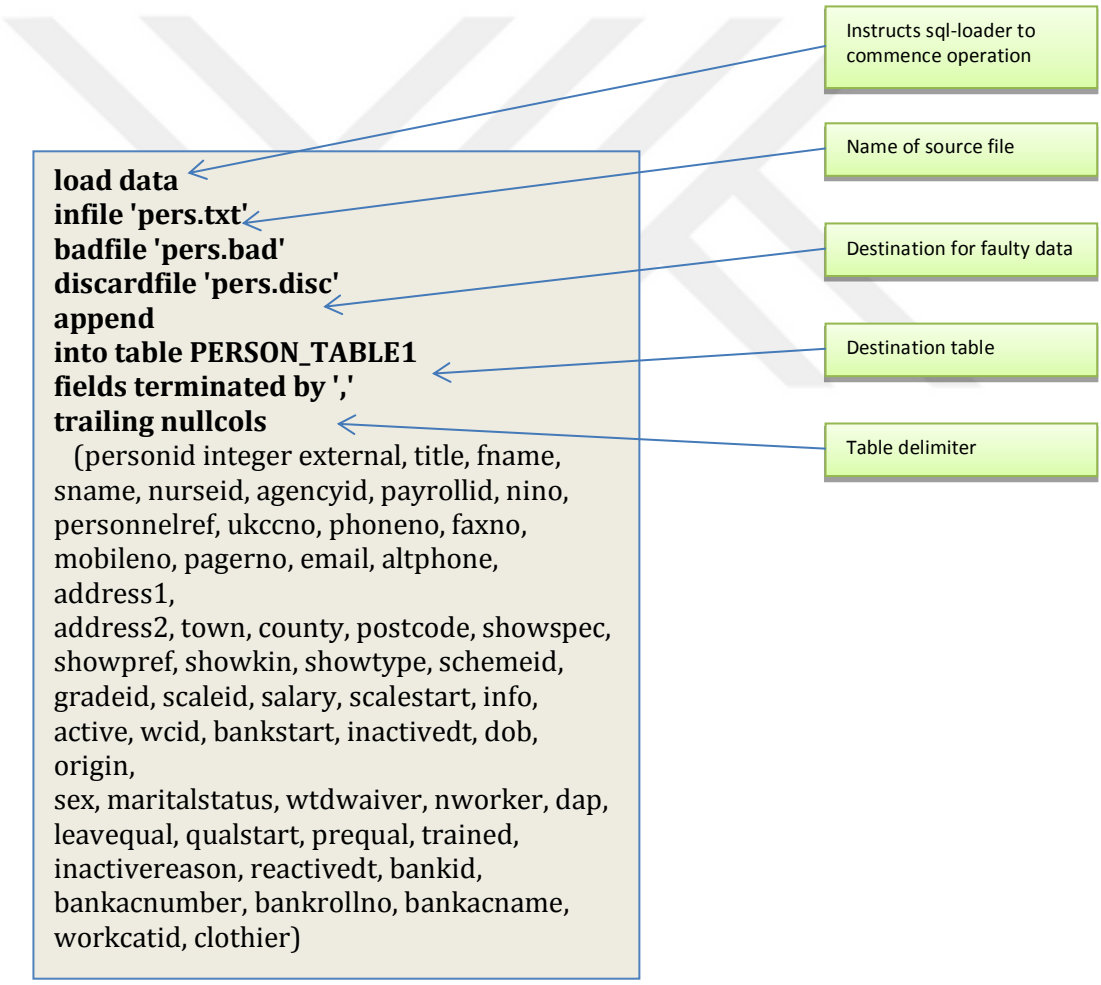


Figure 3. 4 : Control File for Person Table Data Load

The above Control file contains the DDL statements that are part of the Oracle specifications. It was created with Microsoft-WordPad and saved as "Text Document-Ms-Dos Format", and then it appended the extension of "ctl" on the file.

Saving the Control file in this format is very critical because 2 days have been spent trying to resolve the problem saving the Control file in the acceptable format.

3.3.2.4.Performing the SQL-Loader Operation

In these screen-captures, SQL-Loader used to take data from the CSV file and inserted them into an already created database table in ORACLE 11g.

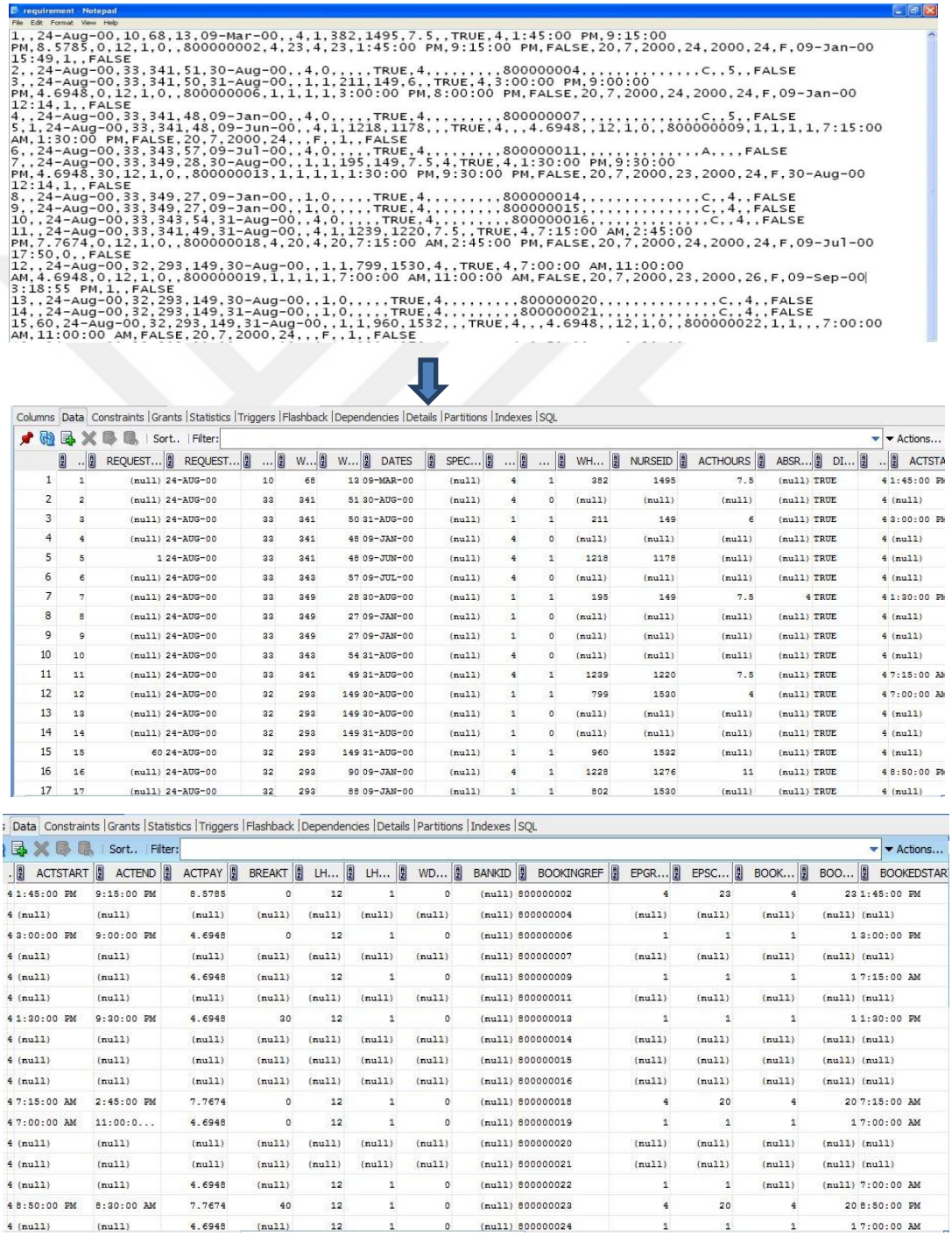


Figure 3.5: Requirement Table From notepad to Oracle

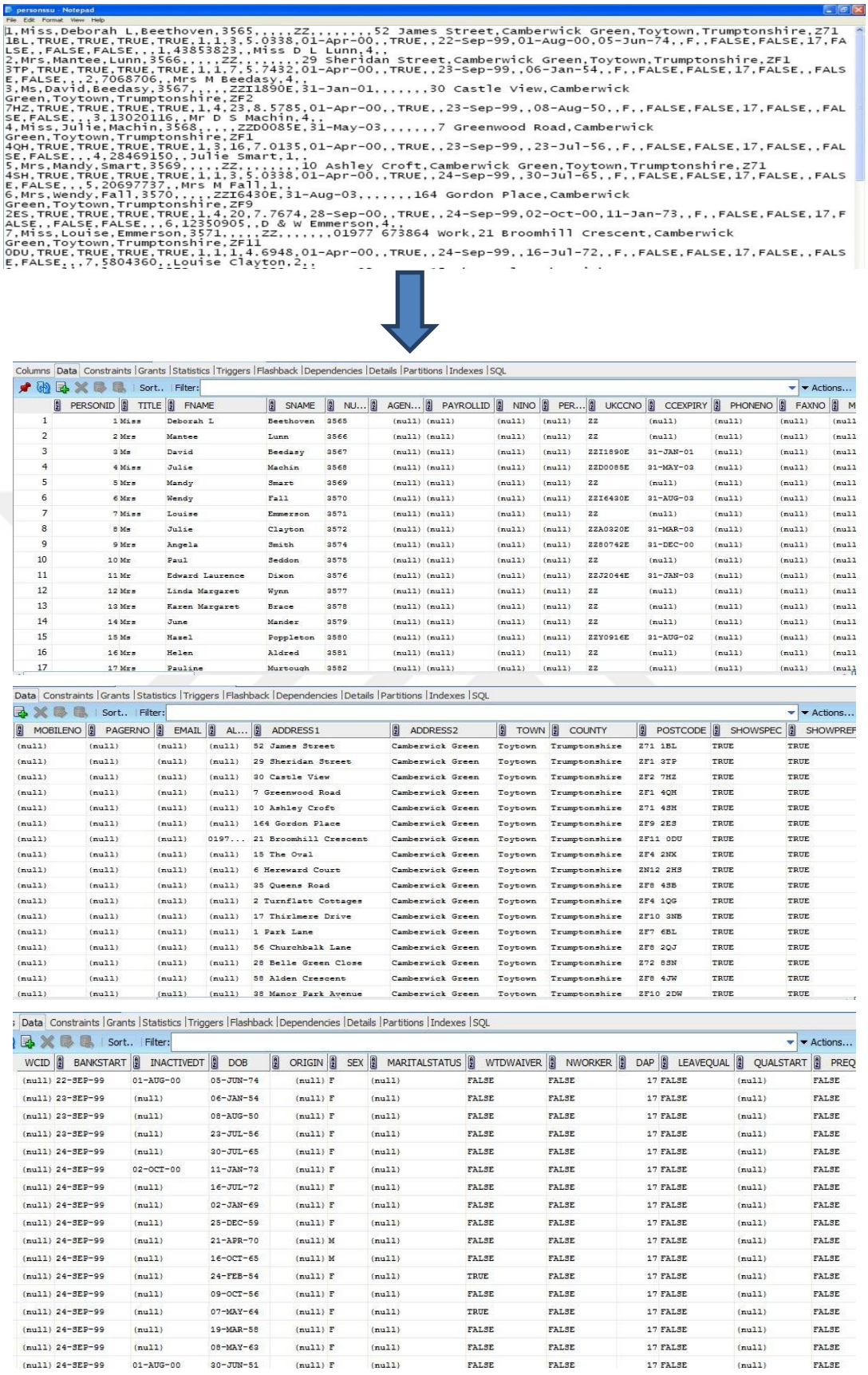


Figure 3.6: Person Table From notepad to Oracle

3.3.2.5. Final Table Clean-up and Data-Quality Management for Experiment1

Since steps were taken to ensure absolute data cleansing and gradual data migration into Oracle with SQLLOADER, the clean-up steps necessary for Experiment1 were very minimal.

