AUTOMATION TOOL FOR ROLE TRANSITION IN ORACLE DATA GUARD DATABASES BY ADOPTING ORACLE BEST KNOWN PRACTICES

SATHIS KRISHNAN

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

AUTOMATION TOOL FOR ROLE TRANSITION IN ORACLE DATA GUARD DATABASES BY ADOPTING ORACLE BEST KNOWN PRACTICES

SATHIS KRISHNAN

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER IN SOFTWARE ENGINEERING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: (I.C/Passport No:)

Matric No:

Name of Degree:

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Field of Study:

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

[AUTOMATION TOOL FOR ROLE TRANSITION IN ORACLE DATA GUARD DATABASES BY ADOPTING ORACLE BEST KNOWN PRACTICES]

ABSTRACT

Database availability is utmost critical factor for most of the businesses to sustain and continue their business. The required availability varies within an organization and unavailability is intolerable. Failed role transition is one of the causes for the unavailability. The predominant Oracle software with Data Guard databases versions is commonly used to run the most demanding, mission-critical database driven applications such as manufacturing, e-banking, etc. In this complex administration of Oracle Data Guard environment maintaining and executing an efficient role transition requires an automated tool to optimize an error free role transition. Having discovered by the survey result which conducted to a group of database administrators, there is a need for the database administrators to have an automated tool to perform the role transition more efficient and error free. With an assessment of the Oracle Data Guard database, we have developed an approach for an efficient role transition of Oracle Data Guard database in term of the architecture concept and components, best practices and their impact on role transition. The approach consists of role transition, implemented with a custom developed tool, and prerequisite and post task of role transitions using the recommended Oracle best practices. The approach is designed by doing a comparative study of existing Oracle best practices and interviewing database administrators. The approach consists of prerequisite and post task of role transition such as verify configuration health check, ensure no redo log transmission gap between primary and standby database, overall health check of production database and its standby databases and other checks. Finally, an evaluation between the existing methods and the proposed approach one is provided based on how they solve the Oracle Data Guard role transitions gaps with considering the external

factors challenges and the degree of automation provided. The results of the study show the proposed a role transition approached, a set of SQL query scripts integrated in a custom designed Perl tool and demonstrated how the tool helped to automate the role transition process. It has significantly reduced the time taken to perform the role transition up to 80% with 99% successful role transition. Moreover, the Oracle Database Automated Role Transition (ODaRT) ease the routine task database administrator and helpful for the novice to perform error free role transition.

Keywords: Oracle Data Guard, switchover, role transition, automation

[ALAT AUTOMASI UNTUK PERALIHAN PERANAN DALAM PANGKALAN DATA ORACLE DATA GUARD DENGAN MENGGUNAKAN AMALAN-AMALAN TERBAIK ORACLE]

ABSTRAK

Ketersediaan pangkalan data adalah faktor kritikal bagi kebanyakan perniagaan untuk mengekalkan dan meneruskan perniagaan mereka. Setiap organisasi mempunyai ketersediaan pangkalan data tersendiri. Kegagalan peralihan peranan adalah salah satu punca untuk ketersedian pangkalan data. Perisian pangkalan data Oracle dengan versi Data Guard paling biasa digunakan untuk menjalankan aplikasi kritikal seperti pembuatan, e-perbankan, dan lain-lain. Alat automasi diperlukan untuk mengekal dan melaksanakan peralihan peranan yang cekap, optimum dan bebas ralat dalam pentadbiran Oracle Data Guard yang kompleks begini. Daripada hasil kajian yang dijalankan diantara sebuah kumpulan pentadbir pangkalan data, kita mendapati keperluan untuk mempunyai alat automatik untuk melaksanakan peralihan peranan yang lebih cekap dan bebas ralat. Kami telah membangunkan satu pendekatan untuk peralihan peranan yang cekap untuk pangkalan data Oracle Data Guard, dengan penilaian dari segi konsep, komponen seni bina, amalan terbaik dan impaknya terhadap peralihan peranan. Pendekatan ini terdiri daripada peralihan peranan yang dilaksanakan dengan alat yang dibangunkan khususnya, prasyarat dan tugas selepas peralihan peranan menggunakan amalan terbaik yang disyorkan oleh Oracle. Pendekatan ini direka dengan melakukan kajian perbandingan terhadap amalan terbaik Oracle yang sedia ada dan dengan menemuramah pentadbir pangkalan data. Pendekatan ini terdiri daripada prasyarat dan tugas selepas peralihan peranan seperti mengesahkan pemeriksaan kesihatan konfigurasi, memastikan tiada jurang penghantaran log semula antara pangkalan data primer dan siap sedia, kestabilan keseluruhan pangkalan data produksi dan pangkalan data siap sedia. Akhirnya, penilaian antara kaedah yang sedia ada dan pendekatan yang dicadangkan disediakan berdasarkan

bagaimana mereka menyelesaikan jurang peralihan peranan Oracle Data Guard dengan mempertimbangkan cabaran faktor luaran dan tahap automasi yang mereka sediakan. Hasil kajian menunjukkan cadangan pendekatan peralihan peranan dan satu set skrip SQL yang disepadukan dalam alat Perl yang direka khusus menunjukkan bagaimana alat itu membantu untuk mengautomasikan proses peralihan peranan yang dapat mengurangkan masa yang diambil untuk melaksanakan peralihan peranan hingga 80% dengan 99% peralihan peranan yang berjaya. Selain itu, Oracle Database Automated Role Transition (ODaRT) memudahkan rutin pentadbir pangkalan data dan juga berguna untuk orang baru untuk melakukan peralihan peralihan dengan bebas kesalahan.

Kata Kunci: Oracle Data Guard, beralih, peralihan peranan, automasi

ACKNOWLEDGEMENTS

I would like to this opportunity to thank the people who have guided, supported and motivated me to complete this dissertation. I will always gratitude for the sincere guidance and advice given by my supervisor Dr. Nazean Jomhari. There is no gift to substitute the knowledge I gained from her, thank you. I would like to say a big thank you to my whole family for being always supportive especially my brother Kesavan who was guided me throughout this dissertation process. I also like to express my thanks to my colleagues and friends for all the motivations.

Finally to the almighty god, sometimes I just look up, smile and say, "I know that was you, God! Thanks!"

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ANSI	:	American National Standards Institute
CPAN	:	Comprehensive Perl Archive Network
CRSCTL	:	Oracle Clusterware Control Utility
DBA	:	Database Administrator
DBI	:	Database Interface
DGMGRL	:	Data Guard Command-line Interface
GUI	:	Graphical User Interface
IDE	:	Integrated Development Environment
JDBC	:	Java Database Connectivity
MML	:	RMAN Media Management Layer
OCI	:	Oracle Call Interface
ODaRT	:	Oracle Database Automated Role Transition
OEM	:	Oracle Enterprise Manager
RAC	:	Real Application Cluster
RDBMS	:	Relational Database Management System
RMAN	:	Recovery Manager
RPC	:	Remote Procedure Call
SPSS	:	Statistical Package for the Social Science
SQL	:	Structured Query Language
SRVCTL	:	Server Control Utility
TCP/IP	:	Transmission Control Protocol/Internet Protocol
TNS	:	Transparent Network Substrate
UML		Unified Modeling Language

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

The Oracle Data Guard database role transition is done either by issuing SQL statements or Data Guard broker interface or through Oracle Enterprise Manager. The Oracle Data Guard database supports two transitions of roles, one of which is to switchover, as in this case the primary database change the production or primary role with one of its standby databases. The other transition is failover which changes a standby database to the primary role in response to a primary database failure automatically depending on the configured setting or a manual failover is initiated by the Database Administrator (DBA) in case of primary failure or for any other reasons which shows the current primary database is unhealthy.

The error or failure free switchover and failover require a recommended pre and post checks that are not embedded with Oracle role transition features, it still needs a DBA to perform those tasks. This research is to study the existing Oracle recommended best practices and reuse the open source Perl modules and produce an automated tool is developed to achieve an automated (need DBA to execute) smooth Oracle database role transitions with a very minimal DBA intervention.

The novice DBAs will enjoy the benefit of this automated tool since to execute the automated role transitions tool will not require an in-depth knowledge on Oracle Database. The tool significantly contribute to the industries which using Oracle Database in their production environments as it save management cost and the use of software reuse reduce

the development cost. Compliance with automated best practices can help mitigate potential performance and infrastructure issues that affects many companies.

1.1 Statement of Problems

The big scaled industries and Database Administrators often face below problems in handling a smooth Oracle Database Role Transitions.

- Role Transitions (Failover or Switchover) often failed/hung due to several technical reasons (long running operations, archived log apply lag, backup is running during the switchover, etc.). The failures are avoidable if the DBA perform the best practices recommended by Oracle by performing Pre-Switchover checks and Post-Switchover (Metalink, n.d.). Below are the documented failures and the troubleshooting documents have been documented and provided for the problems during role transitions (Charles, 2014).
    - Failure to Convert the Original Primary Database
    - Failure to Convert Target Physical Standby Database
    - Failure to Open New Primary Database
    - The broker switchover fails due to problems with redo transport services

- The trouble-free Data Guard Switchover or Failover requires an around 21 tasks involving pre/post checks and health checks which is time consuming. This would require the DBA to have handy commands and really a time-consuming task to perform all the checks and there are possibilities to miss or do a wrong check. The table 1.1 shows the minimal required checks that recommended by Oracle for a Switchover.

Table 1.1: The minimal required checks that recommended by Oracle for a Switchover

Check	Pre/Post Checks
1	Verify disk groups free space
2	Verify observer location
3	Verify configuration health
4	Verify there is no apply delay for the target standby
5	Ensure online redo log files on the target physical standby have been cleared
6	Verify there are no large gaps
7	Verify primary and standby tempfiles match
8	Verify primary and standby disk location
9	Verify all datafiles are ONLINE
10	Verify primary and standby datafiles disk location
11	Verify primary and standby online redo log disk location
12	Suspend Scheduler jobs
13	Check for potential long running operation
14	Suspend backup jobs
15	Clear potential blocking parameters
16	Perform switchover
17	Resume scheduler jobs
18	Resume backup jobs

The failure of role transitions will impact the database uptime and database maintenance downtime. The unforeseen failure will impact the mission critical database/applications, which is not tolerable in any industries. All the above problems is avoidable with an Automatic Switchover or Failover Tool which will perform all the pre and post checks automatically without much human involvement. And the automate solution eases the Database Administrators on their daily operations duties. The development of the automation solution is costly; hence the use of reuse Perl modules is proposed.

1.2 Aim of Research

The aim of this research is to design and development of an automated approach for role transition in Oracle Data Guard Databases by incorporating Oracle and DBA expert's best practices. The automated tool development is aided by Perl modules for performing Oracle Data Guard Database's Failover or Switchover embedded with known best practices steps (prerequisites and post checks) recommended by Oracle and Database Administrators experts.

1.3 Objective of Research

To achieve the research goal, the following objectives are identified and evaluated in this study:

[1] The first objective is to investigate and determine the challenges in the existing Oracle Data Guard role transition.

[2] The second objective is to propose and design an automated approach for Oracle Data Guard role transition that addresses the above challenges and minimize database administrator intervention during the role transition process.

[3] The third objective is to develop an automated tool that can minimize the role transition downtime in Oracle Data Guard Databases. It should demonstrate and measure the efficiency of the proposed approach by the application of the developed tools in the selected test cases, expert survey and the comparison with other existing prominent methods.

1.4 Research Questions

Based on the research objective, a number of research questions are examined and the solution for the each questions will be answered and organized in each chapters of this research paper. The research questions are denoted as "RQ". The solutions are denoted as "S".

RQ1: Why does the existing Oracle Database Role Transition methods of SQL statements, Data Guard broker, and Oracle Enterprise Manager experiencing failures?

RQ2: What are the documented best practice steps in executing existing Oracle Database Role Transition methods through SQL statements, Data Guard broker, and Oracle Enterprise Manager?

RQ3: How important of performing the best practices before and after an Oracle Database Role Transition?

RQ4: How existing Oracle Database Role Transition methods and tools serve the purpose?

RQ5: What are existing automations tool exist to support Oracle Database Role Transition?

RQ6: How efficient the existing automations tool performing the Oracle Database Role Transition?

Figure 1.1 illustrate the research questions raised and solutions that applied throughout this research. Solutions are denoted with ‘S’ and research questions denoted with ‘RQ’.

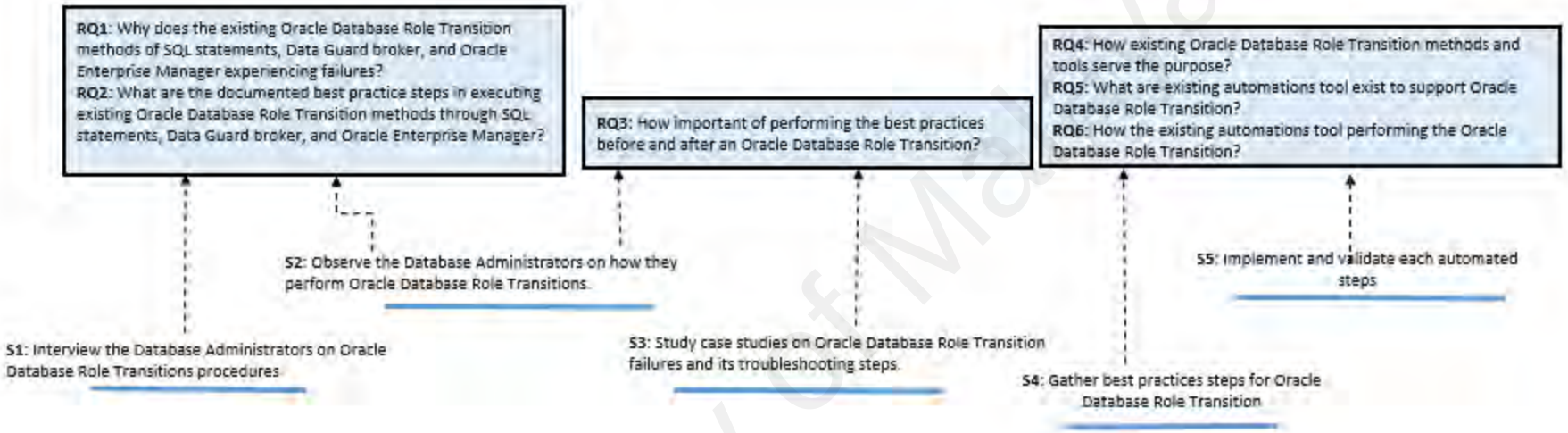

Figure1.1 Research Questions and applied solutions

1.5 Research Significance

With the current market analysis and needs of high availability of a mission critical databases, the importance of this research is aiming at:

- Reducing of role transition time and improving the efficiently.

  Having an automated tool to perform the role transition will significantly speed up the process of role transition and with less human error. The time for a human need to think and prepare for the role transition can be eliminated.

- Reducing the human errors

  Automated tool will help to reduce the human errors since no human is always perfect in performing a task without mistakes, but automation can help to reduce the ratio of failures. With the embedded automation solution in the current role transition will significantly help the database administrator to perform the role transition without mistakes and it can save them from any unexpected recovering situation.

- Reducing database downtime cost and maintenance cost

  For a mission critical-driven application, each single second of unavailability is counted for cost of the return of investment. A database failure is not acceptable during a planned role transition of switchover. The failures are avoidable by practicing and performing the role transition with guided and recommended steps. The automation would help to make sure the recommended best practices

are always compliance rather than a human. The role transition is usually performed during a planned downtime for database or operating systems maintenance such as patch or upgrade. By optimizing the existing database server for the automation will not engage any additional resource or financial cost.

1.6 Organization of Dissertation

There are six (6) main chapters discussed in this dissertation as follows:

- Chapter 1: Introduction

  This chapter consists of introduction to the problem, the objectives of the study that describe the problem and the solutions achieved through the research questions and solution applied.

- Chapter 2: Literature Review

  This chapter present a literature review on prior work on Oracle Data Guard Database Role Transitions. The reviews of various sources regarding the selection criteria and potential techniques are revealed in this chapter. The methods and automation of role transition is discussed heavily. The comparison and evaluation of Oracle Data Guard role transition feature shall be elaborate well on this chapter.

- Chapter 3: Research Methodology

  This chapter describes the development methodology of ODaRT tool. The client- server architecture and the database communication and manipulation

languages such as PL/SQL and DGMGRL is described in detail here. The role transition approach of switchover and its pre-check is elaborated in this chapter.

- Chapter 4: Design and Development of the Approach and Automated Tool

  In this chapter the development of ODaRT tool is discussed. The functional and non-functional of the tool is discussed in detail. The chapter also discussed the development of the tool through the use case modelling, analysis model, system design and details of the source code which integrated with the database administrator tasks.

- Chapter 5: Results and Discussion

  In this chapter we demonstrated the main features of the ODaRT tool and validated the defined testing strategy. Both the black and white testing model been used to find and solve the bugs in the software. Finally, the collected result is presented and evaluated by comparing with other prominent Oracle Data Guard role transitions methods.

- Chapter 6: Conclusion and Future Works

  The final chapter to discuss about the overall summary of the research strengths, weakness and future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The problems and failures in Oracle Database Role Transitions are often discussed in most available Oracle Support also known as Metalink, oracle forums, oracle supports community discussion board, Knowledge Centre and blogs. The failures of role transition would lead to downtime causing outage of data access, operations are down, revenue is lost, customer relationship is damaged and lawsuits. Although there are features provided and embedded by oracle such like Fast-Start Failover or tools like Oracle Enterprise Manager (OEM) and Data Guard command-line interface (DGMGRL) but it does not address the whole best practices that recommended by the Oracle itself and still need manual intervention by database administrator to execute. The reason why the best practices was not imbedded by Oracle in their OEM and DGMGRL features could be due to customers varied setup of the databases. For an example, backup is not taken in all standby databases, but it might require by certain organization to maximize their disaster recovery. Also, these are recommended practices but not a mandatory check to have a successful role transition.

The Oracle White Paper (*Oracle Active Data Guard*, 2010) and books of Role Transition Best Practices (Carpenter et al., 2009; "Oracle10g: Data Guard Switchover and Failover Best Practices,") proves, the Fast-Start Failover automatically executes a database failover within seconds of an outage being detected and can complete in seconds however, it has not provided the same automate solutions for planned switchover where a database administrator is needed to perform prerequisite and post checks in order to perform a smooth switchover (Jim, 2009). The fast start failover did detect the unavailability of production

database and immediately failover to the standby databases but still DBA is needed to complete the failover process. Studies shows executing the switchover without pre check and post check could lead to failure or hung of the databases ("High Availability Data Guard Broker," ; Tuomas, 2010).

The other explicit tools to support the automation of role transition are reviewed in the following sections. The Dbvisit Standby tools which help to rid of the complication of setting up and running Oracle standby databases. It does have some of the features of recommended Pre and Post checks for a graceful Switchover and Failover operations but it does not automated and still a DBA intervention is needed to run separately the processing script. The automate of data guard best practices (Larry, 2011; Nawaz & Soomro, 2013) has demonstrated how DBAs can automate setup, configuration, monitoring and maintenance of standby Data Guard environments with assistance from the Data Guard Toolkit which only assist to automation but with no clear solutions and limited to only Linux platform and using shell scripts.

The following sections of this literature review cover the topics of architecture, requirements and external factors of role transition, methods of role transition, existing automation on role transitions, and failures of role transitions, available best practices and its comparison and finally the evaluation of role transition in their newer version.

2.2 Oracle Data Guard Concept

The Oracle Data Guard is a feature that only comes with Oracle Database Enterprise Edition and operates the database in two separate roles consisting of primary database or standby

database. The databases in Data Guard is configured using Oracle Net for the networking between the primary and standby databases. The databases can be located to different data center irrespective of their location. The Data Guard configuration enables one of the database to function as a primary role and the other one or additional databases as standby role (Burleson, 2015; "Partitioning Concepts. oracle.com," 2015). Both the primary and standby databases is manageable through three available methods. The three methods are, the basic SQL command-line interface, the powerful and distributed management of Data Guard broker interface including the DGMGRL command-line interface and lastly the user friendly graphical user interface which is integrated with its Oracle product of Enterprise Manager (Crosby, Hughes, Lizieri, & Oughton, 2005; Kumar, 2003; Metalink, n.d.)

The redo data is the vital structure for any of an Oracle instance and it is the fundamental data used to synchronize and real time update the standby databases. In Oracle there is a service called Redo Transport Service to manage the redo data sending to the physical databases. The Redo Transport Service is an automated transfer to transmit the data to standby databases from primary database via the configured parameter in both the databases. The *LOG_ARCHIVE_DEST_n* is the parameter used to configure the destination of the log transportation. The service also acts as automatic gap resolver whenever there is found redo log gap between primary and standby databases (S. Alapati, Kuhn, & Nanda, 2007).

The gap of archived redo log usually happens when there is network glitch between the primary and standby databases. While redo transport services support the transmission of redo log to standby databases, the Log Apply Services support apply of the received redo logs to the standby databases (S. Alapati et al., 2007; "Oracle® Database VLDB and Partitioning Guide 11g Release 2 (11.2). oracle.com," 2015). The Log Apply Services also

automatically applies the redo data to uphold the sync between the primary and standby databases.

There are three data guard protection modes available for customers to choose to depend on their business needs. Some customer give importance to not loss any data while some customer give importance for the availability of the database. To cater different business needs Oracle has provided three protection modes called Maximum Protection, Maximum Availability and Maximum Performance. Three of those have their own advantages and limitations due to the requirement.

For an example, Maximum Protection mode ensure there is no data loss, but it cause each of the transaction in primary will only be committed after the redo log data is transmitted to at least one of the standby databases and it causes the production database to not able to give its best performance (Greenwald, Stackowiak, & Stern, 2013).

The Maximum Availability mode enabled databases will give the highest level of database availability in which the primary node will shut down itself when there was problem while redo stream writing in standby database, but it will happen to the primary database which is running in Maximum Protection mode (Kuhn, Alapati, & Padfield, 2016; Tuomas, 2010). While the last Maximum Performance mode enabled database will provide the highest performance of its production database.

This is possible by the asynchronous of redo data transmission to the standby database, the primary database will commit as soon as the redo log is transmitted to at least one standby database and the primary will not wait for the acknowledgment of the redo log apply in standby database. The selection between the data guard protection modes depend on the user business needs.

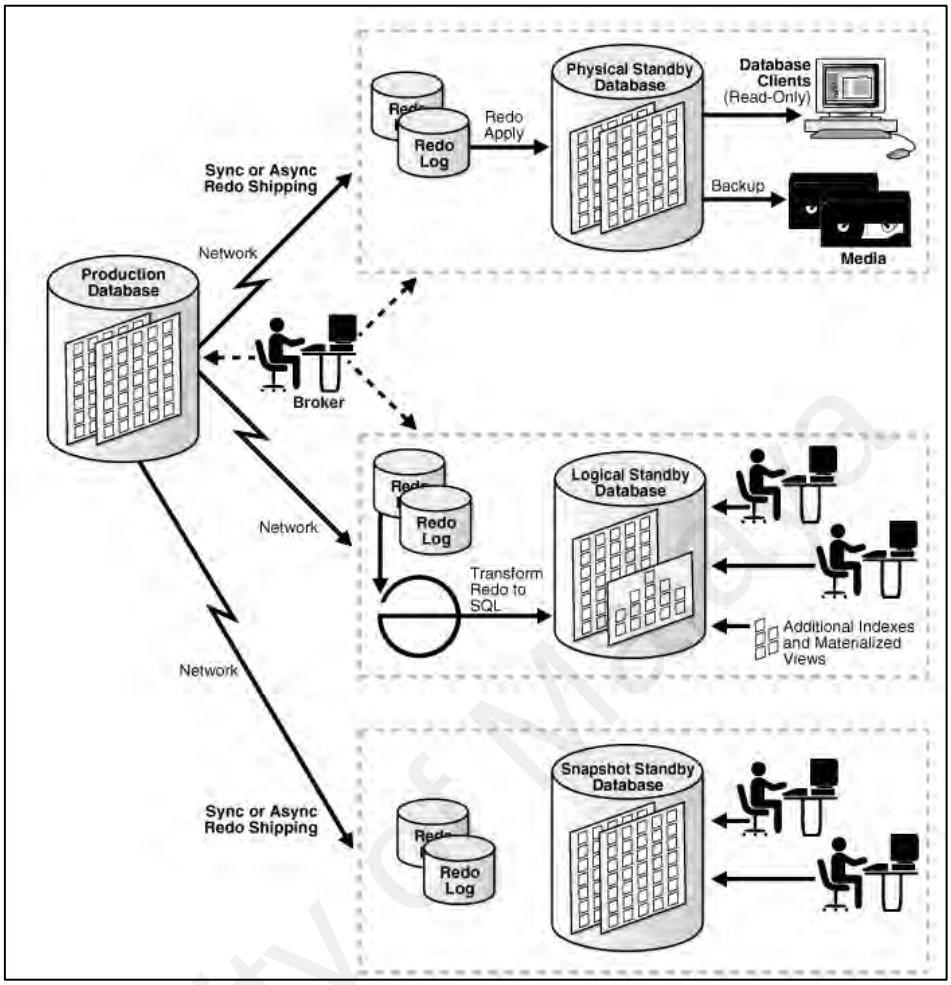

Figure 2.1 Oracle Database with Data Guard Architecture on Primary and Multiple Standby (source: Oracle Database Documentation 11g Release 1 (11.1)(Ashdown et al., 2011; Bryla, 2007))

Figure 2.1 shows the architecture of Oracle Database with Data Guard Broker. The features of Data Guard which is to communicate between the production database and other standby databases either it is a Physical Standby Database, Logical Standby Database or Snapshot Standby Database ("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ," ; "11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.," ; Carpenter, 2009). The choice of the standby database depends on the user business and capabilities needs. All the standby databases are real time copy of production database

which is periodically updated by transporting the redo data via network. The most common data guard configuration is physical standby where it is an identical copy of block-by-block production database for a use of disaster recovery and opened as read-only database for support the reporting applications ("11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.," ; Hayes, Rinkevich, & Lowrey, 2007).

The Logical Standby Database is not identical in terms of its physical structure (data file location) but it is identical for logical information of a production database. A sync with primary database is by transforming the rodo logs into SQL statements and execute in standby database using LogMiner features. The Snapshot Standby Database is usually created for a purpose of testing and it is fully updatable ("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ,"). The updates performed on standby database will be discarded upon its converted to physical standby to sync as production database.

2.3 Oracle Data Guard Role Transition

2.3.1 Switchover

Data guards provides service to change the roles of the database by switching it for any planned outages. Planned downtime usually happens for a regular maintenance of the infrastructure such as hardware or software maintenance and repair, upgrades and patching. A database switchover is also performed to resolve of any data corruption or failures when the production or primary database is open.