- After Loading the database into Oracle, all rows counted by issuing the SQL statement of **"SELECT COUNT(*) FROM PERSON_TABLE1; "** and the result was compared with the record- count obtained from the SQL-launcher utility of bsms. A similar check was done on the two remaining tables to ensure that the record is complete.
- Also, different SQL-query issued to compare the results obtained from destination Oracle tables to the same query that is obtained from the SQL-launcher utility database of BSMS, one of them is shown.

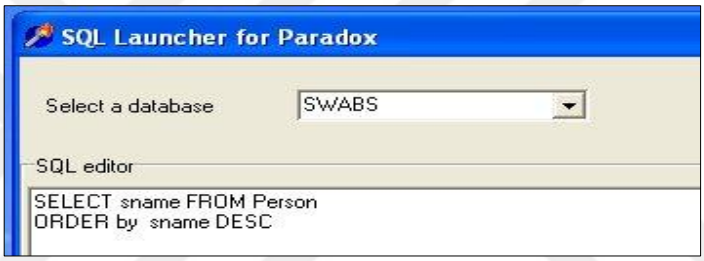


Figure 3.7: Selecting query

- The necessary column-constraints were maintained during the data loading stage to ensure data integrity. But after data transfer was completed, to ensure that the Oracle primary-key technology is enforced in regard to future records, the sequence command displayed in the introductory section was enforced on the table. The number which begins the sequence is chosen because of the current last number of the primary key

3.3.3. Experiment 2 - Migrating from Paradox to Access and Finally to Oracle

In this Scenario, the database table imported from the paradox program folder into an access database that already specifically created for this task. Next, then set up an Oracle ODBC driver to export from Access into Oracle. These steps are shown in the Screen capture below.

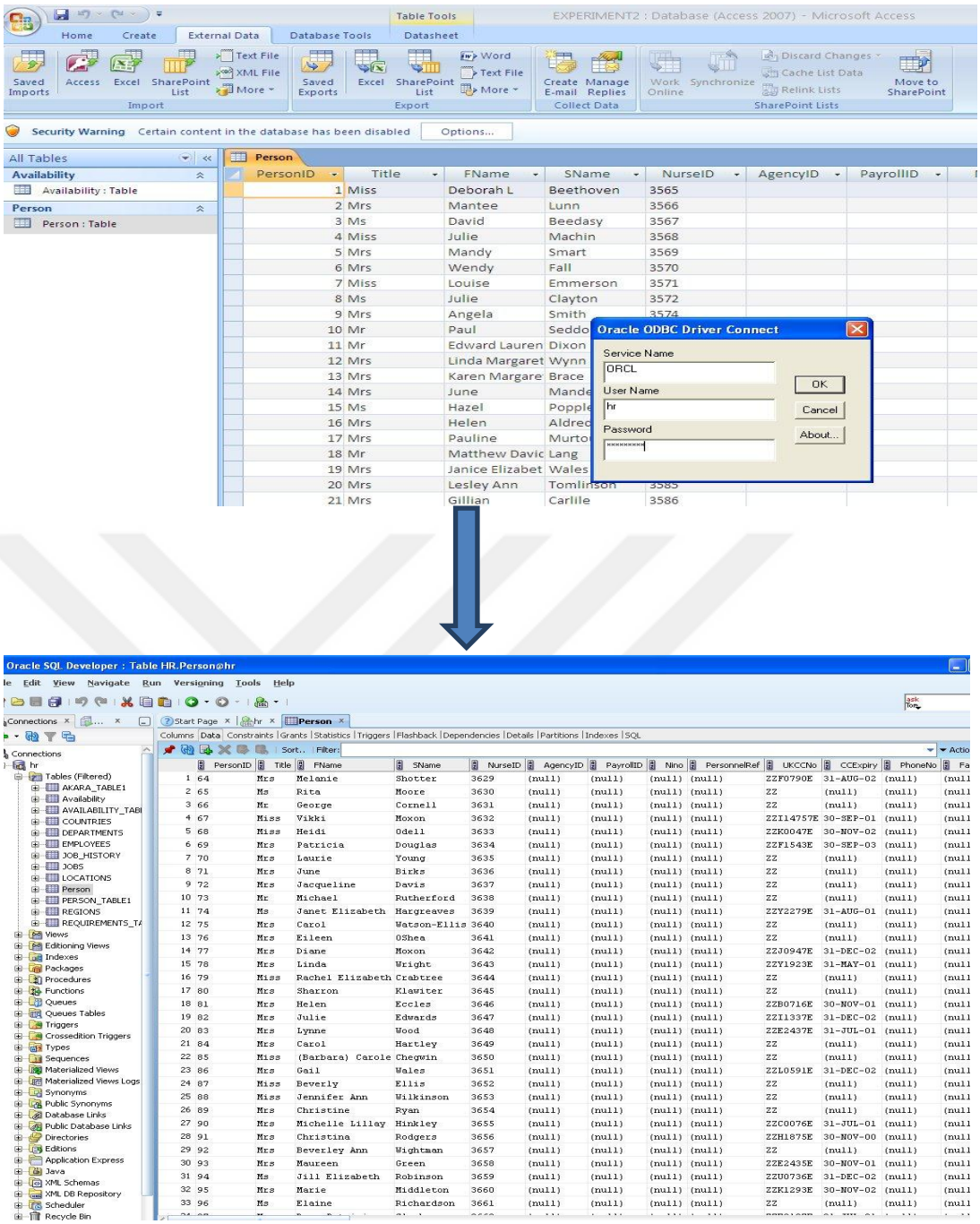
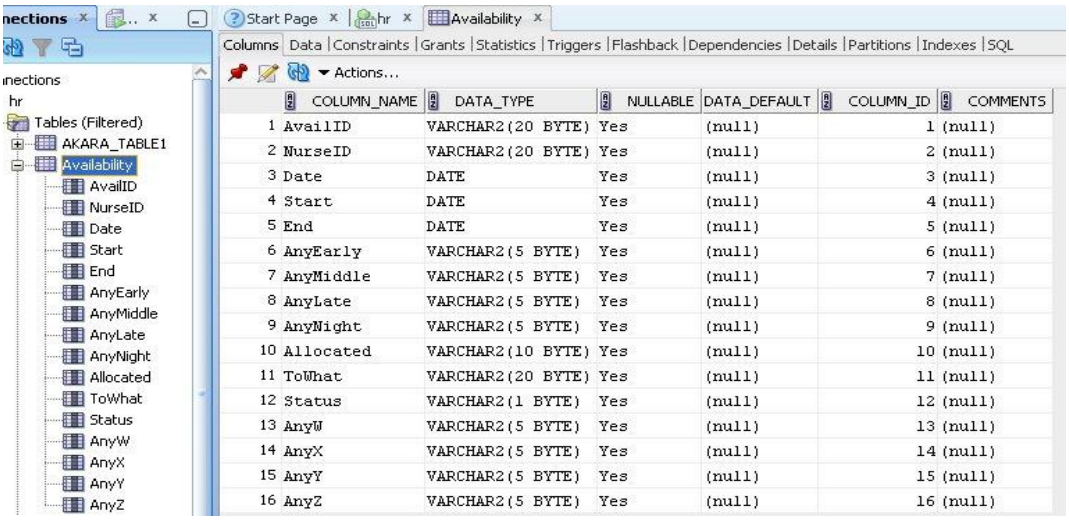


Figure 3. 8: Data migration of person table from access to oracle via ODBC



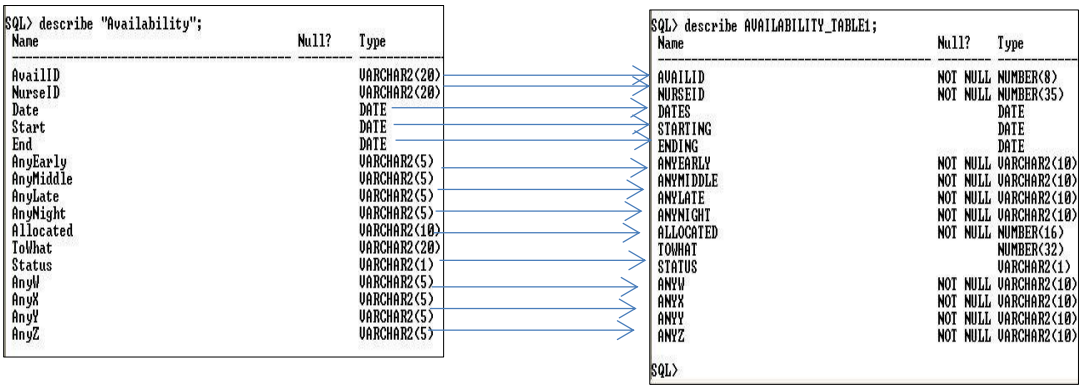
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	AvailID	VARCHAR2(20 BYTE)	Yes	(null)	1	(null)
2	NurseID	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	Date	DATE	Yes	(null)	3	(null)
4	Start	DATE	Yes	(null)	4	(null)
5	End	DATE	Yes	(null)	5	(null)
6	AnyEarly	VARCHAR2(5 BYTE)	Yes	(null)	6	(null)
7	AnyMiddle	VARCHAR2(5 BYTE)	Yes	(null)	7	(null)
8	AnyLate	VARCHAR2(5 BYTE)	Yes	(null)	8	(null)
9	AnyNight	VARCHAR2(5 BYTE)	Yes	(null)	9	(null)
10	Allocated	VARCHAR2(10 BYTE)	Yes	(null)	10	(null)
11	ToWhat	VARCHAR2(20 BYTE)	Yes	(null)	11	(null)
12	Status	VARCHAR2(1 BYTE)	Yes	(null)	12	(null)
13	AnyW	VARCHAR2(5 BYTE)	Yes	(null)	13	(null)
14	AnyX	VARCHAR2(5 BYTE)	Yes	(null)	14	(null)
15	AnyY	VARCHAR2(5 BYTE)	Yes	(null)	15	(null)
16	AnyZ	VARCHAR2(5 BYTE)	Yes	(null)	16	(null)

Figure 3. 9: Transforms integer data types to varchar2

The Oracle ODBC engine automatically transforms the INTEGER data-types in this table into VARCHAR2 and also creates a table with no Constraints. And this requires SQL statements and functions to fix the table.

3.3.3.1.Final Table Clean-up and Data-Quality Management for Experiment2

The ODBC engine does a fairly good job in converting most of the date column data into the default Oracle standard, and whatever field is left unconverted would have to be done manually. The consequence of this method of data transfer is that the resulting oracles table will have to function as a staging table, and from this table, we would transform the data-set and insert it into a well-crafted database-table. A new table created (shown below) with the correct constraints and bigger field-column data -types. Next, the command shown in the image below issued to create a new table in the same schema that contains the old table.



```
SQL> describe "Availability";
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 AvailID                                            VARCHAR2(20)
 NurseID                                            VARCHAR2(20)
 Date                                               DATE
 Start                                              DATE
 End                                                DATE
 AnyEarly                                           VARCHAR2(5)
 AnyMiddle                                          VARCHAR2(5)
 AnyLate                                            VARCHAR2(5)
 AnyNight                                           VARCHAR2(5)
 Allocated                                          VARCHAR2(10)
 ToWhat                                             VARCHAR2(20)
 Status                                             VARCHAR2(1)
 AnyW                                               VARCHAR2(5)
 AnyX                                               VARCHAR2(5)
 AnyY                                               VARCHAR2(5)
 AnyZ                                               VARCHAR2(5)
```

```
SQL> describe AVAILABILITY_TABLE1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 AVAILID                                   NOT NULL NUMBER(8)
 NURSEID                                   NOT NULL NUMBER(35)
 DATES                                              DATE
 STARTING                                           DATE
 ENDING                                             DATE
 ANYEARLY                                  NOT NULL VARCHAR2(10)
 ANYMIDDLE                                 NOT NULL VARCHAR2(10)
 ANYLATE                                   NOT NULL VARCHAR2(10)
 ANYNIGHT                                  NOT NULL VARCHAR2(10)
 ALLOCATED                                 NOT NULL NUMBER(16)
 TOWHAT                                             NUMBER(32)
 STATUS                                             VARCHAR2(1)
 ANYW                                      NOT NULL VARCHAR2(10)
 ANYX                                      NOT NULL VARCHAR2(10)
 ANYY                                      NOT NULL VARCHAR2(10)
 ANYZ                                      NOT NULL VARCHAR2(10)

SQL>
```

Figure 3. 10: Required table data types for managing clean-up and data-quality

```
CREATE TABLE AVAILABILITY_TABLE1
 (      availid NUMBER(8,0) NOT NULL ,
        nurseid NUMBER(35,0) NOT NULL ,
        dates DATE,
        starting DATE,
        ending DATE,
        anyearly VARCHAR2(10 BYTE) NOT NULL,
        anymiddle VARCHAR2(10 BYTE) NOT NULL,
        anylate VARCHAR2(10 BYTE) NOT NULL,
        anynight VARCHAR2(10 BYTE) NOT NULL,
        allocated NUMBER(16,0) NOT NULL ,
        towhat NUMBER(32,0),
        status VARCHAR2(1 BYTE),
        anyw VARCHAR2(10 BYTE) NOT NULL,
        anyx VARCHAR2(10 BYTE) NOT NULL,
        anyy VARCHAR2(10 BYTE) NOT NULL,
        anyz VARCHAR2(10 BYTE) NOT NULL,
        CONSTRAINT REQUIREMENTS_TABLE1_PK PRIMARY KEY (availid) );
```

Figure 3. 11: Codes of required table

As shown in the Sqlplus image below, the data will be move from the table created by ACCESS-ODBC to the table created before with an Oracle "**INSERT INTO**" command that contains functions for data-type transformations. The arrows indicated in the table below clearly shows the intention on which column would be mapped into the final table columns.