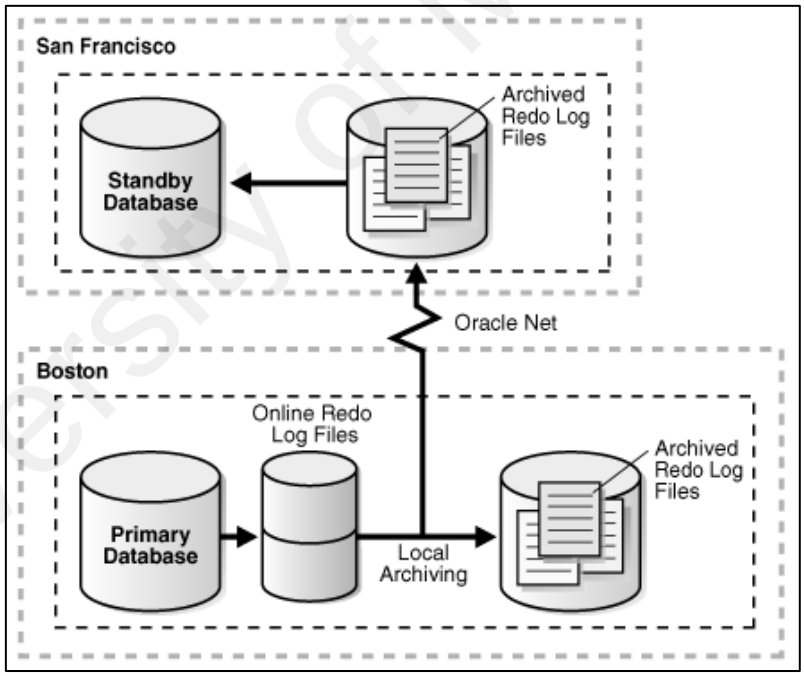

Figure 2.2 Data Guard Environment after Switchover (source: Oracle Database Documentation 11g Release 1 (11.1)(Ashdown et al., 2011; Bryla, 2007))

Figure 2.2 shows the environment of the production and its redundant database. The shipment of Oracle archive logs is reversed once the switchover is done. The old primary San Francisco database now become standby database after the switchover and receives the archived redo log files from new primary database of Boston (Bryla, 2007; Charles, 2014; Chaudhuri & Weikum, 2000). And the Bostan database is now serves as new production database and it is transmits the redo data to the new standby database San Francisco. The switchover is always not 100% success as it also could failed if there is an issue among the primary and standby database. The failures are always prevented by executing the recommended best practices which involves some pre and post checks. The best practices will be discussed more detailed in the following sub topics.

2.3.2 Failover

A Failover is triggered when the primary database is having an issue which it is not reachable, and it will move to one of the standby databases. This situation usually happens when the primary database is not recoverable for a specific time. The failover may happen automatically, or it can be triggered manually when there is a need. In below subtopic it is discussed in detail the need of failover and its criteria to trigger an automatic or manual failover. Failover may or may not result in data loss depending on the protection mode in effect at the time of the failover (*Fast-Start Failover Best Practices*, 2010; "Oracle10g: Data Guard Switchover and Failover Best Practices," ; Singh, 2013). An administrator initiates manual failover when the primary database fails. In contrast, Data Guard automatically initiates a fast-start failover without human intervention after the primary database has been unavailable for a set period (the fast-start failover threshold).

2.3.1.1 Manual Failover

The manual failover is a process of convert a standby database to a primary database when the original primary database fails and there is no possibility of recovering the primary database in a timely manner. There may or may not be data loss depending upon whether the primary and target standby databases were transactional consistent at the time of the primary database failure. The manual failovers performed in three different ways (Jim, 2009):

- Complete and immediate manual failovers

  Using Enterprise Manager or DGMGRL, it can perform either a complete (recommended) or an immediate failover. It automatically recovers the maximum amount of data for the protection mode of the original primary database application data.

- Performing a manual failover operation

  After determining that there is no possibility of recovering the primary database in a timely manner, ensure that the primary database is shut down and then begin the failover operation. The manual failover initiated through using Oracle Enterprise Manager or DGMGRL.

- Re-enabling disabled databases After Failover or Switchover

  To restore the original disaster-recovery solution after switchover to a logical standby database or after failover to any standby database, requires to perform additional steps. Any database that was disabled through multiple role changes cannot be reinstated. The database must be recreated manually and then re-enable the database in the broker configuration.

2.3.1.2 Fast-Start Failover

Fast-start failover allows the broker to automatically fail over to a previously chosen, synchronized standby database in the event of loss of the primary database. Fast-start failover quickly and reliably fails over the target standby database to the primary database role, without requiring performing any manual steps to invoke the failover (*Fast-Start Failover Best Practices*, 2010; "Oracle10g: Data Guard Switchover and Failover Best Practices,"). Fast-start failover can be used only in a broker configuration and can be configured only through DGMGRL or Enterprise Manager (Crosby et al., 2005; "Oracle 9i Database Manageability,")

The observer is a separate OCI client-side component that runs on a different computer from the primary and standby databases and monitors the availability of the primary database. Once the observer is enabled, no further user interaction is required. If both the observer and the standby database lose connectivity to the primary database, the observer waits for the amount of time specified by the *FastStartFailoverThreshold* property before initiating a fast-start failover (*Fast-Start Failover Best Practices*, 2010; "Oracle10g: Data Guard Switchover and Failover Best Practices,"). Moreover, after the failover completes, the former primary database is automatically reinstated as a standby database in the new broker configuration when a connection to it is reestablished.

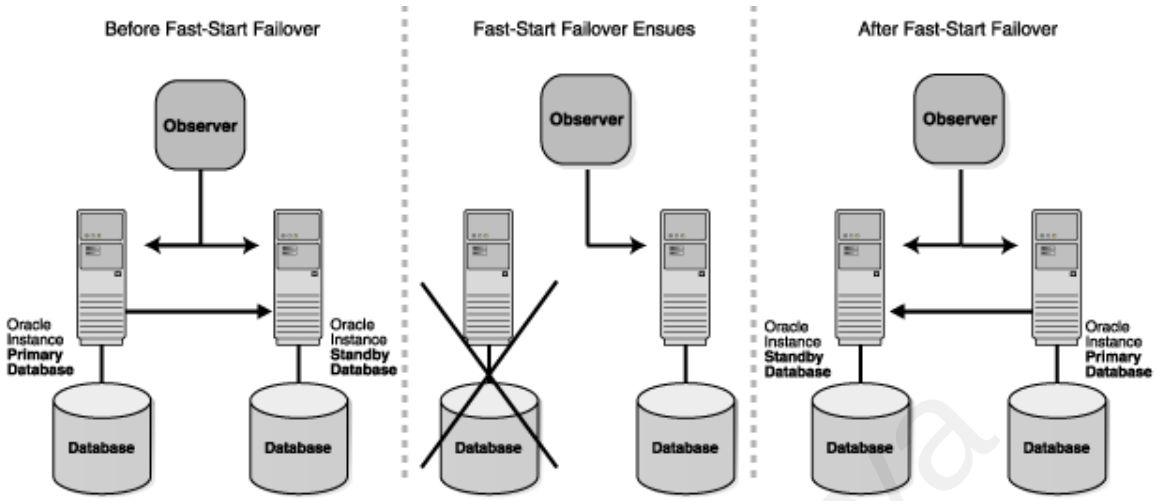

Figure 2.3 Relationship of Primary and Standby Databases and the Observer (source: Oracle Database Documentation 10g Release 2 (10.2)(Kumar, 2003; "Oracle10g: Data Guard Switchover and Failover Best Practices,"))

Figure 2.3 shows the relationships between the primary database, target standby database, and the observer during fast-start failover. After Fast-Start Failover the old primary database is reinstated as new standby database.

2.4 Oracle Data Guard Role Transition Failures

The role transitions are not always a successful task, the database administrator sometime faced errors and failures that need troubleshooting and fix. Oracle has constantly fixed the issues or bugs escalated to their support. In every version of its release or patches, Oracle have provided the permanent solutions. However, some failures are not avoidable if the database administrator did not follow the correct steps in performing a role transition (Larry, 2011; "Oracle10g: Data Guard Switchover and Failover Best Practices," ; *Oracle Active Data Guard*, 2010).

Oracle has always recommended using its best practices guideline to ensure a smooth role transitions. The best practices guidelines will be discussed in more detail in following topics.

Table 2.1 Type of Failures and Resolutions

Failures	Metalink Doc Reference	Resolution
Standby controlfile has corrupt information. Especially v$log_history was showing stale information	Switchover To Physical Standby without broker hangs. (Doc ID 1261877.1("Oracle-Base," 2013))	Ensure all the pre-requisites for switchover are met. Refer Doc ID 751600.1 10.2 Data Guard Physical Standby Switchover ("11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.,")
Problems Switching Over to a Physical Standby Database - Switchover Fails Because Redo Data Was Not Transmitted - Switchover Fails Because SQL Sessions Are Still Active - Switchover Fails with the *ORA-01102* Error - Redo Data Is Not Applied After Switchover - Roll Back After Unsuccessful Switchover and Start Over	(Carpenter, 2009; Charles, 2014) Oracle Data Guard Concepts and Administration, 11g Release 2 (11.2)	Ensure the best practices tasks taken before/after switchover
User gets *ORA-16778* from broker's switchover command when primary's *RedoRoutes* has an *ALT* attribute	Role transition fails with terminal standby as alternate of far sync (*ORA-16778*) (Doc ID 19399918.8 ("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ,"))	Ensure the primary's *RedoRoutes* has no alternate destination setting.

Redo transport service was unable to send redo data to one or more standby databases	*ORA-16778* 'redo transport error for one or more databases' (Doc ID 173073.1 (S. R. Alapati, Kuhn, & Padfield, 2011; Carpenter, 2009))	
12.1 Version Bugs - long open of standby after switchover - flashback adding 7+ seconds to failover - switchover failed with *ORA-00312* - broker adding time to switchover - broker validate issues - cancel recovery on standby is taking 3-5 seconds during switchover	Known issues related to Switchover/Role transitions (Doc ID 1901194.1 ("High Availability Data Guard Broker," ; Kumar, 2003; "Oracle 9i Database Manageability,"))	Apply related patches

Table 2.1 shows the errors or failures happened during a role transition. The failures and solutions are captured in Oracle support also known as Metalink (Metalink, n.d.).

2.5 Oracle Data Guard Role Transition Performance

Role Transitions have become more streamlined and efficient in recent versions of the Oracle Database. Using the best practices recommended by Oracle will help us to achieve the best possible role transition times. Analyzing where the time is being spent during a role transition can help tune and set proper expectations for the production environment(Charles, 2014; Nawaz & Soomro, 2013).

Table 2.2 Role Transition Timings *(Source: Oracle Maximum Availability Architecture, OWP 2016 (Kuhn et al., 2016; Kyte, 2010))*

Database Releases	Failover Time	Switchover Time
10.2 RAC Database/ Data Guard	60 secs	65 secs
11.2 RAC Database/ Active Data Guard	44 secs	54 secs
12.1.0.1 RAC Database/ Active Data Guard	27 secs	41 secs
12.1.0.2 RAC Database/ Active Data Guard	16 secs	34 secs

Table 2.2 shows the improvement of role transition timing throughout the Oracle versions. The data provided taken from the point the transition starts to the point the new primary accept the new client connection. Also, the baseline measurement taken during periods with minimum workload during Data Guard failover and no workload during Data Guard switchover operations("11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.," ; "Oracle10g: Data Guard Switchover and Failover Best Practices,"). There are also some external factors contributes to the role transition timings such as RAC or single instance databases, open or mounted standbys and varying workloads on a primary database or an Active Data Guard standby.

Table 2.3 Average Failover Time *(Source: Oracle Data Guard 10g Release 2 Switchover and Failover Best Practices, OWP 2016 ("Oracle10g: Data Guard Switchover and Failover Best Practices,"))*

	Manual Failover		Fast-Start Failover
	SQL*Plus Statements	DGMGRL or Enterprise Manager	
Physical Standby	17 seconds	18 seconds	17 seconds
Logical Standby	10 seconds	12 seconds	14 seconds

Table 2.4 Average Switchover Time *(Source: Oracle Data Guard 10g Release 2 Switchover and Failover Best Practices, OWP 2016 ("Oracle10g: Data Guard Switchover and Failover Best Practices,"))*

	Switchover using SQL*Plus	Switchover using DGMGRL or Enterprise Manager
Physical Standby	0:52	2:49
Logical Standby	0:50	1:48

Tables 2.3 and 2.4 shows the average role transitions of failover and switchover timings. The timing were achieved using the optimal failover and switchover methods which will be described more in next topic of role transition best practices. A role transition performed using Enterprise Manager takes longer because of the sequence in which the instances were restarted during the switchover and because the new production database was restarted. In addition, Data Guard Broker processing time contributed to the overall switchover time.

2.6 Oracle Data Guard Role Transition Best Practices

A business needs high-availability best practices that involve both technical and operational aspects of its IT systems and business processes. Such a set of best practices removes the complexity of designing a high-availability architecture, maximizes availability while using minimum system resources, reduces the implementation and maintenance costs of the high-availability systems in place, and makes it easy to duplicate the high-availability architecture in other areas of the business ("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ," ; "11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.," ; "Oracle10g: Data Guard Switchover and Failover Best Practices,").

One of the best ways to reduce downtime is incorporating operational best practices. It can often prevent problems and downtime before they occur by rigorously testing changes in test environment, following stringent change control policies to guard the primary database from harm, and having a well-validated repair strategy for each outage type.

2.6.1 Switchover Best Practices

Table 2.5 shows the total of 20 tasks with subtasks need to be executed by a database administrator to complete a switchover. The tasks are strongly recommended and mandated to execute to perform a switchover. These tasks are not automated and have to perform by a database administrator ("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ," ; "11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.," ; "Oracle10g: Data Guard Switchover and Failover Best Practices,"). For an environment where the high availability and running mission critical application, the

traditional method of using SQL statements is not preferred and usually the DBA will prefer broker method. Switchover using SQL statements will take time as its needs DBA to complete all the recommended tasks.