As the above diagram shows, our goal is to move data from this staging data on the left to the one on the right

```
INSERT INTO AVAILABILITY_TABLE1
SELECT  to_number('AvailID', 9999999),  to_number('NurseID', 99999999), Date, Start,
End, AnyEarly, AnyMiddle, AnyMiddle, AnyLate, AnyNight, Allocated, ToWhat, Status,
AnyW, AnyY, AnyZ ;
```

Figure 3. 12: Transferring query code

This type of insertion command caused a lot of problems when used in the other 2 tables because they contained a lot of faulty data that Oracle functions could not

transform, and so the rows were only partially inserted or never inserted. But with every attempt, Oracle reveals the row that is faulty, and then manually track it down and correct it with the UPDATE command.

3.3.4. Experiment 3 - Moving Data Microsoft Sql-Server Integration Services

SSIS or SQL-SERVER INTEGRATION SERVICES is robust data integration and filtering application that accompanies Microsoft SQL-server. It contains a set of functionalities that we can use to build enterprise-grade database migration and transformation solutions. This application can do very complex operations on raw data such as standardizing data, transforming and merging data into a single database file system. The most useful ability of SSIS is that it could take all these complex operations and package it as an executable file for future repetition if needed. Two techniques of data movement tried-out with SSIS technology, the first is the "SQL-server import and export wizard".

a) Import and Export Wizard: - This is the simpler method of using SSIS, but its weakness is that you cannot build a complicated package that leverages the T-SQL programming. It is a straight-forward method of getting the job done, and it only gives you some control of specifying data-type and data-sizes during the mapping-stage of the utility. After Importation into SQL-server, T-SQL functions used to modify some of the table columns, and then finally the Import and Export Wizard used again to export the database table to Oracle using the OLEDB driver provided by Microsoft.

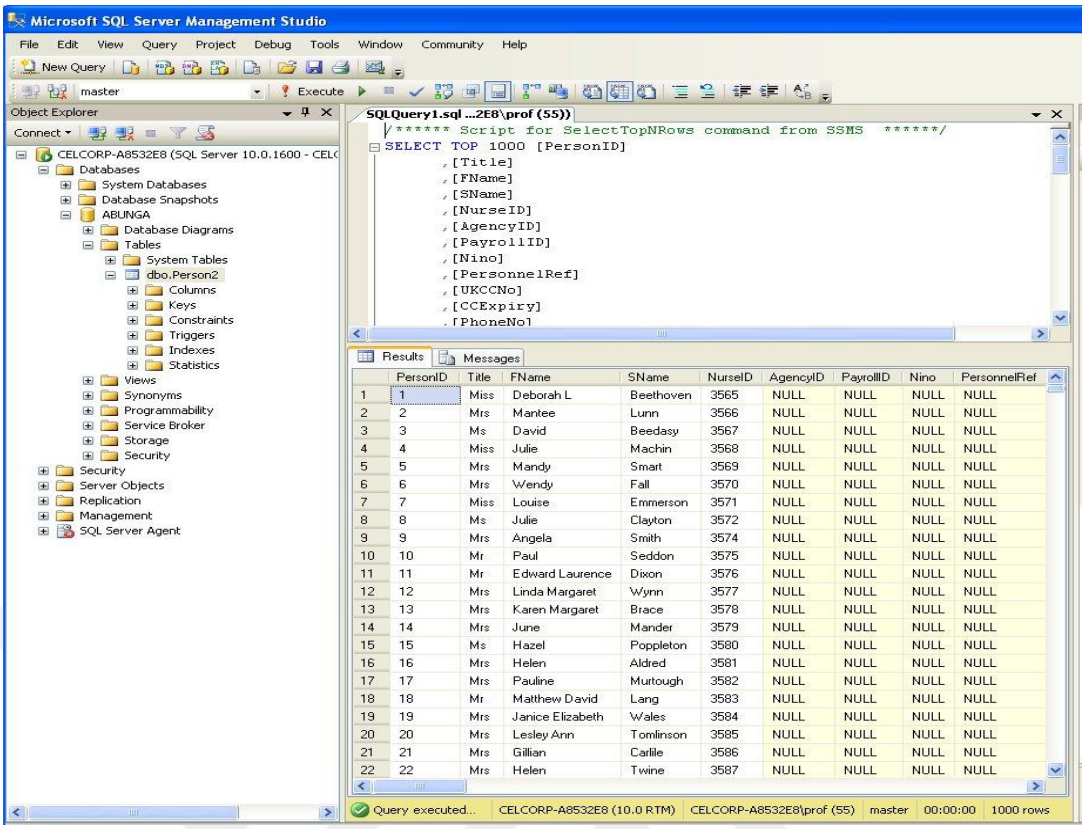


Figure 3. 13: Person table during modification

This Screen-shot shows the Person table during its modification in Sql-Server 2008. In this Scenario, it's been prepared for export to Oracle 11g

b) Using the SSIS-BIDS Interface: - This Interface allows us to engineer a data movement and integration solution. This interface requires that we specify both a data source and a destination file or RDBMS. In this project, the data source was the excel files that were obtained by opening the Paradox database in excel. The diagram below shows some of the various configuration steps we must enforce on the BIDS Interface to ensure a successful data transformation and insertion into the destination folder.

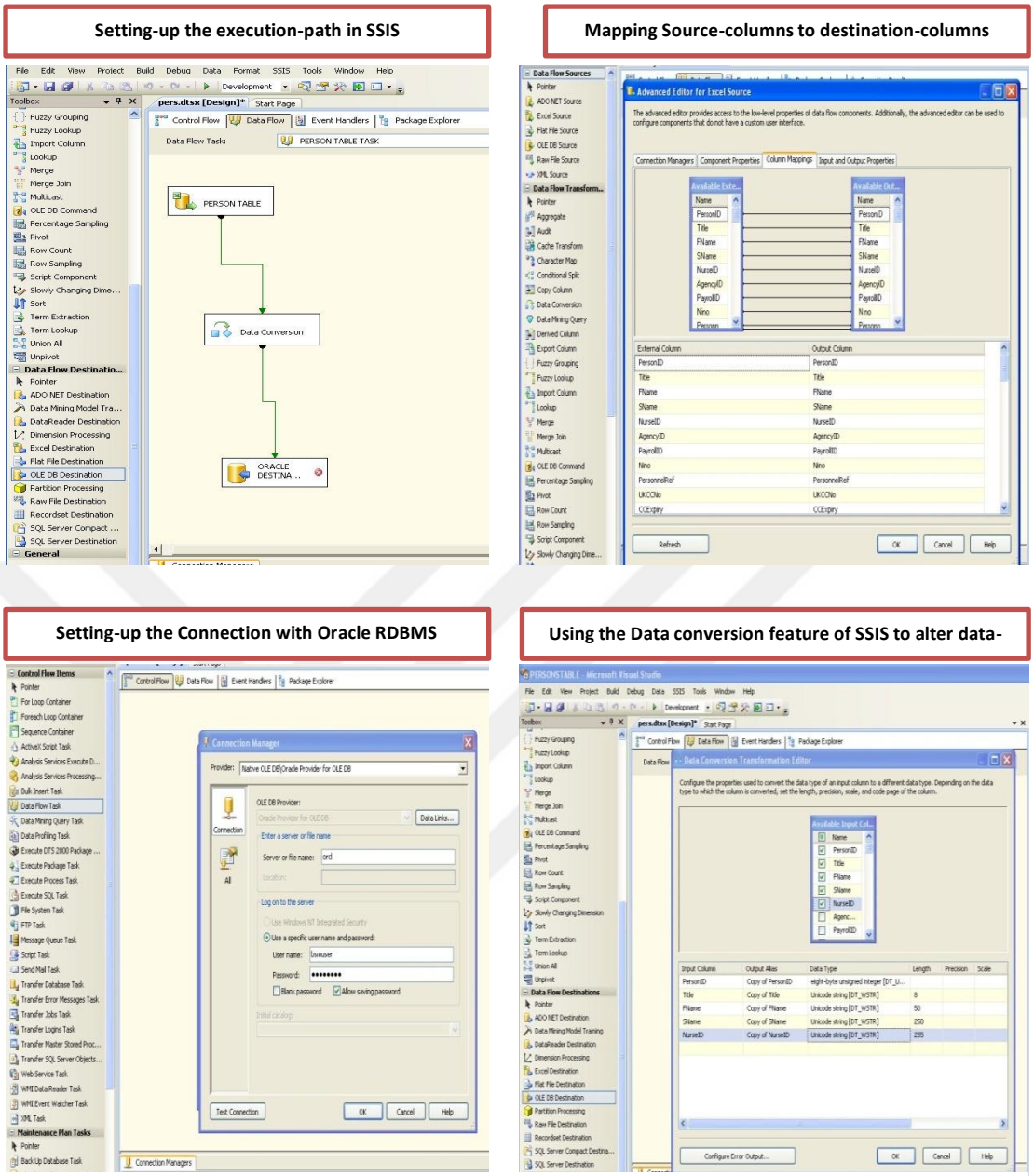


Figure 3. 14: SSIS-BIDS Interface to perform a data movement and integration

3.3.4.1. Final Table Clean-up and Data-Quality Management for Experiment3

During the setup process for experiment3, the platform configured to ignore any errors; it was done to prevent breaks in the process due to faulty data. So, after the data-movement, the error-report took and searched-out the faulty rows and then manually made corrections. Finally altered the Oracle tables and inserted the necessary column constraints. Data-quality checks like those carried-out on experiment1 were also done.

3.3.5. Experiment 4 - Migrating Data to Oracle Using Full Convert

This is a Graphical User Interface tool that reduces the burden of database movement. This software migrates our database tables with high efficiency, creates indexes, and maintains the foreign key relationship on the destination database. These are some unique features of Full Convert listed by its vendor (Spectral Core, 2011).