Table 2.5 DBA Tasks for Switchover *(Source: Data Guard Physical Standby Switchover (Doc ID 751600.1)("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ,"))*

Number	DBA Tasks
	Pre-Switchover Checks
1	Verify Configuration Health
	With Broker
	a. Verify Data Guard Environment Health
	b. Cancel apply delay for the target standby using CLI or GUI
	Without Broker
	a. Verify Managed Recovery is Running (non-broker) on the standby
	b. Cancel apply delay for the target standby using SQL
2	Ensure Online Redo Log Files on the Target Physical Standby have been cleared
3	Check for Previously Disabled Redo Threads
4	Check if the standby has ever been open read-only
5	Verify there are no large GAPS.
6	Use “THROUGH ALL SWITCHOVER” on Bystander Standbys
7	Verify Primary and Standby TEMP Files Match
8	Verify that there is no issue with V$LOG_HISTORY on the Standby
9	Verify no old partial Standby Redo Logs on the Standby
	Switchover
10	Clear Potential Blocking Parameters & Jobs
11	Shutdown all mid-tiers (optional)
12	Monitor Switchover
	With Broker
	a. Turn on Data Guard tracing on primary and standby
	b. Tail Broker Logs (optional) on all instances
	Without Broker
	a. Turn on Data Guard tracing on primary and standby
	b. Tail Primary and Standby alert logs on all instances
13	Create Guaranteed Restore Points (optional)

14	Switchover
	With Broker
	a. Data Guard Broker command line utility
	b. EM switchover
	Without Broker
	a. Verify that the primary database can be switched to the standby role
	b. If RAC, then shutdown all secondary primary instances
	c. Switchover the primary to a standby database
	d. Verify the standby has received the end-of-redo (EOR) log(s)
	e. If the standby is a RAC configuration, then shutdown all secondary standby instances
	f. Verify that the standby database can be switched to the primary role
	g. Check if the standby has ever been open read-only
	h. Switchover the standby database to a primary
	i. Open the new primary database:
	j. Correct any tempfile mismatch
	k. Restart the new standby
15	Contingency or Fallback
	Post-Switchover Steps
16	Set Trace to Prior Value
17	Reset Jobs
18	Schedule and conduct the incremental backup, roll-forward, and tape backups
19	Reset apply delay for the target standby
20	Drop any Switchover Guaranteed Restore Points

There are two options can be used for performing switchover with broker enabled which are using Oracle Enterprise Manager (OEM) or Data Guard Broker command line utility (DGMGRL) ("11.2 Data Guard Physical Standby Switchover Best Practices using the Broker.,"). The most preferred method for database administrators are using DGMGRL since the OEM is graphical user interfaced which needs to execute by clicking few pages before doing the final switchover. Also, the OEM is connected via client connection to the central database called OMS and could influence the network or traffic congestion which may cause

slowness in executing the switchover. The DGMGRL command would be the best method since the execution happens on the target servers.

Table 2.6 Verify there are no large gaps

Primary Database:

SQL> SELECT THREAD#, SEQUENCE# FROM V$THREAD;

Standby Database:

SQL> SELECT THREAD#, MAX(SEQUENCE#)

FROM V$ARCHIVED_LOG

WHERE APPLIED = 'YES'

AND RESETLOGS_CHANGE# = (SELECT RESETLOGS_CHANGE#

FROM V$DATABASE_INCARNATION

WHERE STATUS = 'CURRENT')

GROUP BY THREAD#;

Table 2.6 is the SQL statement need to be executed by database administrators before switchover to verify there is no gaps of redo logs transmission between primary and standby databases. Imagine the time it takes for a DBA to completely perform the above 20 tasks for a smooth and error free switchover. Those are recommended only not mandated to perform for a switchover.

```
Welcome to DGMGRL, type "help" for information.
DGMGRL> connect /
Connected.
DGMGRL> show configuration

Configuration - db11g1tst.acme.com

  Protection Mode: MaxPerformance
  Databases:
    db11g1tst - Primary database
    db11g2tst - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL>
```

Figure 2.4 'Show Configuration' using DGMGRL

The figure 2.4 shows one of the command of verification of the primary and standby databases using the data guard broker command line (DGMGRL).

Figure 2.5 Process of switchover using OEM

Figure 2.5 shows the progress of switchover by the execution from GUI based Oracle Enterprise Manager (OEM)

2.7 Evolution of Oracle Data Guard Features Specific to Role Transitions

Oracle has periodically introduced new features specific to role transitions in all its version. Each versions have introduced a better capability and features to support a robust role transitions.

Table 2.7 Evolution of Oracle Data Guard features specific to Role Transitions

Oracle 8i	Oracle 9i	Oracle 10g	Oracle 11g	Oracle 12c
Year 1998-2000	Year 2001-2002	Year 2004 - 2005	Year 2011-2015	Year 2017

* Introduction of Oracle Data Guard concept with creation of a standby database for disaster recovery	* Data Guard Broker feature is introduced with Data Guard Manager GUI to manage, monitor and automate the configuration * The role transition of Switchover and Failover operations is introduced * Data guards detect the gaps of archive log and synch automatically to standby * 3 protection modes of maximum protection, maximum availability and maximum performance is introduced for data protection	* Real-time apply is introduced to fasten the process of switchover or failover * Rolling upgrades by use of logical database switchover *Fast-start failover feature where it automate the process of failover during production database outage	* Active Data Guard feature is introduced to enable a physical standby database is quarried while it's actively synching with primary database * Redo Logs compression is introduced to fasten the gap resolving * Supports heterogonous operating systems of Standby Database	* Far sync feature is enable to transmit redo from a standby database to other standby databases regardless of its distance * Global Data Service (GDS) is in like a RAC service which provides failover and load balancing * Verify and Validate commands introduced to ensure the system is healthy before switchover

Table 2.7, shows the evolution of the role transitions in its entire Oracle version since it was introduced on 1998 the first 8i. From the table its clearly shows Oracle had done significant milestones for its products advancement and provided its best up to the to the market trend. Oracle have announced the new version of Oracle Database called 18c on October 2017 at Oracle Open World 2017 but it yet to publish the product in details. It promised and promoted as autonomous database, expecting most the DBA tasks to be automated. The validate and

verify commands feature which introduced in Oracle 12c version is almost close to the subject being reviewed in this paper.

2.8 Summary

This chapter reviews the Oracle Data Guard concept and architecture in the beginning and followed by a detailed study of the each Oracle Data Guard Role Transition which are Switchover and Failover. The architecture, requirements, failures and performance of the role transitions is discussed in detail. The methods of role transitions are done either by using SQL statements, Data Guard Manager Interface or Oracle Enterprise Manager and these methods have been used by third-party companies to produce automated tools called Dbvisit Standby and DG toolkit.

The role transition failures and performance data are reviewed in this chapter. Based on the literature review, it's concluded that the currently available tools, features, technology and reference are Oracle best practice documents, fast-start failover technology which is to prevent downtime and data loss. The third party tool of Dbvisit Standby is providing prerequisites and post processing script with database administrator intervention. The DG toolkit is only featured in Linux platform.

The research studies revealed that the consumers have different set of configurations and uses in performing Oracle Data Guard role transitions. The expectations which were derived from this chapter are, a tool with reduced manual process and robust role transition in multi hosted platform.

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Introduction

This chapter presents the research methodology process involved and the produces carry out on this study. The study was conducted with the experts of database field. The respondents were the database administrators who working in multinational private sectored company. A set of interview questions was designed to investigate and to study the respondent's opinion based on experience and point of view on approach that can help them to perform database role transitions. The researcher also provides set of questionnaires for the respondents on the same view and had conducted interview sessions with the database administrators. These chapters are presented based on the following sections: (1) method of analysis and data collections, (2) quantitative research method and qualitative research methods.

3.2 Method of Analysis and Data Collections

A research methodology defines which methods and research instruments are currently used and how the results of the research studies are carried out, measured and analyzed. Two method of research was conducted which are quantitative and qualitative research methodology. Questionnaire is one of the types of research instruments that were used under quantitative method. The questionnaire was designed with two (2) parts of research questions which

were related to Oracle Data Guard and role transition activity to get respondents feedback. Apart from this the qualitative research methodology is focused based on the interview and comparative research instruments. The interview is conducted with database administrators and experts of the filed. This method is used to determine about what is the perception of respondents on this role transaction activity if used automation on performing their task. This was reviewed at the end of edge of interview sessions. In additionally, the comparative research method also was used to investigate the issues and challenges faced by respondents and determine whether they willing to accept this automated tool as learning tools for their Oracle Data Guard role transition process.

These both quantitative and qualitative research methods are conducted on the chosen respondent's' which generally associated with this research studies of Oracle recommended best practices. The figure 3.1 shows the process involved involving the data collection and analysis. The points below elaborate more on the process involved.

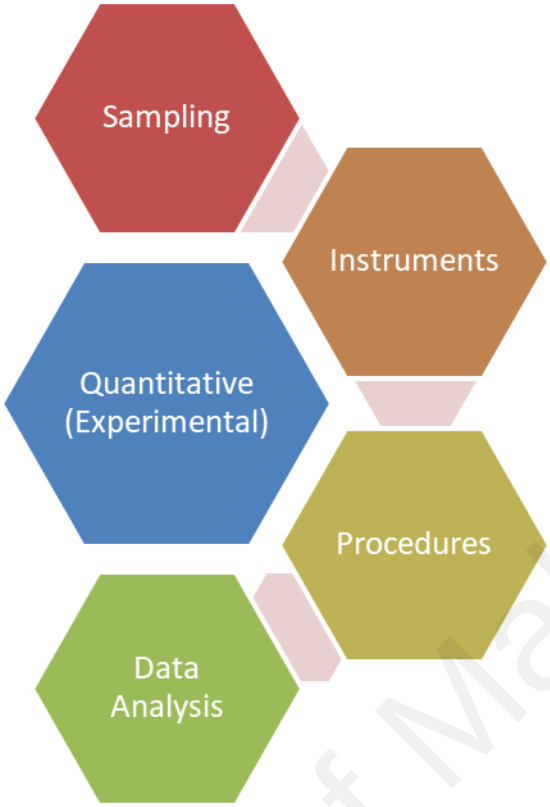

Figure 3.1 Method of analysis and data collection process (Palinkas et al., 2015; Patton, 1990; Wynekoop, 1991)

- Sampling

A group of database administrator from a private manufacturing industry who are knowledgeable in Oracle Data Guard Database administration is identified and taken as sample to conduct the survey and test cases. The feedback and behavior change of participants/respondents is collected through the identified instruments. The data gathered will be use to study the objective of this research.

- Instruments

The data collected from the participants is very crucial to justify the deliverables of this research study. The participants will be provided with set of test case scenarios to perform.

The test case scenarios are identified after aligning with the literature reviews of Oracle best known practice and objective of this research. The respondents will be delivered with a set of questionnaires (refer to Appendix A) to answer at the end of the session. The questions are mainly to answer about the result of the test case conducted scenarios and feasibility study of the ODaRT tool.

- Procedures

The responded given a set of instruction to perform the role transition in integration environment using the ODaRT tool and the traditional method of using data guard broker.

- Data Analysis

For the data analysis we selected the Excel Spreadsheet and SPSS (Statistical Package for the Social Sciences) tool. The gathered quantitative data is mined, measured and reported using the SPSS tool. The raw data which captured in excel format spreadsheet is exported to the SPSS tool and analyzed using the probabilities analysis. The analysis is based on below criteria:

  - Time taken to perform the role transition – the time taken to perform database switchover or failover including the pre and post tasks.
  - Ratio of successful – the proportion of successful role transition.
  - Switchover time – the time taken to perform the switchover should less than 2 minutes.
  - Failure of role transition execution

The results of the collected data are analyzed and presented in graphical form with explanations in the result chapter.

3.3 Quantitative Research Method

Quantitative research method it's conducted to measure the user's perceptions on the tool and what are the entities or variable that the tool should have to define the tool and how it's can be satisfied the user needs. This quantitative method is measured based on the statistical analyzing method. As earlier discussed, the questionnaire of research instrument is used to collect the data, and to analysis the outcome of research.

3.3.1 Questionnaire Designs

Analysis of data is a process of investigation, measuring, and validating the gathered information into a useful of report which used to make decision and to bring a conclusion of a research hypothesis. This began with a multiple way of measuring techniques and methods. The data which was gathered from the respondents/participants are arranged and organized into an excel format spreadsheet and exported to a pivot table. An analytical study was performed on the pivot table using the data mining process on each of the data analysis stage. At the questionnaires design, all quantitative questions were scored on a 5 point like scale indications where 1= not true, 2= somewhat true, 3=mostly true, 4= true, 5= very true. The mean scored were also ranked from the low level (close to 1= not true) to higher scores levels (close to 5= very true) to investigate the perception of the reliability data. These studies also pointed out and calculated that the data based on the mean, percentages, and frequency. The questionnaire set was designed with two parts which inclusive of part 1 (Method and Degree of Automation in Role Transition) and part II (Ratio of Successful). Part I consist of 5 questions to gain information method used to perform role and role transition and to identify

the degree of an automation used to perform the role transition. In part II, a set of 5 questions were asked to gather information on role transitions failures and success. The person answering the questionnaires are database administrators who are the expert on the field and the person who perform the role transition in their organization routinely.

3.3.2 Data Collections

Respondents from database backgrounds were selected for this case study because the study is focused on Oracle Data Guard role transition. This questionnaire (also called survey) was a set of questions was given to a sample of database administrators from a multinational private company. The main objectives of this data collection process are to make understand about the participants/respondents background and perception of Oracle Data Guard role transition. As earlier mentioned in the previous section, the participants feedback (answer) are will complied and organized to perform analysis studies. Therefore, this was clearly shown that the questionnaire is right research instrument to collect and validate the data in effective way to approached and to validate the objective of this research study. These enable to find out the opinion from the respondents regarding the view on Oracle Data Guard role transition via automation. The result of this data collection is analyzed and presented in graphical form with elaborations of it in the following section.

3.3.3 Result

This section discussed the result of the analysis from the questionnaire survey question. Based on the Figure 3.2 shows the respondents on this study are 70% are senior database administrator who are the experts having more than 5 years of experience in the field of database.