- Powerful Graphical User Interface
- Easy to use wizard
- Global data type translation rules
- Support for SQL expressions
- Command line support
- Error logging
- Built-in Scheduler

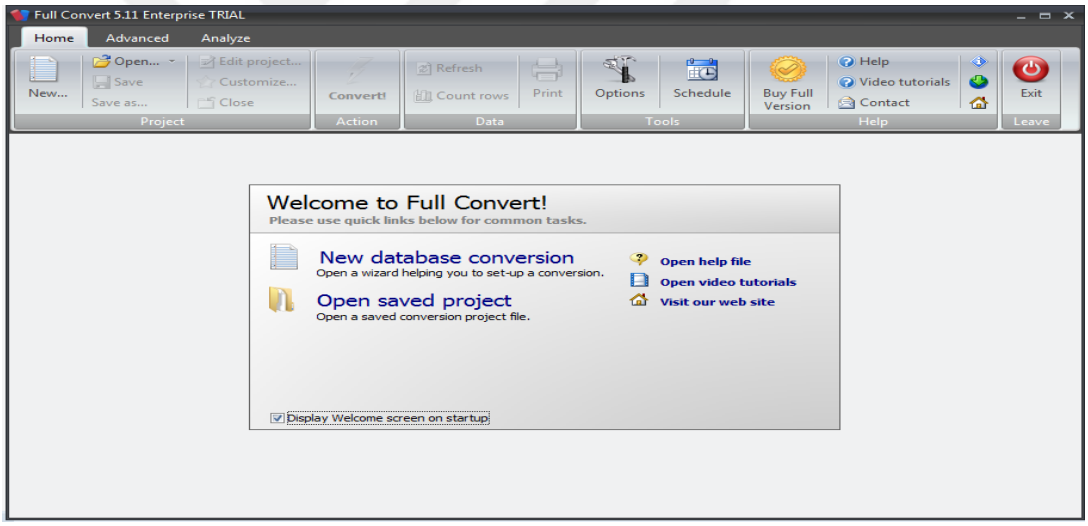


Figure 3. 15: Welcome screen for the trial version of the Full Convert tool.

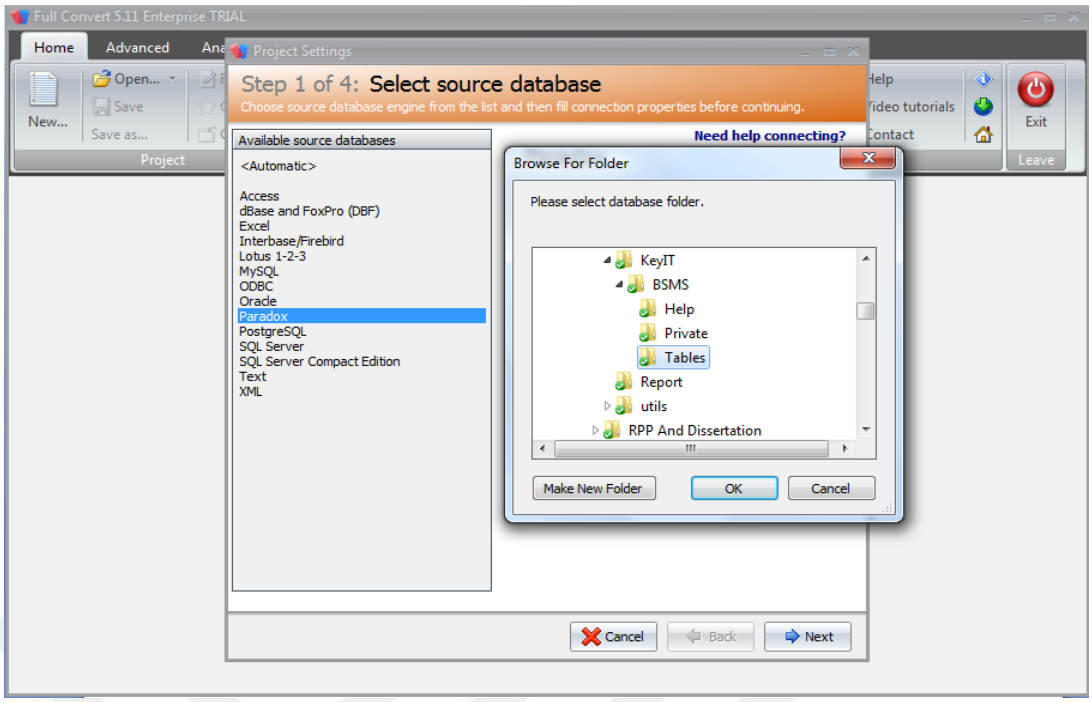


Figure 3. 16: Selecting database type and location folder of the source data.

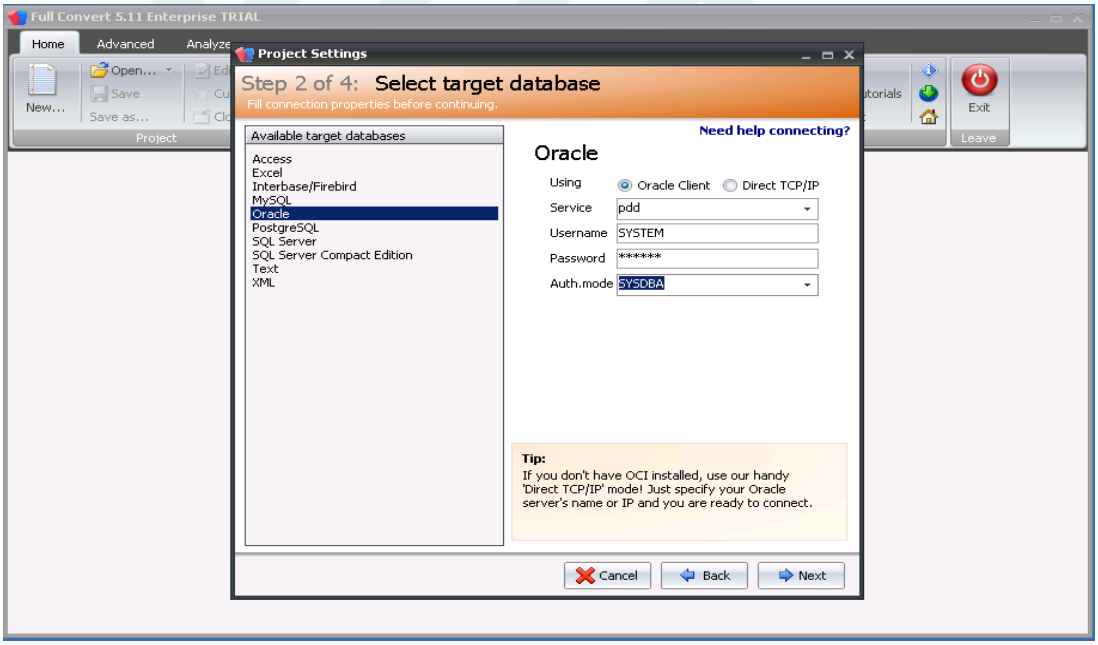


Figure 3. 17: Selecting the destination database and entering the username and password for the Oracle database

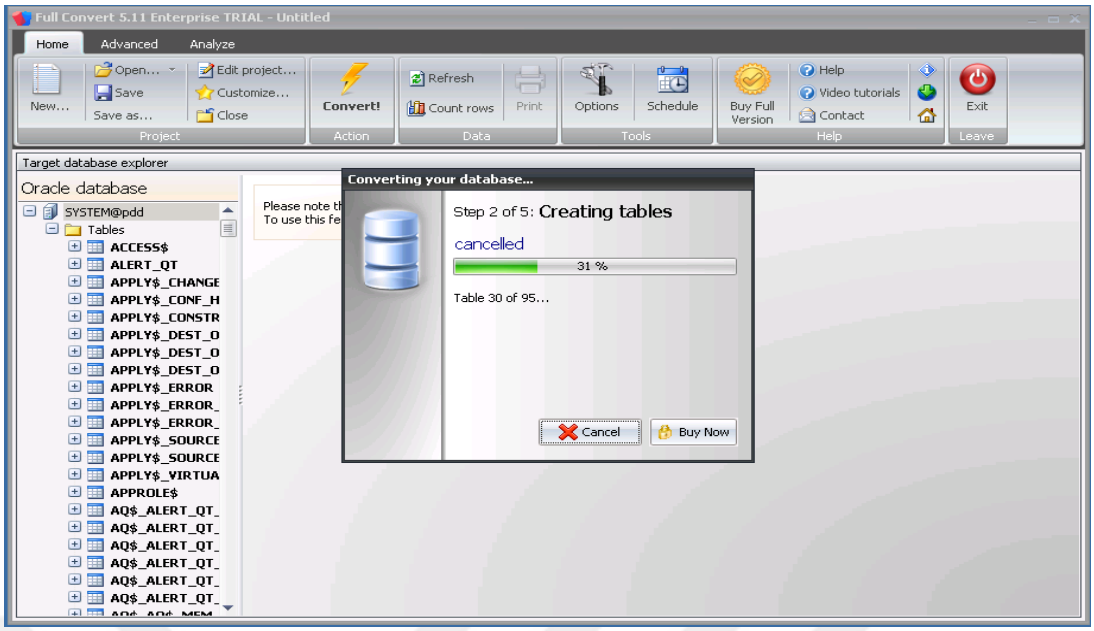


Figure 3. 18: Creating tables in Oracle to hold paradox database tables.