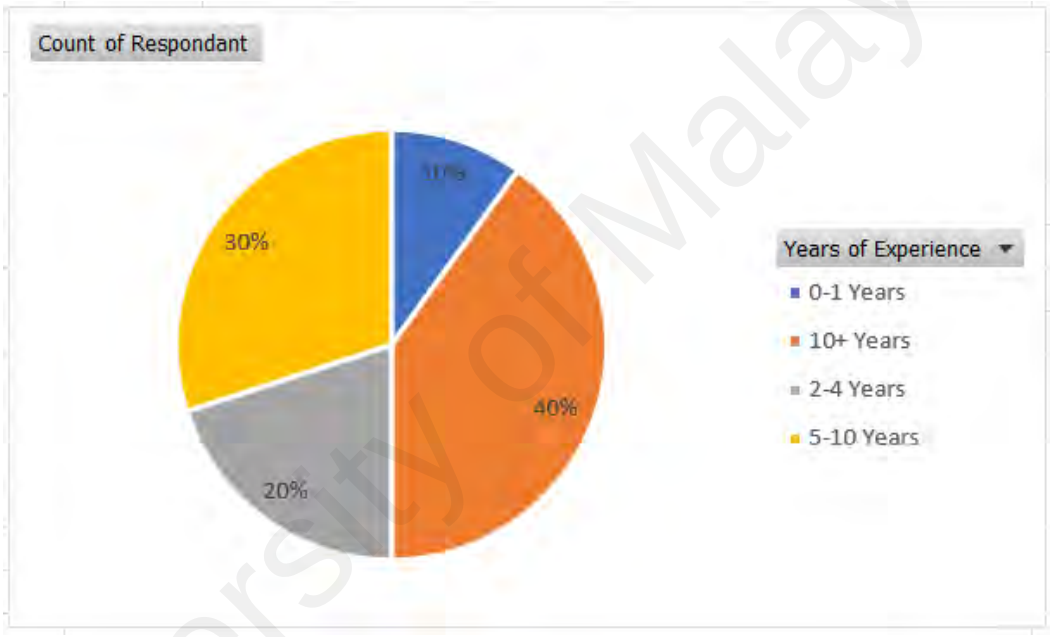

Figure 3.2 Pie Chart of Respondent's Years of Experience

The researcher has more confident on conducting the research as the data provided by the more senior database administrators would be more accurate and reliable considering their technical expertise and years of experience in the field of database.

3.4 Qualitative Research Method

Qualitative research is a method which has ability to precise and provides on particular group of individual perception towards a research question. Apart of used quantitative method, this qualitative method assists a researcher to be having better understand about the objective and issues on the research. Moreover, at this research studies, the techniques of interview and comparative case study instruments are used to analysis how the automated tool can be facilitated the database administrators.

3.4.1 Interview

The main objective of this interview conducted is to get the perception and opinion of the participants. Since this was open-minded questions involved, the prepared questions are can be far more to personal form of question when compared to questionnaire method of study. Thus, its important factor that the questions which pointing to participants should be relevant to the research hypothesis. This research is focused to a group of database administrators.

3.4.1.1 Interview Session Procedure (Database Administrators)

Seven (7) database administrators from a multinational compare are chosen for the interview session. The participants are chosen based on their experience in the filed of database administration specifically to Oracle Data Guard features. The DBAs were asked on how they perform and handle role transition process. They were asked on the degree of automation uses, or Oracle best practices uses in their daily routine. The researcher also has interviewed on the matter of challenges faced in performing the role transition and maintaining the health

of Oracle Data Guard databases. Here we have listed few samples of quotes from the database administrators after the interview section on Oracle Data Guard role transitions.

"*It is really hard for me to remember the SQL statements to perform before the switchover activity*"

"*We often failed to bring the database online in designated time during the switchover or failover activity due to the issue of human error*"

"*I usually keep a text file containing all the script needed to perform the pre or post check for a role transition*"

"*In my organization we use DGMGRL to perform the switchover or failover activity, it is really easing our administration"*

"*Yes, I would feel convenient if I can use a tool to perform the pre-tasks before a role transition*"

3.5 Summary

The overall findings of this study have proven that Oracle Data Guard role transition needs an automation and it is agreed and the purpose of the development of this tool is a good move for database administrator. Lesson learned from this survey is that role transition using Oracle best practices is suitable for the respondents as they were faced challenge to provide a perfect successful role transition all the time.

The chapter covered the components of client- server architecture and the implementation of the architecture in developing a Data Guard role transition tool. The approach is defined and guided by a well-constructed methodology. With incorporating the Oracle recommended practices into the proposed transition approach the database administrators are enabled to perform an efficient role transition. The approach is derived into three (3) functions called HealthCheck, PreCheck and the actual role transition, Switchover.

The chapter also identifies the methodology and stages of ODaRT tool engineering process consist of Oracle best known practices. The check list of the three identified modules are constructed based on the Oracle best known practices. The methodology of data collection and analysis is defined to develop and deliver the ODaRT tool.

CHAPTER 4

DESIGN AND DEVELOPMENT OF THE APPROACH AND AUTOMATED TOOL

4.1 Introduction

In this chapter we have covered the components of client- server architecture and the implementation of the architecture in developing a Data Guard role transition tool. The approach is defined and guided by a well-constructed methodology. With incorporating the Oracle recommended practices into the proposed transition approach the database administrators are enabled to perform an efficient role transition. The approach is derived into three (3) functions called HealthCheck, PreCheck and the actual role transition, Switchover.

The chapter also identifies the methodology and stages of ODaRT tool engineering process consist of Oracle best known practices. The check list of the three identified modules are constructed based on the Oracle best known practices. The chapter also covered the ODaRT tool development requirements, functions and modeling using UML diagrams.

4.2 Two-Tier Client/Server Architecture

Two-tier architecture is usually takes direct communication between client and server. The architecture has no intermediate between the client and server. The client layer serves functions to run the codes or scripts and write into the database which reside the data layer. The system developed following this architecture design usually a simple and direct way.

The ODaRT tool is perfect to engineer based on the two tier architecture since the tool will do a direct communication to the database server to perform administration tasks. The codes and SQL/DGMGRL commands are resides in the client layer to help maintain the codes easily and faster communication rather than having a separate middle layer. The disadvantage of having two-tier architecture is many client connections to the server would directly cause server performance degradation. But the ODaRT tool is used mostly for administration tasks by the DBAs to administrate the database by performing the regular DBA activity, hence choosing the client-server architecture is the best choice.

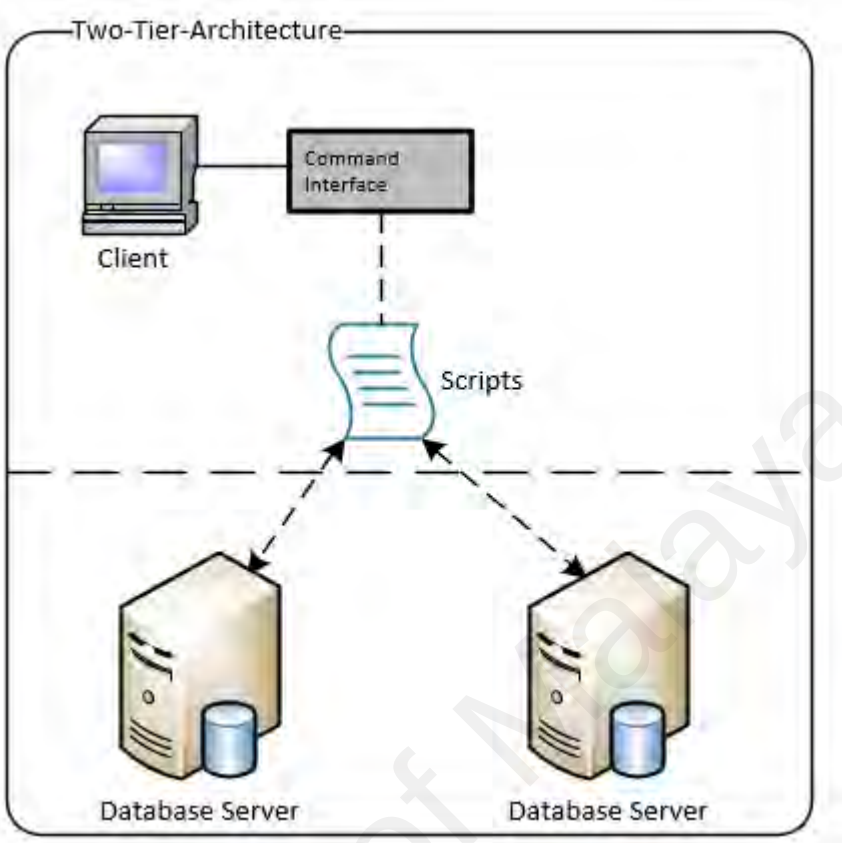

Figure 4.1 Client-Server/Two-Tier Architecture Design

The Figure 4.1 shows the client-server architecture where the client and server layers are separated to illustrate the two major components of client and server. The interactions of the subsystems between the layers are normally communicated through *SQL interactions*, *Remote Procedure Calls (RPCs),* or *pipes*.

4.3 Data Guard Broker Command Line and PL/SQL Programming Language

The structured query language or procedural language is a programming language used to access an Oracle Database("11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. ," ; Kyte, 2010).

In two-tier architecture, with Oracle Data Guard as a database component and the application component resides on the client machine, the communication of send and request information to the database server is done using SQL statements or a set of SQL statements embedded as one PL/SQL block

In Oracle, the SQL language is used as a standard language to query RDBMS database which is supported by the ANSI (American National Standard Institute) and defined as 'Database language SQL'. The Data Guard command-line interface (DGMGRL) enables us to manage a Data Guard broker configuration and its databases directly from the command line, or from batch programs or scripts. The Data Guard command-line interface can be used as an alternative to Oracle Enterprise Manager for managing a Data Guard configuration.

4.4 Data Guard Role Transition Approach

In the following section, we have summarized the best practices need to perform before a role transition specific to switchover activity. In the sections we shall elaborate more on the proposed role transition approach for running efficiently a switchover for a high availability Oracle Data Guard database.

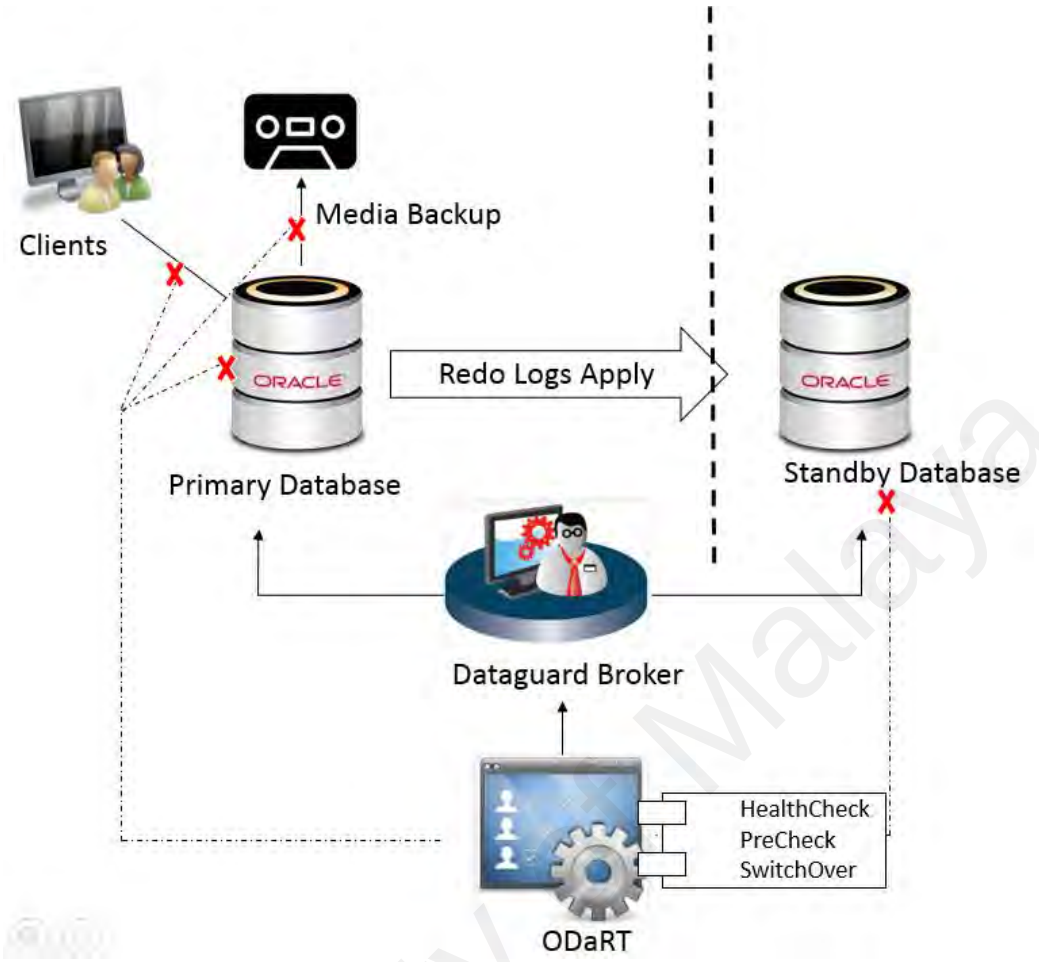

Figure 4.2 Proposed Role Transition Approach

Figure 4.2 shows the proposed the role transition approach. It explains the relationship between the automated tool ODaRT, Data Guard Broker, Primary/Standby databases and external clients including the backups. There are modules bundled with both SQL and DGMGRL commands to perform the switchover. The modules are categorized as Healthcheck, Prechek and Switchover. The sub-topics below explains in details about the three modules.

4.4.1 HealthCheck

Basically, HealthCheck is to be used for verifying the health of the database systems. It's containing all the needed check to ensure the database is healthy before and after a switchover.

It also can be used anytime by database administrator to determine the health of the database systems. The Perl module is embedded with some SQL scripts and DGMGRL commands to query and verify the health of the systems.

Table 4.1 HealthCheck checks list

Check#	Health Checks	Command
1	Verify disk groups free space	SQL
2	Verify observer location	DGMGRL
3	Verify configuration health	DGMGRL
4	Verify there is no apply delay for the target standby	SQL
5	Ensure online redo log files on the target physical standby have been cleared	SQL
6	Verify there are no large gaps	SQL
7	Verify primary and standby tempfiles match	SQL
8	Verify primary and standby disk location	SQL
9	Verify all datafiles are ONLINE	SQL
10	Verify primary and standby datafiles disk location	SQL
11	Verify primary and standby online redo log disk location	SQL

Table 4.1 shows the check list gathered from the Oracle recommended best practices document for performing the health check of Oracle Data Guard database. It also explains how the checks is queried, either by SQL or DGMGRL commands.