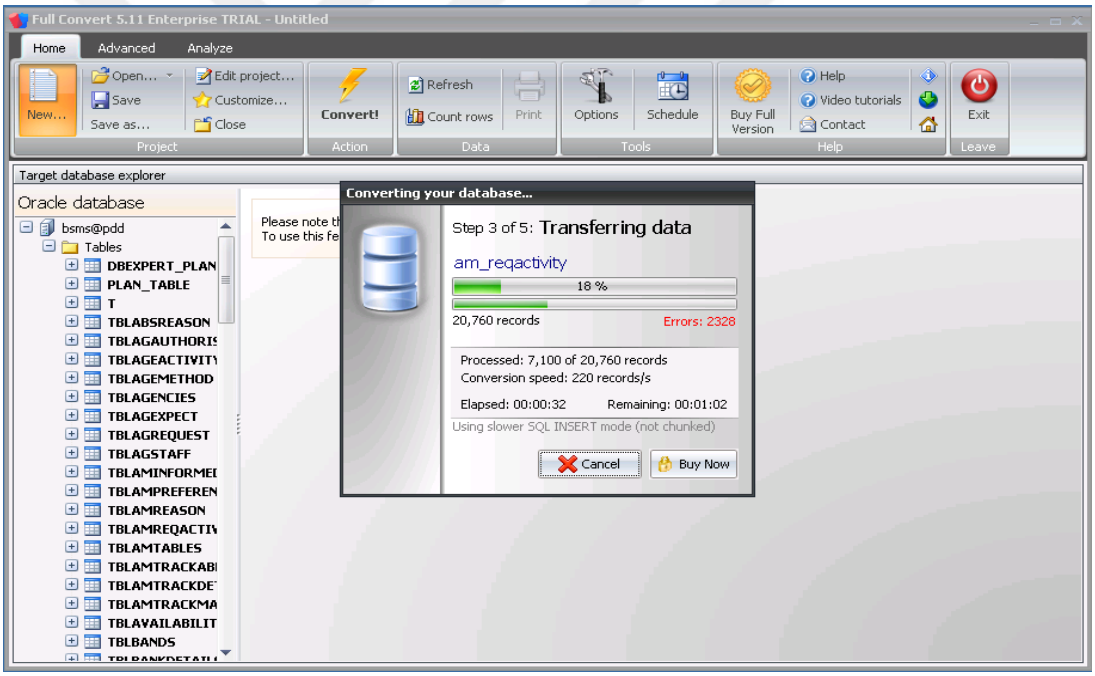


Figure 3. 19: Displaying number of moved tables and a number of errors

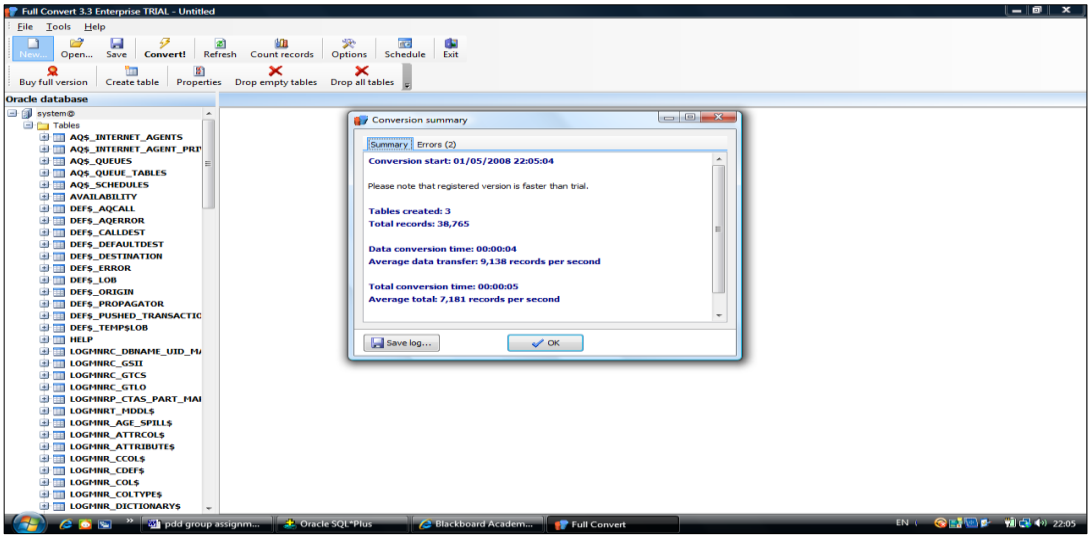


Figure 3. 20: Displaying summary of migrated tables

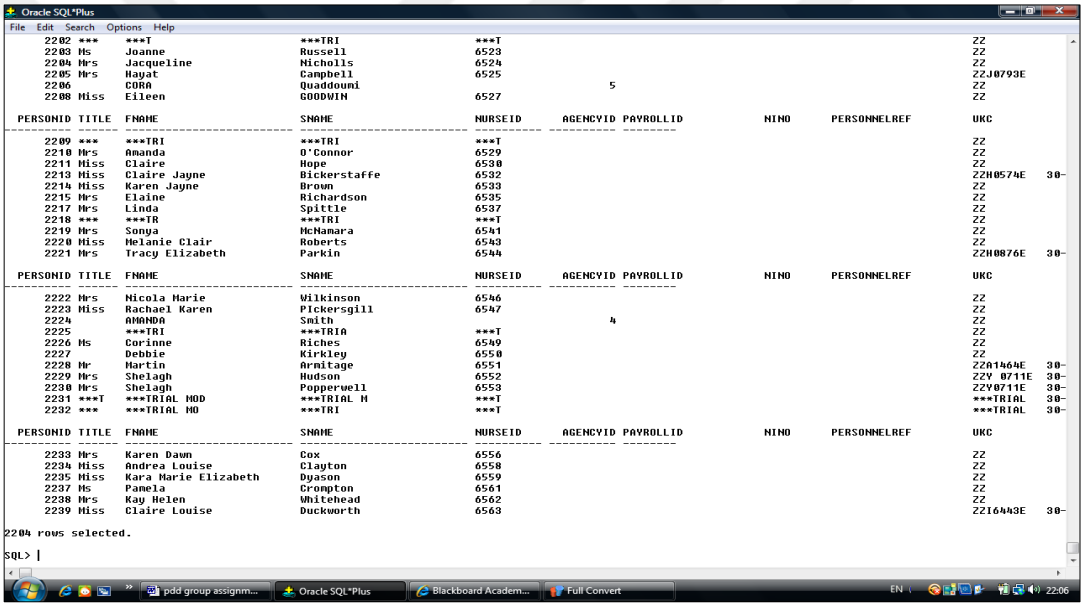


Figure 3. 21: Checking inconsistencies of transferred data using SQL Utility

Figure 3.15: displays the welcome screen for the trial version of the Full Convert tool. We choose the option of new database conversion by double-clicking that option. In Figure 3. 16: we choose the type of database that holds our current database, and then choose the folder location of the source data. Figure 3. 17 here we choose the destination database, and we are prompted to enter the Username and password for the Oracle database we have chosen. Figure 3. 18 shows our software creating tables in Oracle to hold our paradox database tables. Figure 3. 19 shows Our data been migrated, it also displays the number of tables moved and a number of errors encountered. after the full migration, the Software displays a summary of tables migrated Figure 3.20 . Last Figure 3. 21 shows the checking of transferred data for inconsistencies by using SQL utility.

Full Convert like many graphically oriented Softwares performs well in an ideal situation of a perfectly cleansed database. It made several errors such as skipping records that had faulty data types, and it also limited our ability to make conditional choices in the event of faulty data.

CHAPTER 4

FINDINGS AND PERFORMANCE TUNING

4.1. Introduction

During the performance testing stage of this data migration, we made use of various tools to ensure the data was transferred without inconsistencies. For this, we adopted two testing methods which are used to test whether the transferred data is accurate and gives the appropriate results. They are

- SQL Plus
- Keep Tool

These two tools are used to see the consistency and correctness of the data after the migration process takes place.

The tools used to export the data are

- Oracle SQL Plus Utility
- Keep's Tool Hora Utility

SQL plus Utility helps identify the number of rows transferred to Oracle database effectively and efficiently. This could be achieved by using the following query on SQL plus Utility.

Select Count(*) from Person;

4.2. Performance Tuning:

Performance Tuning is more than making an attempt to improve the performance of a query. We made use of Indexing technique in order to improve the performance of a query by retrieving the data from Oracle database at a rapid speed without any inconsistency and loss of data.

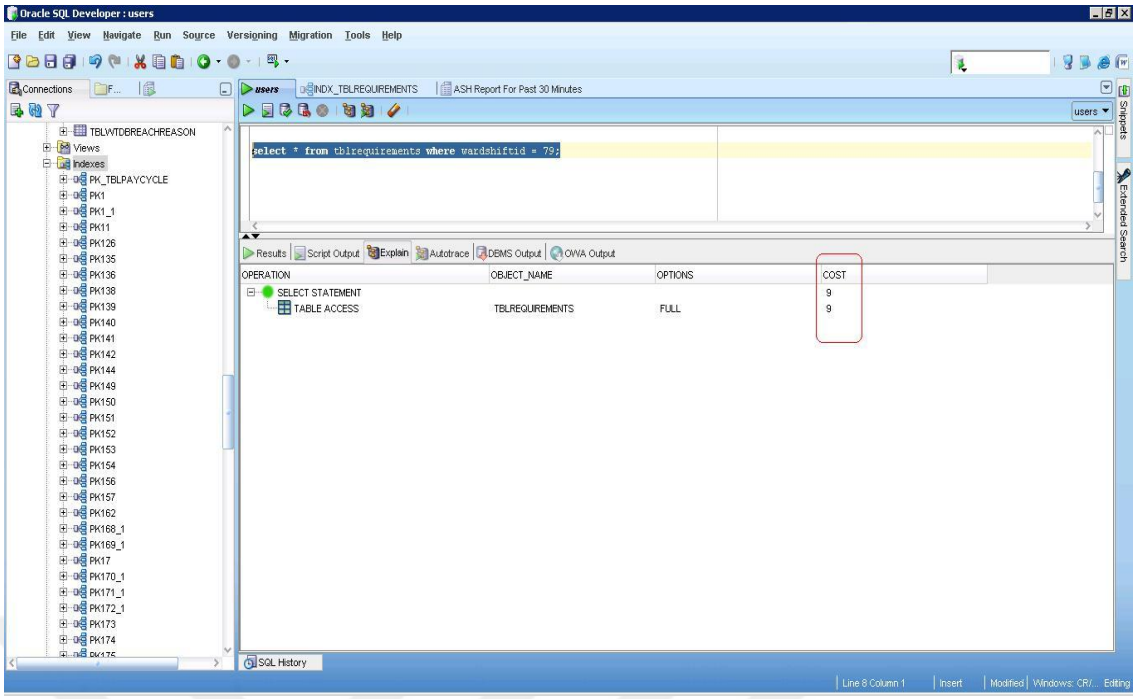


Figure 4. 1: Cost of table requirement before indexing technique

Total cost on table Requirement = 9 + 9 =18 before creating Index. The **cost** is an estimated value proportional to the expected resource use needed to execute the statement with a particular plan. The optimizer calculates the cost of access paths and joins orders based on the estimated computer resources, which includes I/O, CPU, and memory [24].

Partitioning is useful for very large tables. By splitting a large table's rows across multiple smaller partitions, we could make the queries faster because Oracle may have to search only one partition (one part of the table) instead of the entire table to resolve a query. The table might also become easier to manage because the partitioned table's data is stored in multiple parts, it may be easier to load and delete the partitions than in a large table with no partition. From my observation, the largest table in the BSMS database is the Availability table, if we had the benefit of time, we would have tried to perform partitioning testing on the Availability table.

Indexing is a database technology that is usually applied to the columns in a database that is most used during table searches. Oracle uses indexes to decrease the amount of time we consume in finding particular information in a database. The Indexing technology would have been useful in the Person table because it will help us a query and extract nurses names very quickly. The index normally helps the Oracle-optimizer to retrieve queries very fast and efficiently, this technique prevents a full

table scan. Also, Oracle gives us several indexing types that will fit into the particular situation we find ourselves.

Another method of improving speed in our database is through the use of views. A View is a different way of looking at a table or tables. They are representations of the underlying tables that are generated by a query. The use of an indexed materialized view will prevent the database from doing joins every time, and this speeds up efficiency.

The figure below shows the total cost taken to retrieve data from the database after creating Indexing technique.

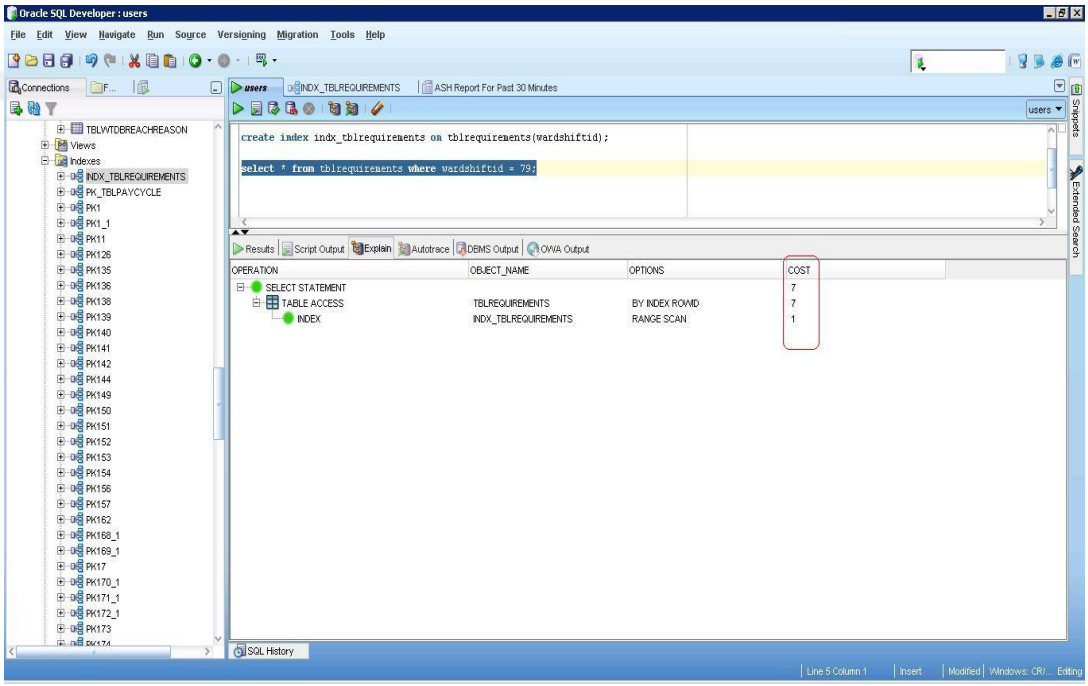


Figure 4. 2: Cost of table requirement after indexing technique

Total Cost for table Requirement is 7 + 7 + 1 = 15 which is less after creating Index. Hence we have seen the total cost for retrieving data from the database is less after creating an index which leads to improving the performance of the database.

4.3. Findings

The intention of this Section is to document and explain what happened during the data migration and optimization project.

The premise of the activities carried out in the Physical Database Design module is to perform ETL operations on the BSMS software, so that the database tables can be moved into an Oracle database environment. ETL means extract, transform and Load

operations. Oracle's centerpiece technology for moving data is called Data Pump. Migrating data using different available technologies has been examined.

4.3.1. From Paradox to Access to Oracle

Moving data into Oracle with the help of Microsoft-Access technology was a relatively comfortable operation. After carefully researching the structure of the three database tables (Person table, Availability table, and Requirement table) that we planned to migrate, we then had to build an ODBC connection between the Microsoft-Access software and the Oracle database platform. For researching the structure of the Paradox database, we made use of the Table Documenter and P20 software; these two applications enabled us to understand the table design and data-types which subsequently allowed for verification of success when the tables were inserted into the Oracle database.

The greatest fascination with this type of data migration is the very important role that the ODBC driver had to perform during the entire process. It really fascinated me that this 20-year-old API technology that was originally designed to access any type of data source still manages to perform at a highly efficient level during data migration tasks, and it seems the ODBC driver will remain a powerful ETL tool for years to come.

The most significant problem has encountered during the whole process of data movement into Oracle was the problem of data-type consistency after migration. This is a really troubling feature of ODBC driver that adds a lot of additional time to completely fix. The ODBC platform also removes any table constraint that existed in the Original Paradox tables. So this implied that manually inserted the table constraints such as primary keys on the destination tables. Getting the Paradox database tables into the Microsoft-Access software. This task required to first use two utility software named Table documenter and P20 to evaluate and document the column data-types and constraints on the source database. What discovered from this particular migration operation was the ease of moving data from one database to another could be made much simpler by simply using a basic tool like Microsoft access and sometimes using Microsoft-excel as a cleaning tool.

Retrospectively, if the process needed to be repeated, would try and reduce the very difficult task of cleaning and repairing tasks that happened in Oracle environment

after the data movement. This will achieve by attempting to make the data types conform to Oracle data types when the data are still in the Microsoft-access database software. Would probably copy the tables from Microsoft-access database and use Microsoft-excel to clean it and also search for inconsistencies. the excel will use to change the Boolean data-types into a data-type that Oracle will easily work with, and then return the data to Microsoft-access before moving it to Oracle.

4.3.2. From Paradox to Oracle Using SQL-Loader

This was a very technically challenging operation. This operation required that we convert the tables of the Paradox database into a CSV format before using SQL-Loader to import the tables to the Oracle database. The difficulty comes mainly from the fact that the CSV file must be in a perfect condition before we import the data into Oracle, any little imperfection in data causes serious compromises in the final data and sometimes leads to a complete collapse of the migration operation.

In order to prevent the constant problems we faced during the SQL loader operation, first move the necessary Paradox tables into excel, and then use the built-in functions of excel to clean up the data. After the necessary cleaning is done, then saved the table in CSV format, so it can be in a format necessary for use in PowerPivot.

The most important lesson acquired from this method of data movement is how to accurately prepare a control file. Then carefully explore the Oracle website to obtain the details of the syntax of the control file, and even with all these researches done, still there is lots of issues with SQL loader until perfected the data loading routine.

Also, a lot of challenges we experienced during the process of SQL loader operation was because we lacked the ability to interpret the error messages being presented to us by the error log-files after each attempt.

4.3.3. From Full-Convert to Oracle

This application helped move the database tables from the Paradox database into the Oracle database. But like most automated applications, mistakes and skipped rows were prevalent in the final tables. The conclusions reached after numerous problems were that databases are very complex systems and attempting to automate the processes that a human database administrator carries out by using an application is

very difficult and often times will lead to a collapse of the migration process or very faulty data in the destination tables.

A very poignant example of this was the Full convert application transforming the Boolean data-types in Paradox to character data-types in Oracle. Another sad example is the complete loss of referential integrity and primary key integrity that happened to the database tables after the moving operation had been completed.

the process of using automated tools in data movement is that after the database has been moved into the destination database, verification tests on the destination databases should always done to ensure that the data transfer is efficient. And if it happen to encounter a partially flawed database table, then manually rely on the relevant SQL queries to fix the problems.

4.4. Optimization of Database in Oracle

The data - pump is a utility that was introduced in Oracle 11g to comprehensively move Oracle databases between Oracle platforms These following topics are a reflection of the various optimization activities that we carried-out and also on some performance optimization that we could have performed but did not have the required time to accomplish it.

4.4.1. Indexing Technique

Indexes are usually used to dramatically increase the performance of some queries; they are schema objects that are used to speed up data retrieval. By indexing a frequently queried table like the nurse person-table on the queried column, we would have reduced the disk input-output operation, thus making the database instance more efficient at data retrieval.

The highest cost method to satisfy a query is to run a table to scan where the oracle is going through the tables and fetching the table row by row, and column by column,

Due to the fact that leveraging a “where” and “join” clauses in extracting data from most of the columns, critical indexing tests made on those columns that contain the “where” data and criteria.

Also, the Bitmap Indexing system would have liked to experiment; this is because this indexing system sometimes has a lower cost index of non-numeric search parameters like names.

4.4.2. Utilization of Views

A view is a stored SQL-select query, if there is enough time for more comprehensive testing routines; the high-cost SQL statements would converted into materialized-views and then re-engineered the BSMS software codes to query the materialized-views instead of performing complex join operations on-demand. Adding indexes would have also attempted to the materialized views; this would have contributed to the significant performance increase.

4.4.3. Bind Variables Technique

In the world of performance optimization, it is necessary to avoid ad-hoc queries as much as humanly possible. Also, due to the fact that splitting the BSMS software into a client-server system, a very positive performance increment will be realized if transformed the most constantly used queries into "Bind variables". A massive increase would experience in performance because Oracle will then have to rely on soft parses for its execution plan. The query statement that extracts the details of nurses by using their names is a great candidate for a query that should utilize a bind variable. This is because, with a networked architecture, the various client systems will be querying the Oracle database, and with no bind variables in the execution plan, Oracle will have to do hard parses, and this is very inefficient for the database engine.

4.5. Monitoring Execution Path with Explain Plan

The SQL explain plan statement used to discover the execution path of some of the frequently used SQL queries. This can sometimes be a very tricky procedure because interpreting the results of an explain-plan statement requires some skill and finesse to accomplish. When using this explain plan method, two principal objectives focused, and they are

- How best to eliminate table scans
- trying to ensure that Oracle chose the optimal table joining order

Due to the time constraints that experienced during this work, the sufficient time to make modifications could not find that on some slow execution-plan was chosen by Oracle. According to documentation in the Oracle website, the Oracle Optimizer-

hints could leverage to make forced modifications to the Oracle execution path, but the constraints in time that experienced made it impractical during this work.

Another critical performance boosting method is the Oracle SQL profiles technology, this would have allowed us to store a persistent execution plan that have determined is better than what the Oracle execution plan is currently utilizing.

CHAPTER 5

RECOMMENDATION AND CONCLUSION

5.1. Recommendation

We have used various types of tools for migrating data from Paradox table to Oracle database. After going through the entire migration process and keeping all tools in mind, we recommend SQL Loader tool will be the best to use for migration purpose. This tool is considered an integrated utility of the Oracle database. This tool is used to transfer heterogeneous data in a friendly manner. We got the desired accuracy and correctness of data after the migration process by using SQL Loader tool. The following are the advantages of using this tool

- Generate Error Report that helps to identify the error easily and rectify them.
- Generate Report containing information related to a number of records transferred, time take to transfer the data etc.

5.2. Conclusion

After careful consideration and using different types of tools for the migration process, we have successfully migrated the data without inconsistency and loss of data. However, we came across different issues during migration using different tools. The most common error we found in using every tool is data types are not matching when compared to Oracle Database data types. But we have rectified this data types issue by changing or modifying data types according to Oracle database. We have recommended SQL Loader as one of the best tools to be used for the migration process. However, we don't recommend Full Convert because only trial version of this tool is available. We used Indexing technique to improve the performance of the database. Hence we are able to migrate the data successfully and test it after the migration process done and we achieved an accurate result.

5.3 Future Work

Since a majority of cloud applications are data driven, database management systems (DBMSs) powering these applications form a critical component in the cloud software stack. One of our future work could be using cloud platforms for performing database migration, using the concepts of data fission and data fusion, enabling lightweight elasticity using low cost live database migration.

On the other hand, as a future work and extension to this study, other Indexing techniques will be used for the purpose of comparison and measurement the performance of migration method. A comparative study could be done in order to come out with the suitable indexing technique, in a specific scenario.

REFERENCES

[1] Anand, N. (2014). ETL and its impact on Business Intelligence, *International Journal of Scientific and Research Publications*, ***4***(2), 1.

[2] Brian Fretwell (2018) Linked in profile. Available at: https://www.linkedin.com/in/brian-fretwell-b658aa5/. Accessed April 10, 2018.

[3] Levine R. (2009). *Data migration strategies*. Retrieved from "http://wikibon.org/wiki/v/Data_migration_strategies".

[4] Wu, B., Lawless, D., Bisbal, J., Richardson, R., Grimson, J., Wade, V., & O'Sullivan, D. (1997, September). The butterfly methodology: A gateway-free approach for migrating legacy information systems, *In Proceedings. Third IEEE International Conference on Engineering of Complex Computer Systems (Cat. No. 97TB100168)* (pp. 200-205). IEEE.

[5] Russom, P. (2006). Best practices in data migration, *Renton/USA*.

[6] Velimeneti, S. (2016). Data Migration from Legacy Systems to Modern Database.

[7] Redman, T. C. (1995). Improve data quality for competitive advantage, *Sloan management review*, ***36***(2), 99-108.

[8] Matthes, F., Schulz, C., & Haller, K. (2011, September). Testing & quality assurance in data migration projects, In *2011 27th IEEE international conference on software maintenance (ICSM)* (pp. 438-447). IEEE.

[9] Rakov, I. (1997). Quality of Information in Relational Databases and Its Use for Reconciling Inconsistent Answers in Multidatabases, *Fourth Doctoral Consortium on Advanced Information Systems Engineering*.

[10] Brodie, M. L. (1995). Migrating legacy systems: gateways, interfaces & the incremental approach, Morgan Kaufmann Pub.

[11] Oladipo, F. O., & Raiyetumbi, J. O. (2017). Re-Engineering Legacy Data Migration Methodologies in critical sensitive systems, *Journal of Global Research in Computer Science*, ***6***(11).

[12] International Business Systems. How to Implement ERP for Your Business: All at Once or In Phases? 2013. Available at: http://globial.com/globialtalksbusiness/how-to-implement-erp-for-your-business-all-at-once-or-in-phases/ Accessed 29.3. 2018.

[13] Oracle Corporation, (2011). Successful data migration. Oracle white paper. Available at: http://www.oracle.com/technetwork/middleware/oedq/successful-data-migration-wp-1555708. Accessed 9.2.2018.

[14] Fouché, G., & Langit, L. (2011). Foundations of SQL server 2008 R2 business intelligence. Apress.

[15] Jensen, C. S., Pedersen, T. B., & Thomsen, C. (2010). Multidimensional databases and data warehousing, *Synthesis Lectures on Data Management*, ***2***(1), 1-111.

[16] Harinath, S., Pihlgren, R., Lee, D. G. Y., Sirmon, J., & Bruckner, R. M. (2012). Professional Microsoft SQL Server 2012 Analysis Services with MDX and DAX, John Wiley & Sons.

[17] Nielsen, P., & Parui, U. (2011). *Microsoft SQL server 2008 bible (Vol. 607).* John Wiley & Sons.

[18] Oracle Corporation, (2013). *Express Mode Loading with SQL Loader in Oracle Database12c*. Available at: https://www.oracle.com/technetwork/database/enterprise/edition/learnmore/sqlldr-express-mode-wp-1991038. Accessed 15.12.2018.