4.4.2 PreCheck

PreCheck is the list designed to check before executing a switchover an Oracle Data Guard database. The prerequisite that need to be executed before performing a switchover is highly recommended and mandated by Oracle. These checks are specific to run at an Oracle Data Guard with Broker enabled. The ODaRT tool is designated to perform the switchover for a database running with Oracle Broker feature. The switchover by using the DGMGRL method is selected to perform the switchover and was well explained in the literature chapter on why it was selected. Table 4.2 shows the checks will be done during a pre check of switchover based on the Oracle recommended best practices.

Table 4.2 PreCheck checks list

Check #	**Pre Checks**	**Command**
1	Suspend Scheduler jobs	SQL
2	Check for potential long running operation	SQL
3	Suspend backup jobs	SQL
4	Clear potential blocking parameters	SQL

4.4.3 Switchover

Switchover is the module to perform the switchover with using the Oracle Broker feature. The broker will use the own language of DGMGRL to perform the switchover.

After confirming the perquisites checks are passed it will get through to perform the switchover. Table 4.3 table shows the few steps it does for performing the switchover.

Table 4.3 Switchover tasks list

Task #	**Tasks**	**Command**
1	Perform switchover	DGMGRL
2	Resume scheduler jobs	SQL
3	Resume backup jobs	SQL

4.5 Development of ODaRT Tool

Our objective, to perform a database role transition while minimizing the impact of role transition on the overall database availability, consist of developing SQL script and DGMGRL to query or command in custom developed role transition tool, ODaRT. The ODaRT tool is a tool developed in using Perl scripting language. The tool engineering process (Figure 4.3), consist of currently accepted best practice in information system development which has Unified Software Development Process (USDP) (Jacobson, .et all).

The following stages are taken for the development of the ODaRT tool.

- **Scope of Requirements** stage describe objectives, functional and non- functional requirement of the ODaRT tool.
- **Requirement Modelling** decays the functional requirements in the form of use case components. Each use case is describe using details methodology.
- **Analysis model** provides ODaRT tool's analysis class diagram.
- **System Design** consist of high level subsystem component diagram and deployment diagram
- **Detailed design** deliverables of SQL queries and DGMGRL commands for the retrieval and command the task in the database.
- **Implementation** is the tool development using Perl scripting language, windows commands. A set of Oracle Data Guard Enterprise version 11g is installed in two Windows Server 2013 operating system. ODaRT tool use and Oracle JDBC drive to communicate with both the primary and standby Oracle database.

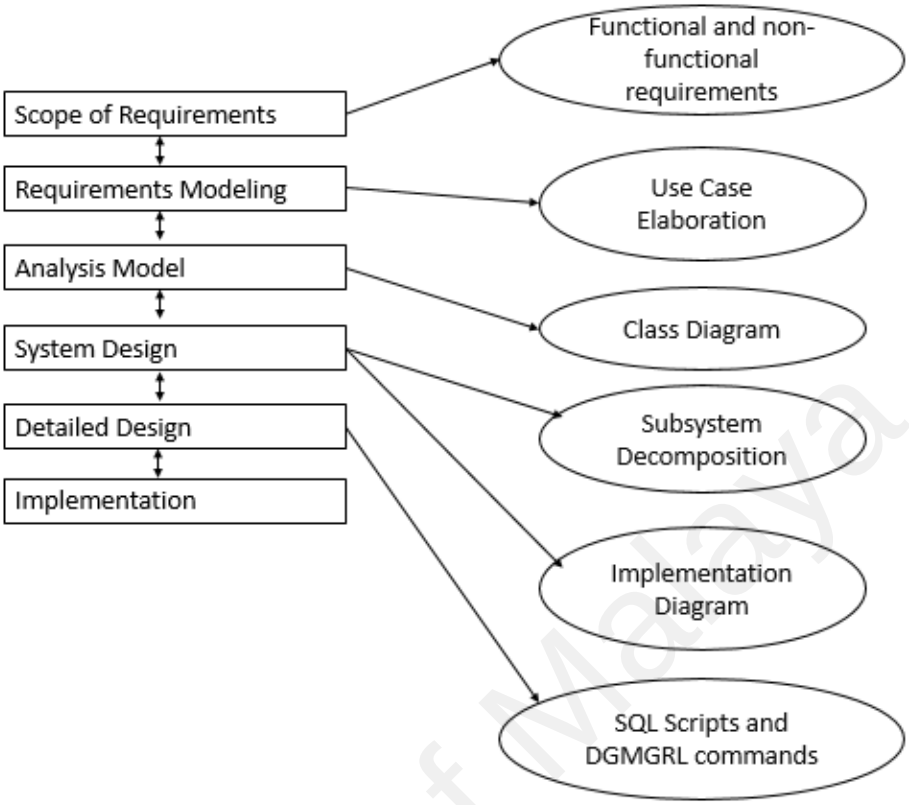

Figure 4.3 Stages of ODaRT Tool Engineering Process

Figure 4.3 illustrate the stages of engineering process in producing the automated tool called ODaRT.

4.6 Scope of Requirements

In this section we discuss in details about the aim, functional and non-functional of ODaRT tool. To achieve an error less role transition using

The goals of the ODaRT tool are:

- To deliver an approach for the Database Administrator to establish a connection to the Oracle Data Guard databases which are implemented in client/server architecture.

- To deliver an approach for the Database Administrator to perform a complete health check of both primary and standby databases to determine the status of databases are healthy and synch.
- To deliver an approach for the Database Administrator, based on the need to perform a role transition and convert the physical standby database to the production (primary) database, and to execute the SQL or DGMGRL commands for carrying out the role transition.
- To deliver a simple and 'one-command' approach for the Database Administrator to complete the pre- check before the role transition which it's requires to execute commands (SQL and DGMGRL), compare, validate and output the results.

The **functional requirements** of the ODaRT tool are:

- The ODaRT tool should able the user/Database Administrator to connect to database instance and check if the database is running as primary or standby database.
- The ODaRT tool should able the user/Database Administrator to perform health check of both primary and standby database and verify if any check is problematic.
- The ODaRT tool should prompt to user/Database Administrator to fix or update to correct values/parameter where ever applicable.
- The ODaRT tool should execute SQL/DGMGRL commands against both primary and standby database for all below checks and report out

- Verify disk group's free space
- Verify observer location
- Verify configuration health
- Verify there is no apply delay for the target standby
- Verify there is no block corruptions
- Logging long running SQL, RMAN jobs and MML sessions
- Ensure online redo log files on the target physical standby have been cleared
- Verify there are no large gaps
- Verify primary and standby tempfiles match
- Verify primary and standby disk location
- Verify all datafiles are ONLINE
- Verify primary and standby datafiles disk location
- Verify primary and standby online redo log disk location

- The ODaRT tool should decides to perform switchover (database role transitions) based on the output of the pre checks.
- The ODaRT tool should prompt to Database Administrator and initiate the switchover using DGMGRL.
- The ODaRT tool should check primary database status for every 30 seconds for all primary database resources during the switchover.
- The ODaRT tool should ensure the primary database is online and print the message for the Database Administrator to perform manual fix if database and or any of the resources are offline.

The **non-functional requirements** of the ODaRT tool are:

- *Security*. The ODaRT tool should only executed by authenticated Database Administrator. The password of the utmost privileged of SYS account should be protected and not visible in any prompt or log file.
- *Usability*. The ODaRT tool should be easy to operate for the users with minimal execution of commands.
- *Configurability*. The ODaRT tool should be easy to configure the database parameters and alerts exclusion for the health check and configurable to use for any Oracle Data guard Database.
- *Efficiency.* The ODaRT tool should be perform the role transition timely and without errors.

4.7 Requirement modeling

We have used the UML use case diagram to model all the firmed functional requirements. The detailed of the use case diagram is describe for each of the use case.

4.7.1 Use Case: DatabaseConnection

Figure 4.4 is the use case diagram for Database Administrator to make a database connection and table 4.4 shows the use case description. The Database Administrator should be able to connect to database instance and check if the database is running as primary or standby database. The ODaRT should report an error message if the database is not connectable or if the DBA connected to the primary database node.

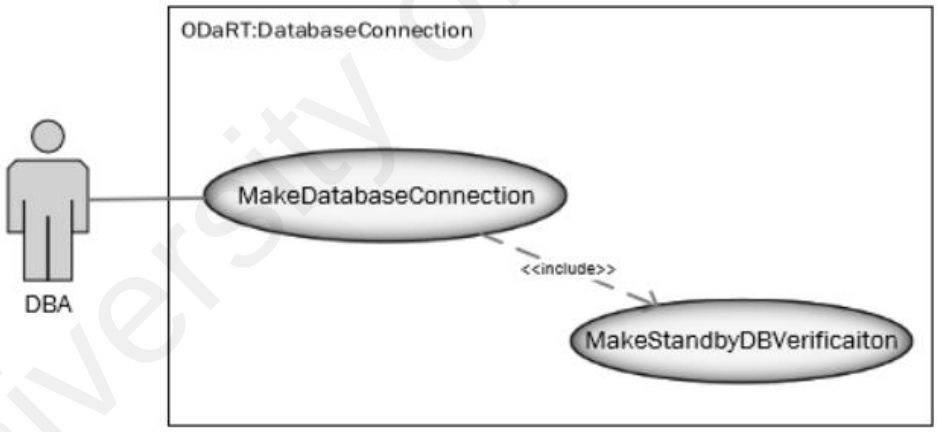

Figure 4.4 UML Use Case Diagram for Database Connection

- Use case Description: Database Connection

Table 4.4 Use Case Description for Database Connection

Use Case Name	Database Connection
Actors	Database Administrator (DBA)
Summary Description	Allow DBA to make connection to database.
Priority	Mandatory
Pre-condition	There is active network connection to database server
Basic Flow	1. Double Click on ODaRT execution 2. Display a command prompt 3. Type 'PRECHECK' or type 'p' + tab OR Type 'HEALTHCHECK' or type 'h' + tab OR Type 'SWITCHOVER' or type 's' + tab 4. If database is standby run and display the result. Else display error message
Alternative Flow	none
Post-condition	Exit the command prompt

4.7.2 Use Case: PreCheck

Figure 4.5 is the UML use case diagram for the Database Administrator to perform pre check on database before a switchover and the table 4.5 shows the description of the use case. The Database Administrator has option to kill the long running session, to clear the blocking parameters and to suspend the scheduler/backup jobs and will report the result at the final stages. The pre check will just continue to run and report the final result even if options are not selected.

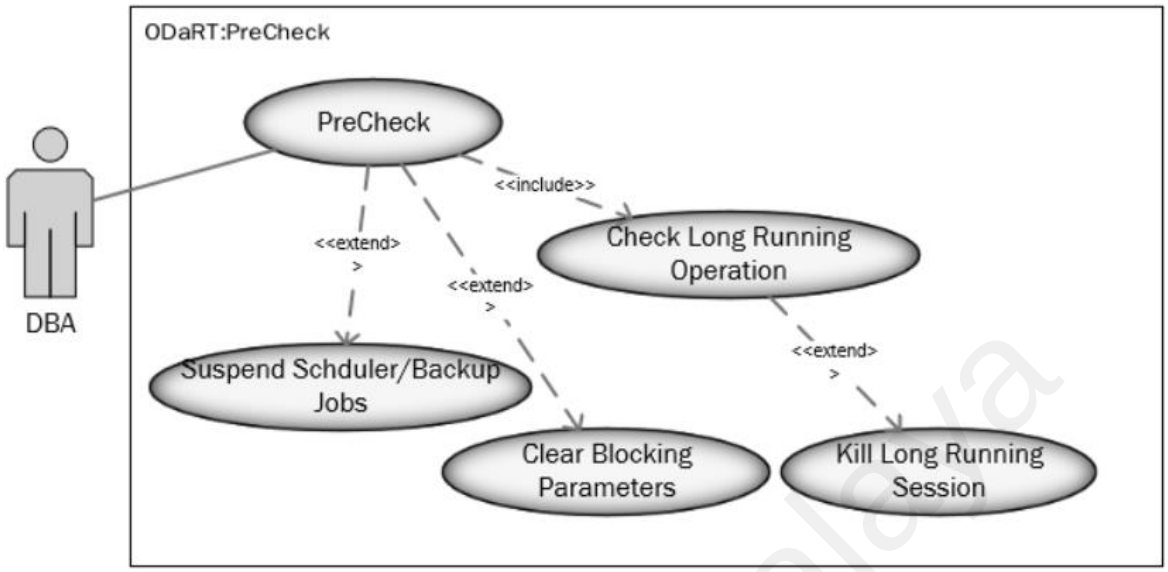

Figure 4.5 UML Use Case Diagram for PreCheck

- **Use case Description: PreCheck**

Table 4.2 Use Case Description for PreCheck

Use Case Name	PreCheck
Actors	Database Administrator (DBA)
Summary Description	Allow DBA to perform PreCheck on the database.
Priority	Mandatory
Pre-condition	The DBA has successfully connected to the database
Basic Flow	1. Double Click on ODaRT execution 2. Display a command prompt 3. Type 'PRECHECK' or type 'p' + tab 4. If database is standby. Run and display the result else display error message. 5. Prompt for 'Kill Long Running Session' if there is active. 6. DBA respond 'Yes' or 'No' for kill long running session 7. Execute the SQL statements to kill the jobs and output the log for 'Yes' response and if 'No' do continue.