[19] Rouse M. (2017, May). What is data? - Definition from WhatIs.com. Retrieved from "https://searchdatamanagement.techtarget.com/definition/data".

[20] IBM Corporation, (2010). What is a database management system? https://www.usg.edu/galileo/skills/unit04/primer04_01.html. Accessed 15.5.2018.

[21] DbTalks (2016, March, 24). What Is Paradox Database? Retrieved from "http://www.dbtalks.com/article/what-is-paradox-database2/".

[22] Studytonight, (2018). What is a database? Available at: https://www.usg.edu/galileo/skills/unit04/primer04_01.html. Accessed 18.3.2018.

[23] Oracle. (2005). *Introduction to the Oracle Database*. Retrieved from "https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm"

[24] Moore, V. (2003). Oracle R Database Performance Tuning Guide 10g Release 1 10.1 Part No. *B10752-01, December*.

APPENDIX 1

DETAILS OF THE PARADOX DATABASE TABLES

Details for Table: absreason
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
AbsID	Auto-incrementing	
AbsSDesc	String [15]	
AbsLDesc	String [40]	

Record count: 11

Details for Table: ageactivity
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
ActID	Auto-incrementing	i
Activity	String [25]	

Record count: 3

Details for Table: ageexpect
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
ExpID	Auto-incrementing	i
ExpValue	String [20]	

Record count: 5

Details for Table: agelog
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
ALogID	Auto-incrementing	i
ARequestID	integer(32bit)	
ActivityID	integer(32bit)	
ContactID	integer(32bit)	
MethodID	integer(32bit)	
LogDate	Date	
LogTime	Time	

Record count: 8

Details for Table: agemethod
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX

Details for Table: am_trackables
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
TrackableID	Auto-incrementing	i
Description	String [40]	

Record count: 3

Details for Table: am_trackdetail
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
AuditID	integer(32bit)	i
TrackableInfoID	integer(32bit)	i
Value	integer(32bit)	
Date	Date	

Record count: 4768

Details for Table: am_trackmaster
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
AuditID	Auto-incrementing	i
Date	Date	
Time	Time	
UserID	integer(32bit)	
UserShortName	String [5]	
TableName	integer(32bit)	
Action	integer(16bit)	
Key	integer(32bit)	

Record count: 5554

Details for Table: arequests
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
AgencyID	integer(32bit)	i
WardReqID	integer(32bit)	i
ARequest	Auto-incrementing	
RequesterID	integer(32bit)	
RequestedDate	Date	
BReason	integer(32bit)	

AgeMethodID Auto-incrementing i
AgeMethod String [25]
Record count: 1

Details for Table: agencies
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
AgencyID Auto-incrementing i
AgencyName String [50]
MainContact integer(16bit)
Address1 String [40]
Address2 String [40]
Town String [40]
County String [40]
Postcode String [20]
PhoneNo String [20]
FaxNo String [20]
PagerMobileNo String [20]
Email String [30]
Website String [30]
AltContact String [40]
AltPhone String [20]
Record count: 6

Details for Table: agencyntype
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
ANTypeID Auto-incrementing i
ANTDesc String [40]
Record count: 0

Details for Table: agencyrates
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
AgencyID integer(32bit) i
GradeID integer(32bit) i
ShiftTypeID integer(32bit) i
NurseTypeID integer(32bit) i
Fee Real
Commission Real
InclusiveCost Real
Record count: 0

Details for Table: agencystype
Database Alias: SWABS
FIELD NAME FIELD TYPE
HospID integer(32bit)
WardID integer(32bit)
WardShift integer(32bit)
Date Date
Specialism integer(32bit)
Grade integer(32bit)
Allocated integer(16bit)
WhoTo integer(32bit)
ActStart Time
ActEnd Time
ActHours Real
ActPay Real
AbsReason integer(32bit)
DisplayHistory Boolean
BreakT Real
AuthorisedBy integer(32bit)
AuthDate Date
AuthTime Time
PlacedWith integer(32bit)
PlaceDate Date
PlaceTime Time
Status integer(16bit)
Expectation integer(16bit)
Record count: 822

Details for Table: assdetail
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
AssType integer(16bit) i
AssID Auto-incrementing i
AssDate Date
AssRenews Date
AssRef String [15]
AssLoc String [50]
Record count: 0

Details for Table: astaff
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
AgencyID integer(32bit) i
StaffID Auto-incrementing i
Name String [40]
Record count: 15

Details for Table: Availability
Database Alias: SWABS

INDEX
AShiftID Auto-incrementing i
ASTDesc String [25]
StartTime Time
EndTime Time
Record count: 0

Details for Table: agencystypes
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
AgencyID integer(32bit) i
AShiftID integer(32bit) i
ASTDesc String [25]
StartTime Time
EndTime Time
Record count: 0

Details for Table: am_authorisor
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
AM_AuthID Auto-incrementing i
AM_AuthTitle String [6]
AM_AuthInitials String [6]
AM_AuthFName String [20]
AM_AuthSName String [20]
Record count: 75

Details for Table: am_davailabilities
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
AvailID integer(32bit) i
NurseID integer(32bit)
Date Date
Start Time
End Time
AnyEarly Boolean
AnyMiddle Boolean
AnyLate Boolean
AnyNight Boolean
Allocated integer(16bit)
ToWhat integer(32bit)
Record count: 0

Details for Table: am_drequirements
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
AvailID Auto-incrementing i
NurseID integer(32bit)
Date Date
Start Time
End Time
AnyEarly Boolean
AnyMiddle Boolean
AnyLate Boolean
AnyNight Boolean
Allocated integer(16bit)
ToWhat integer(32bit)
Status String [1]
AnyW Boolean
AnyX Boolean
AnyY Boolean
AnyZ Boolean
Record count: 21329

Details for Table: bankdetails
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
BankID Auto-incrementing i
BName String [60]
Btype String [10]
BSort String [8]
BContact String [40]
BAdd1 String [60]
BAdd2 String [60]
BTown String [40]
BCounty String [40]
BPostcode String [12]
BPhone String [15]
BFax String [15]
BEmail String [60]
Record count: 1633

Details for Table: Banned
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
BannedID Auto-incrementing i
LetterRef String [15]
LetterDate Date
LetterSource String [30]
Surname String [30]
FirstName String [30]

INDEX
ReqID integer(32bit) i
RequesterID integer(32bit)
RequestedDate Date
BReason integer(32bit)
HospID integer(32bit)
WardID integer(32bit)
WardShift integer(32bit)
Date Date
Specialism integer(32bit)
Grade integer(32bit)
Allocated integer(16bit)
WhoTo integer(32bit)
ActStart Time
ActEnd Time
ActHours Real
ActPay Real
AbsReason integer(32bit)
DisplayHistory Boolean
BreakT Real
Record count: 0

Details for Table: am_informed
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
AM_InformedID Auto-incrementing i
AM_InformedTitle String [6]
AM_InformedInits String [6]
AM_InfFName String [20]
AM_InfSName String [20]
Record count: 7

Details for Table: am_preferences
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
TableID integer(32bit) i
ActionID integer(32bit) i
TrackableID integer(32bit) i
Record count: 6

Details for Table: am_reason
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
AM_ReasonID Auto-incrementing i
AM_ReasonSDesc String [15]
AM_ReasonLDesc String [40]

SecondName String [30]
ThirdName String [30]
MaidenName String [30]
Alias String [50]
DoB Date
Details Text memo
Record count: 1

Details for Table: breachreasons
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
BReasonID Auto-incrementing i
BRDescription String [30]
Record count: 10

Details for Table: breason
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
BReasonID Auto-incrementing i
BReasonShort String [15]
BReasonLong String [40]
Record count: 10

Details for Table: cancelled
Database Alias: SWABS
FIELD NAME FIELD TYPE
INDEX
CancelledID Auto-incrementing i
CDate Date
CTime Time
CancelledBy integer(32bit)
CReason String [30]
ByWard Boolean
CWardID integer(32bit)
CRequestorID integer(32bit)
CShiftStart Time
CShiftEnd Time
CGradeID integer(32bit)
CSpecID integer(32bit)
ReBooked Boolean
Informed String [30]
InformedBy String [30]
InfMethod String [20]
InfDate Date
InfTime Time
Record count: 0

PayNurseExtra Boolean
Amount Real
Record count: 8

Details for Table: am_reqactivity
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
RID	integer(32bit)	i
When	Date and time	i
Type	integer(32bit)	
Who	String [5]	

Record count: 20760

Details for Table: AM_Tables
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
TableID	Auto-incrementing	i
Description	String [40]	
TableName	String [40]	

Record count: 2

Details for Table: requirements
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
ReqID	Auto-incrementing	i
RequesterID	integer(32bit)	
RequestedDate	Date	
HospID	integer(32bit)	
WardID	integer(32bit)	
WardShift	integer(32bit)	
Date	Date	
Specialism	integer(32bit)	
Grade	integer(32bit)	
Allocated	integer(16bit)	
WhoTo	integer(32bit)	
NurseID	integer(32bit)	
ActHours	Real	
AbsReason	integer(32bit)	
DisplayHistory	Boolean	
BReason	integer(32bit)	
ActStart	Time	
ActEnd	Time	
ActPay	Real	
BreakT	Real	
LHoursWkd	Real	
LHoursEnt	Real	

Details for Table: comment
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
NurseID	integer(32bit)	i
CommentID	Auto-incrementing	i
CType	integer(32bit)	
Comment	Text memo	
MadeBy	String [30]	
TParty	String [30]	
CDate	Date	
CTime	Time	
SysOp	String [6]	

Record count: 1959

Details for Table: coursemaster
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
NurseID	integer(32bit)	i
CourseID	integer(32bit)	i
Renews	Date	
CourseDt	Date	

Record count: 90

Details for Table: courses
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
TrainID	Auto-incrementing	i
CourseName	String [40]	
CourseDate	Date	
CertRenewel	Date	
RenewDays	integer(16bit)	

Record count: 8

Details for Table: ctypes
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
CType	Auto-incrementing	i
CTypeShort	String [15]	
CTypeLong	String [40]	

Record count: 21

Details for Table: customer
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX

WdContract Real
BankID integer(32bit)
BookingRef String [10]
EPgradeid integer(32bit)
EPscaleID integer(32bit)
BookedGrade integer(32bit)
BookedScale integer(32bit)
BookedStart Time
BookedEnd Time
BookedPaidBreak Boolean
BookedBreak Real
BookedAfterxHours Real
ReqPayYear integer(16bit)
ReqPayWeek integer(16bit)
TimePayYear integer(16bit)
TimePayWeek integer(16bit)
Status String [1]
TimeInput Date and time
Outcome integer(16bit)
Compensation Real
Altered Boolean
Record count: 15232

Details for Table: seqs
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
Table String [3] i
LastChanged Date
LastNum integer(32bit)
Record count: 2

Details for Table: shift_types
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
ShiftCode String [1]
ShiftType String [6]
ShiftStart Time
ShiftEnd Time
Record count: 8

Details for Table: specialism
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
SpecialismID Auto-incrementing i
ShortDesc String [15]
LongDesc String [80]
TrustID Auto-incrementing i
ShortName String [6]
LongName String [60]
UseCommP Boolean
CommR Real
CommP Real
ERNIP Real
Record count: 14

Details for Table: declinedreasons
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
DeclinedID Auto-incrementing i
Description String [30]
Record count: 6

Details for Table: dislike
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
DislikeID Auto-incrementing i
PersonID integer(32bit)
WardID integer(32bit)
HospID integer(32bit)
Record count: 40761

Details for Table: ep_changes
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
ReqID integer(32bit) i
NID integer(32bit) i
Uneek Auto-incrementing i
Code String [5]
PayPerHour Money
Rate Money
From Time
To Time
Duration Time
Cost Money
Type integer(16bit)
Record count: 21

Details for Table: ep_rules
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
Code String [5] i

Record count: 69

Details for Table: swabsusers
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
UserID	Auto-incrementing	i
UserName	String [40]	
UserInits	String [4]	
UserLevel	String [10]	
Password	String [10]	
InputTime	Boolean	
ModifyTime	Boolean	

Record count: 2

Details for Table: trusts
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
TrustID	Auto-incrementing	i
SName	String [20]	
LName	String [50]	

Record count: 3

Details for Table: Ward
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
HospitalID	integer(32bit)	i
WardID	Auto-incrementing	i
WardName	String [60]	
Costcode	String [12]	
WardPhone	String [15]	
Clothier	Boolean	
Notes	Text memo	

Record count: 602

Details for Table: ward_shifts
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
HospID	integer(32bit)	i
WardID	integer(32bit)	i
Shift_Code	String [5]	i
WardShiftID	Auto-incrementing	
Shift_Description	String [25]	
Shift_Type	String [1]	
ShiftStart	Time	
ShiftEnd	Time	
ScaleID	integer(32bit)	i
Type	integer(16bit)	
Date	Date	
Day	integer(32bit)	
StartTime	Time	
EndTime	Time	
ShiftPercent	integer(16bit)	
TrainedRate	integer(16bit)	
UnTrainedRate	integer(16bit)	
EnhancedRate	Real	
Comment	String [30]	

Record count: 330

Details for Table: ep_times
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
ReqID	integer(32bit)	i
NID	integer(32bit)	i
Uneek	Auto-incrementing	i
Code	String [5]	
PayPerHour	Money	
Rate	Money	
From	Time	
To	Time	
Duration	Time	
Cost	Money	
Type	integer(16bit)	

Record count: 21823

Details for Table: grade
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
GradeID	Auto-incrementing	i
ShortDesc	String [15]	
LongDesc	String [50]	
AvPay	Real	

Record count: 0

Details for Table: groupings
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
LOMGrpID	Auto-incrementing	i
GroupID	integer(32bit)	
UnitID	integer(32bit)	

Record count: 0

Breaktime Real
AfterXHours Real
PaidBreak Boolean
Record count: 692

Details for Table: wcentres
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
WCID Auto-incrementing i
WCDesc String [40]
Record count: 1

Details for Table: wcmaster
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
WCID integer(32bit) i
WardID integer(32bit) i
Record count: 0

Details for Table: wgroups
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
WGroupID Auto-incrementing i
GroupSDesc String [15]
GroupLDesc String [40]
LOMlevel integer(16bit)
Record count: 0

Details for Table: wkpatterns
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
MaxDays integer(16bit)
Record count: 17

Details for Table: Worddots
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
LtrID Auto-incrementing i
Type String [5]
Desc String [40]
Path String [50]
Filename String [50]
Record count: 7

Details for Table: groups
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
GroupID Auto-incrementing i
GroupSDesc String [15]
GroupLDesc String [40]
LOM integer(16bit)
Record count: 1

Details for Table: groupward
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
GroupID integer(32bit)
WardID integer(32bit)
Record count: 0

Details for Table: group_master
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
LOM integer(16bit) i
GroupID integer(32bit) i
Unit integer(32bit) i
Record count: 0

Details for Table: hospital
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
TrustID integer(32bit) i
HospitalID Auto-incrementing i
HospitalName String [60]
ShortDesc String [15]
HUseCommP Boolean
HCommR Real
HCommP Real
HERNIP Real
Record count: 37

Details for Table: inactivereason
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
InactiveReasonID Auto-incrementing i
InactiveReasonDesc String [40]
Record count: 9

Details for Table: workcat
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
WorkCatID	Auto-incrementing	i
WorkCatType	String [6]	
WorkCatDesc	String [50]	

Record count: 4

Details for Table: workoffered
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
RID	integer(32bit)	i
NID	integer(32bit)	i
When	Date and time	i
Accepted	Boolean	
Reason	integer(32bit)	

Record count: 13803

Details for Table: wtdbreach
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
BreachID	Auto-incrementing	i
NurseID	integer(32bit)	
WT	integer(16bit)	
Waiver	Boolean	
SDate	Date	
SFrom	Time	
STo	Time	
Chk1	Real	
Chk2	Real	
Chk3	Real	
Chk4	Real	
Chk5	Real	
Chk6	Real	
Chk7	Real	
Chk8	integer(16bit)	
Chk9	integer(16bit)	
UserID	integer(32bit)	
BreachDt	Date	
BreachTime	Time	
BreachReason	integer(32bit)	

Record count: 35

Details for Table: nursewtdw
Database Alias: SWABS

Details for Table: informed
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
ReqID	integer(32bit)	i
AvailID	integer(32bit)	i
When	Date and time	i
NurseID	integer(32bit)	
NurseInfD	Date	
NurseInfT	Time	
NurseInfBy	String [20]	
WReqID	integer(32bit)	
WReqInfD	Date	
WReqInfT	Time	
WReqInfBy	String [20]	
AllocFlag	integer(16bit)	

Record count: 3200

Details for Table: like
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
LikeID	Auto-incrementing	i
PersonID	integer(32bit)	
WardID	integer(32bit)	
HospID	integer(32bit)	

Record count: 17258

Details for Table: loms
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
LOM1ID	integer(32bit)	
LOM1Name	String [30]	
LOM2ID	integer(32bit)	
LOM2Name	String [30]	
LOM3ID	integer(32bit)	
LOM3Name	String [30]	

Record count: 0

Details for Table: lom_master
Database Alias: SWABS

FIELD NAME	FIELD TYPE	INDEX
LOM	integer(16bit)	i
Description	String [20]	

Record count: 2

FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
RecID Auto-incrementing i
SignedDt Date
DocRef String [15]
DocLocation String [40]
Terms integer(16bit)
RenewDt Date
Notice integer(16bit)
TermDt Date
TermRef String [15]
TermLocation String [40]
Record count: 1

Details for Table: origin
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
OriginID Auto-incrementing i
Origin String [40]
Record count: 7

Details for Table: pcreason
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
PCReasonID Auto-incrementing i
PCReason String [30]
Record count: 8

Details for Table: Person
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
PersonID Auto-incrementing i
Title String [6]
FName String [25]
SName String [25]
NurseID String [10]
AgencyID integer(32bit)
PayrollID String [20]
Nino String [9]
PersonnelRef String [20]
UKCCNo String [10]
CCExpiry Date
PhoneNo String [20]
FaxNo String [20]
MobileNo String [20]

Details for Table: nextofkin
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
PersonID integer(32bit) i
NOKName String [30]
NOKTitle String [6]
NOKRel String [15]
NOKAdd1 String [50]
NOKAdd2 String [50]
NOKTown String [50]
NOKCounty String [50]
NOKPostcode String [12]
NOKPhone String [20]
NOKAltPhone String [20]
Record count: 1988

Details for Table: nursecontract
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
PersonID integer(32bit) i
WardID integer(32bit)
ContractHours Real
GradeonWard integer(32bit)
OpeningBalance Real
TrustID integer(32bit)
OtherHours Real
WTDW Boolean
Effective Date
Record count: 461

Details for Table: nursecontractmany
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
ContractID Auto-incrementing i
PersonID integer(32bit)
WardID integer(32bit)
CHours Real
WGrade integer(32bit)
CStartDt Date
CEndDt Date
Record count: 15

Details for Table: NurseLeave
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX

PagerNo String [20]
Email String [50]
AltPhone String [20]
Address1 String [40]
Address2 String [40]
Town String [40]
County String [40]
Postcode String [20]
Showspec Boolean
Showpref Boolean
ShowKin Boolean
ShowType Boolean
SchemeID integer(32bit)
GradeID integer(32bit)
ScaleID integer(32bit)
Salary Real
ScaleStart Date
Info Text memo
Active Boolean
WCID integer(32bit)
BankStart Date
InactiveDt Date
DOB Date
Origin integer(32bit)
Sex String [1]
MaritalStatus String [15]
WTDWaiver Boolean
NWorker Boolean
DAP integer(16bit)
LeaveQual Boolean
QualStart Date
PreQual Boolean
Trained Boolean
InactiveReason integer(32bit)
ReactiveDt Date
BankID integer(32bit)
BankACNumber String [20]
BankRollNo String [20]
BankACName String [50]
WorkCatID integer(32bit)
Clothier Date
Record count: 2204

Details for Table: pm_grade
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
SchemeId integer(32bit) i
GradeID Auto-incrementing i
NurseID integer(32bit) i
LeaveDate Date i
HoursTaken Real
When Date and time
PayAvg Real
Cost Real
Record count: 0

Details for Table: NurseLeaveOpening
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
LeaveYear Date i
OpeningBalance Real
Record count: 0

Details for Table: nursenwork
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
Limit integer(16bit)
Every integer(16bit)
DAP integer(16bit)
Hazard Boolean
AssType integer(16bit)
AssDate Date
AssRef String [15]
AssLoc String [40]
Record count: 0

Details for Table: nurseowork
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
TrustID integer(32bit) i
OHours Real
Record count: 2

Details for Table: nursepaymany
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
PersonID integer(32bit) i
SchemeID integer(32bit) i
GradeID integer(32bit) i
ScaleID integer(32bit) i

GradeLDesc String [40]
SDesc String [20]
Record count: 27

Details for Table: pm_payweek
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
weekending Date i
PWyear integer(16bit)
PWweek integer(16bit)
PWwkstart Date
PWtimesheetsBy Date
PWpayon Date
Frozen Boolean
Record count: 105

Details for Table: pm_scale
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
GradeID integer(32bit) i
ScaleID Auto-incrementing i
SDesc String [20]
StdRate Real
Record count: 108

Details for Table: pm_scheme
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
SchemeID Auto-incrementing i
SchemeLDesc String [50]
SDesc String [20]
Record count: 8

Details for Table: Preferences
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
TrustName String [60]
Default_Hospital integer(32bit)
AllocTolerence Time
BetweenShifts Real
WarnHours Real
LeaveYearStarts Date
LeaveYearEnds Date
EveryxHoursWorked Real
EntitledToxHoursOff Real
PStartDt Date i
PEndDt Date
PayID Auto-incrementing
Rate Real
ReasonCode integer(32bit)
SysUser String [5]
SysDate Date
SysTime Time
Record count: 2273

Details for Table: nursepref
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
WardID integer(32bit) i
Record count: 0

Details for Table: nursespec
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
SpecID integer(32bit) i
Record count: 6471

Details for Table: nursetype
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
TypeID integer(32bit) i
Record count: 0

Details for Table: NurseWorkAreas
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
NurseID integer(32bit) i
HospID integer(32bit) i
Record count: 3764

Details for Table: reports2
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
GroupID integer(16bit) i
ReportID Auto-incrementing i

UnpaidBreak Real
AfterxHours Real
Audit Boolean
IncompleteShiftAllocationBoolean
MinComplete integer(16bit)
LockRetryInterval integer(16bit)
DemoData Boolean
LastVersion String [10]
Updating Boolean
DefaultView integer(16bit)
EPay Boolean
WeekStarts integer(16bit)
PaidBreaks Boolean
NormDAP integer(16bit)
NWLimit integer(16bit)
NWEvery integer(16bit)
NWDAP integer(16bit)
DRest integer(16bit)
WRest integer(16bit)
LeaveQP integer(16bit)
WTDAllocCheck Boolean
DisableHols Boolean
WTDBreachReasons Boolean
RegUser String [60]
RegKey String [25]
ALEntitlementWeeks integer(16bit)
ALAVGHrsWeek Real
ALMAXLeave Real
ALAVGPayOver integer(16bit)
ALShaftNurse Boolean
Record count: 1

RepName String [15]
RepDesc String [60]
SelectionC integer(16bit)
Record count: 28

Details for Table: reqcomments
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
ReqID integer(32bit) i
When Date and time i
Type integer(32bit)
MadeBy String [5]
Comment String [50]
Record count: 258

Details for Table: reqcommenttypes
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
Type Auto-incrementing i
Short String [15]
Long String [40]
Record count: 6

Details for Table: requesters
Database Alias: SWABS
FIELD NAME FIELD TYPE INDEX
RequesterID Auto-incrementing i
HospID integer(32bit)
WardID integer(32bit)
Name String [40]
Initials String [6]
Record count: 2068