	8. Prompt for 'Clear Blocking Parameters' if there is exist 9. DBA respond 'Yes' or 'No' for clear blocking parameters. 10. Execute the SQL and DGMGRL commands to clear the blocking parameters for 'Yes' response and if 'No' do continue. 11. Prompt for 'Suspend Scheduler/Backup Jobs' if there is running. 12. DBA respond 'Yes' or 'No' for suspend scheduler/backup jobs. 13. Execute the SQL commands to suspend the schedulers/jobs. 14. Print the output for DBA showing the system readiness for Switchover.
Alternative Flow	none
Post-condition	Exit the command prompt

4.7.3 Use Case: HealthCheck

Figure 4.6 is the UML use case diagram for the Database Administrator to perform general health check on database and the table 4.6 shows the description of the use case. Once the Database Administrator has run the 'HEALTHCHECK' command, the ODaRT tool will do the database connection and perform all specified database checks and report the output of the checks and save it as log text file.

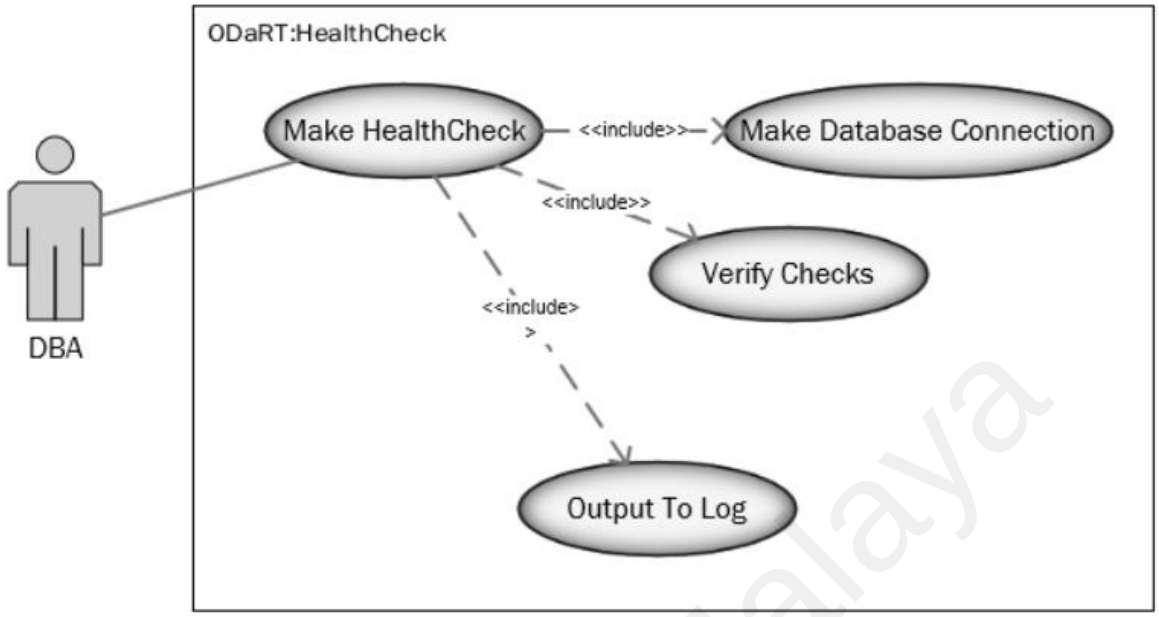

Figure 4.6 UML Use Case Diagram for HealthCheck

- **Use Case Description: HealthCheck**

Table 4.6 Use Case Description for HealthCheck

Use Case Name	PreCheck
Actors	Database Administrator (DBA)
Summary Description	Allow DBA to perform Healthcheck on the database.
Priority	Mandatory
Pre-condition	The DBA has successfully connected to the database
Basic Flow	1. Double Click on ODaRT execution 2. Display a command prompt 3. Type 'HEALTHCHECK' or type 'h' + tab 4. If database is standby. Run and display the result and write the output file to log file folder else display error message.
Alternative Flow	none
Post-condition	Exit the command prompt

4.7.4 Use Case: Switchover

Figure 4.7 is the UML use case diagram for the Database Administrator to perform the role transition of database by converting (switchover the role) the standby database to primary/production database and table 4.7 shows the description of the use case. Once the Database Administrator has run the 'SWITCHOVER' command, the ODaRT tool will do the database connection and perform all specified database checks and report the output and exit if the health check is reported error and not successful. The ODaRT tool will continue to prompt for optional 'Precheck' for the successful 'Healthcheck', upon the response the ODaRT too will perform the switchover by executing the DGMGRL command. The result of the switchover is shown and logged in a text file into the log file folder. The ODaRT tool will prompt for post tasks of resuming the scheduler or backup jobs upon successful switchover.

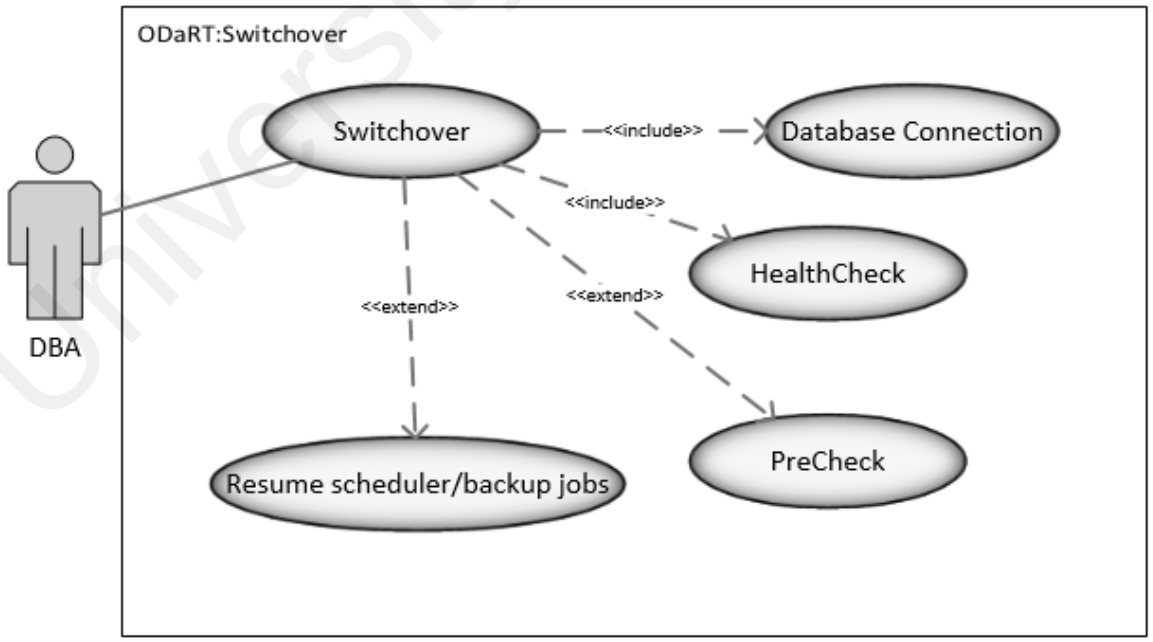

Figure 4.7 UML Use Case Diagram for Switchover

Use case Description: PreCheck

Table 4.7 Use Case Description for Switchover

Use Case Name	PreCheck
Actors	Database Administrator (DBA)
Summary Description	Allow DBA to perform Switchover on the database.
Priority	Mandatory
Pre-condition	The DBA has successfully connected to the database
Basic Flow	1. Double Click on ODaRT execution 2. Display a command prompt 3. Type 'SWITCHOVER' or type 's' + tab 4. If database is standby. Run and display the result and write the output file to log file folder else display error message. 5. Prompt for 'Precheck'. 6. DBA respond 'Yes' or 'No' for performing the pre check. 7. Output the pre check result for 'Yes' response and if 'No' do continue. 8. 8.Output the health check result and exit if error else continue 9. Prompt for pre checks. 10. DBA respond 'Yes' or 'No' for confirming the switchover. 11. Execute the DGMGRL command to perform role transition. 12. Print the output for DBA showing the switchover result. 13. Prompt for continue for post task else exit. 14. DBA respond 'Yes' or 'No' for post tasks (resume scheduler/backup jobs). 15. Execute the SQL commands to resume the jobs for the response 'Yes' else exit.
Alternative Flow	none
Post-condition	Exit the command prompt

4.8 Analysis Model

Analysis model is needed to elaborate the relationship, attributes and operation of a class for object-oriented system. The conceptual model diagram is very useful to implement the analyzed requirement. The ODaRT tool is analyzed and demonstrated using UML Class diagram to shows its aggregation and generalization. Figure 4.8 illustrate the class diagram for ODaRT tool.

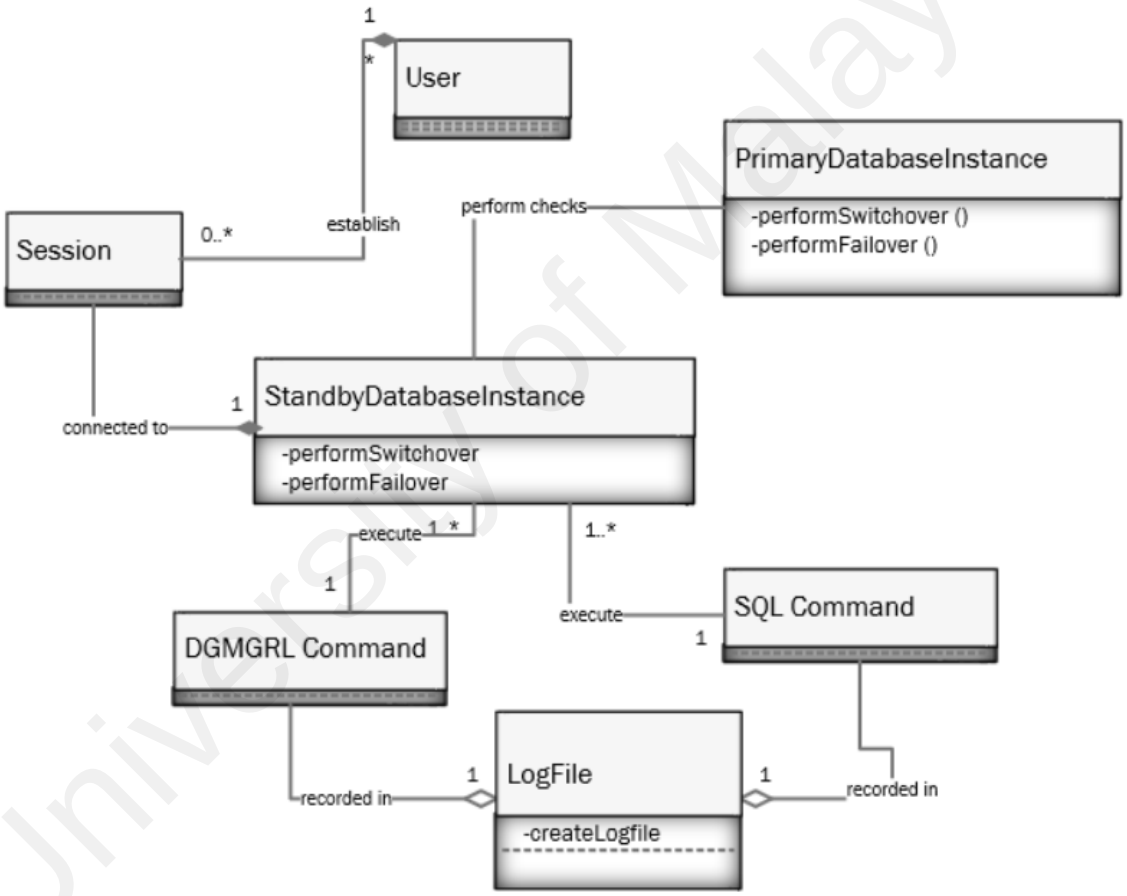

Figure 4.8 UML Analysis Class Diagram for ODaRT tool

From Figure 4.8, a *User* establish one or many *Session* connection to the *Standby Database Instance* and as per Oracle definite a *Session* belong to only one *User*. The *Standby Database*

Instance is associated with *Primary Database Instance* to perform the checks between both database objects. A connected Session can execute one to many *DGMGRL Command* or *SQL Command* and record the output to a *LogFile*. Only one *LogFile* is created by the *DGMGRL Command* or *SQL Command*.

4.9 System Design

Figure 4.9 illustrates the decomposition of the ODaRT tool's subsystem. The packages of perl modules which are embedded with SQL and DGMGRL commands/statements are resided in the client/application layer. The ODaRT tool establish connection using the Oracle SQL Net driver to communicate to the database objects which resides in the server/database layer. The ODaRT tool package writes the result or output of Perl modules executions to a text file.

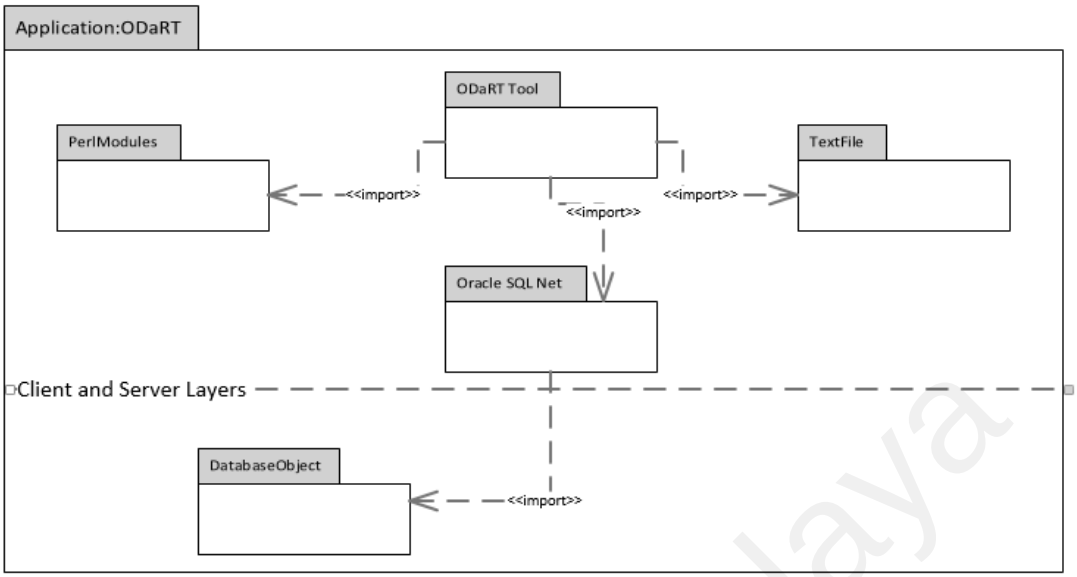

Figure 4.9 UML Class Diagram (subsystem decomposition) for ODaRT tool

Figure 4.10 illustrates the software and hardware components of ODaRT tool. The hardware's are client PC and Server whereas it is hosting the software of Oracle Data Guard 11g Database, Oracle Client 11g, Perl 5 and ODaRT tool contains the application scripts.

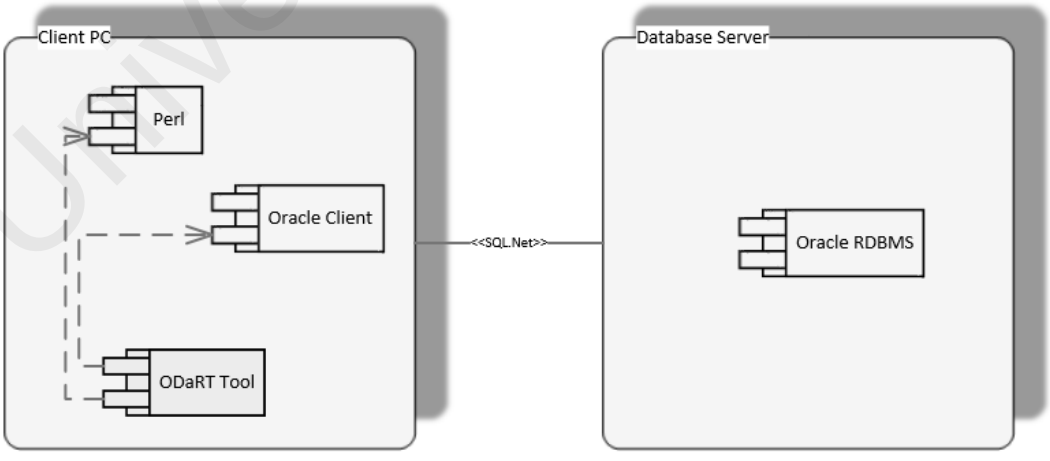

Figure 4.10 UML Deployment Diagram for ODaRT Tool

4.10 Detailed Design of ODaRT Tool

Source coding is very important segment in the process of software development. We have coded the ODaRT tool using the v5.28.0 and Perl developer tool called Padre, the Perl IDE for debug and coding, once identified the functional requirement and its logical analysis models. The function of each source code is explained in the table 4.8. The Perl scripts is embedded with SQL, DGMGRL and system administration commands which are needed to perform the tasks. We have hundred over SQL statements and DGMGRL commands are embedded in the ODaRT tool sources codes. The commands and statements are written based on the Oracle best practice to perform role transition which were discussed in the literature review. Some of major tasks related commands are discussed in the following sections.

Table 4.8 Detail of ODaRT tool source codes

Module	Function
ODaRT.pl	This is the main Perl program file call all the Perl module to execute. It's programmed and embedded with all the SQL commands needed to perform the subtasks. The functions of this Perl program file is discussed further in next section.
OraPasswordUtil.pl	This Perl program functioned to encrypt and decrypt the password in order to protect the passwords.
ParameterCompare.pl	This Perl program functioned to compare the database parameters.
Password.pl	This Perl program functioned to process the user inserted password.
AlertLogParser.pm	It's a Perl module to parse the log file and report the filter report out only the required log.
AutoInstCommon.pm	It's a Perl module to install the ODaRT tool.
CfgUtil.pm	It's a Perl module to verify all the configuration of the database.
CrsUtils.pm	It's a Perl module to execute the Oracle Stack and Cluster ware (SRVCTL and CRSCTL) commands for all the checks.

Database.pm	It's a Perl module that allow to login to database and run queries.
Dgmgrl.pm	It's a Perl module stored all the DGMGRL commands and logics for Broker execution.
Email.pm	It's a Perl module to send email notification on warning/critical errors to be addressed.
Logger.pm	It's a Perl module to output and keep the log.
OraParser.pm	It's a Perl module to parse the user inputs.
WMI.pm	It's a Perl module that allow to execute WQL queries using the DBI.
HEALTHCHECK.cmd	It's a windows command processor to call the ODaRT tool and execute health check.
PRECHECK.cmd	It's a windows command processor to call the ODaRT tool and execute pre check.
SWITCHOVER.cmd	It's a windows command processor to call the ODaRT tool and execute switchover.
SetPerlPath.cmd	It's a windows command processor to set the Perl libraries and environment.
ODaRT_MYDB1.xml	It's a user configurable variables input file to be processed during ODaRT installation and execution.

Table 4.8 described there were four types of file extension which are Perl program file as .pl, Perl modules as .pm, Windows Command batch file as .cmd and a document markup language as .xml. The following sections describing the queries, commands and statements extracted from Oracle Best Known Practice documents which are studied in the literature review.

4.10.1 Design of HealthCheck

Health check are done comprising mainly both SQL and DGMGRL queries/statements. There are also CRSCTL and SRVCTL commands are used to get the database resources status which provided by the Oracle utilities. There are eleven checks are done to conclude the health of the databases. All the checks important and show stopper if it's not fixed, but here we only took some major checks as an example to show the combination of SQL and

DGMGRL statements/commands usage in the tool. Figure 4.11 show the SQL command taken from Oracle Best Practice in order to perform disk group free space. The SQL select command of *free_perc* from *v$asm_diskgroup* view is incorporated as sub method of the main Perl program file.

```
################################################################################
#============Verify disk groups free space
#This method call for verify disk groups free space.
        verify_disk_free_space;
=cut
#-
================================================================================
################################################################################
sub verify_disk_free_space
{
        my $db_pri_handle = create_dba_primary_handle ();
        Logger::Fatal ("Failed to connect to PRIMARY database") unless
(defined ($db_pri_handle));

        my    $r    =    $db_pri_handle->GetAllRows    ("select    name,
round((free_mb/total_mb)*100,2) free_perc from v\$asm_diskgroup");
        my $pry_row_count = (scalar @$r);
        Data::Dumper::Dumper ($r);
                foreach my $l  (@$r) {
                Logger::Info ("Diskgroup : $l->[0] \t free_perc :   $l-
>[1] ");
                Logger::Fatal ("Diskgroup : $l->[0] has free_perc :  $l-
>[1]. Please fix the PRIMARY Node diskspace and rerun the Tool")
unless (defined($l->[1]) and ($l->[1] > 5));
        }
                my $db_psb_handle = create_dba_psb_handle ();
        Logger::Fatal ("Failed to connect to Physical Standby database")
unless (defined ($db_pri_handle));
        my    $r1    =    $db_psb_handle->GetAllRows    ("select    name,
round((free_mb/total_mb)*100,2) free_perc from v\$asm_diskgroup");
        my $psb_row_count = (scalar @$r1);
        Data::Dumper::Dumper ($r1);
                foreach my $l  (@$r1) {
                Logger::Info ("Diskgroup : $l->[0] \t free_perc :   $l-
>[1] ");
                Logger::Fatal ("Diskgroup : $l->[0] has free_perc :  $l-
>[1].Please fix the Physical Standby Node diskspace and rerun the Tool")
        unless (defined($l->[1]) and ($l->[1] > 5)) ;
        }
        log_SCN();
        return (1);
}
################################################################################
```

Figure 4.11 Verify disk groups free space

Figure 4.12 shows the DGMGRL command that incorporated in the Perl module to call the method of verify Data Guard configuration heath. The command '*show database verbose*' is native to DGMGRL language to view the Data Guard broker and database configurations.

```
#=============================================================================
#================Verify configuration health
sub get_database_config  ($)
{
        my $self = shift or die 'no self';
        my $t = shift or die 'no t';
        my $db = AppCfg::Get("DB_UNIQUE_NAME_${t}");
        my  $ret  =  Utils::CommandPipe  ("DGMGRL_$db",  "DGMGRL  -silent",
"connect /\@$db\nshow database verbose $db;\nquit\n");
        Logger::Info ("Ret :$ret\n");
        if ($ret =~ /^(ORA-16532|ORA-16596)/m) {
                Logger::Info ("Dataguard broker config not setup yet.");
                return ({});
        } elsif ($self->find_ora_errors ($ret)) {
                Logger::Warn ("DGMGRL SHOW DATABASE $db has ORA- errors");
                my $resp = Logger::Prompt ("Do you want to continue with
the Above ORA Error ? [Y = Yes N = No]", 'Y|N');
                return (undef) if ($resp eq 'N');
        }
        my $h = $self->parse_database_configuration ($ret);
        Logger::Warn ("Failed to parse dgmgrl database status.") unless
defined ($h);
        return ($h);
}
#=============================================================================
========
```

Figure 4.12 Verify configuration health

Checking the archive logs gap and successful transfer to standby database from primary database is very crucial for a Data Guard database. This is to make sure both production and the standby are always in synch and a failover or switchover performed without issue. Figure 4.13 shows the complete check of archival transfer and log gap suing SQL statements. The SQL queries are gathered from Oracle Best Practice documents. The database view *v$archived_log* contains all the needed data to validate the archive logs gap check.

```
#=================================================================
========
#============Verify there are no large gaps
sub check_dg_currency ()
{
        my $db_pri_handle = create_dba_primary_handle ();   #cretae
primary db handle
        my $rt = $db_pri_handle->GetAllRows("select thread#, sequence#
from v\$thread");
        return Logger::Warn ("Failed to query v\$thread") unless defined
($rt);
        if (@$rt) {
                foreach my $g (@$rt) {
                        Logger::Info ("thread#: " . $g->[0] . "
sequence#: " . $g->[1]);
                }
        }
                my ($status, $r1) = $db_pri_handle->GetOneRow ("SELECT
MAX(SEQUENCE#) as MAXSEQ FROM V\$ARCHIVED_LOG where
resetlogs_change#=(select max(resetlogs_change#) from V\$ARCHIVED_LOG)
AND THREAD# = 1 AND DEST_ID = 1");
        return Logger::Warn ("Failed to query V\$ARCHIVED_LOG") unless
defined ($r1);
        return Logger::Warn ("Failed find max archived_log SEQ") unless
defined ($r1->{MAXSEQ});
        my $primary_max_seq = $r1->{MAXSEQ};                    #max
sequence number from PRY DB
        Logger::Info ("Primary Max Archivelog SEQ: $primary_max_seq");

        my ($s, $r) = $db_pri_handle->GetOneRow ("SELECT DESTINATION,
STATUS, GAP_STATUS, ARCHIVED_SEQ# FROM V\$ARCHIVE_DEST_STATUS WHERE
ARCHIVED_THREAD# = 1 AND DEST_ID = 2");
        return Logger::Warn ("Failed find ARCHIVE_DEST_STATUS for
DEST_ID 2") unless defined ($r);
        return (0) unless Utils::EqCheck
("ARCHIVE_DEST_STATUS.DESTINATION", "DESTINATION", $r->{DESTINATION},
'Exp Dest', AppCfg::Get('DB_UNIQUE_NAME_B'), 1);
        return (0) unless Utils::EqCheck ("ARCHIVE_DEST_STATUS.STATUS",
"Cur Value", $r->{STATUS}, 'Exp Value', 'VALID');
        return (0) unless Utils::EqCheck
("ARCHIVE_DEST_STATUS.GAP_STATUS", "Cur Value", $r->{GAP_STATUS}, 'Exp
Value', 'NO GAP');

        ($status, $r) = $db_pri_handle->GetOneRow("SELECT
SWITCHOVER_STATUS FROM V\$DATABASE");
        return Logger::Warn ("Failed to query V\$DATABASE") unless
defined ($r);
        Logger::Info ('Switchover Status : ' . $r->{SWITCHOVER_STATUS});
        return 0 unless ($r->{SWITCHOVER_STATUS} =~ /TO STANDBY/i or $r-
>{SWITCHOVER_STATUS} =~ /SESSIONS ACTIVE/i);
                        my $db_psb_handle = create_dba_psb_handle ();

        Logger::Info ("Checking for ARCH GAP");
        my ($ee, $rr) = $db_psb_handle->GetAllRows ("SELECT
LOW_SEQUENCE#, HIGH_SEQUENCE# FROM V\$ARCHIVE_GAP");
        return Logger::Warn (" Failed to query V\$ARCHIVE_GAP") unless
defined ($rr);
        if (@$rr) {
                foreach my $g (@$rr) {
```

```
                        Logger::Warn ("ARCHIVE_GAP Low: " . $g->[0] . "
High Seq: " . $g->[1]);
                }
                return (0);
        }
        my ($e2, $r2) = $db_psb_handle->GetOneRow ("SELECT
MAX(SEQUENCE#) as MAXSEQ FROM V\$ARCHIVED_LOG where
resetlogs_change#=(select max(resetlogs_change#) from V\$ARCHIVED_LOG)
AND THREAD# = 1 AND DEST_ID =1 AND APPLIED= ?','YES');
        return Logger::Warn ("Failed to query V\$ARCHIVED_LOG") unless
defined($r2);
        return Logger::Warn ("Failed find max archived_log SEQ") unless
defined ($r2->{MAXSEQ});
        my $psb_max_seq = $r2->{MAXSEQ};
        Logger::Info ("Cross check MAX SEQ at PRIMARY ($primary_max_seq)
and PSB ($psb_max_seq)");
        return Logger::Warn ("PSB MAX Log Sequence ($psb_max_seq)
should be >= to PRIMARY Log Sequence ($primary_max_seq)") unless
($primary_max_seq - $psb_max_seq <= 1 );
        return Logger::Info ("Archive Log Shipping Verification
successful");
}
#=============================================================================
========
```

Figure 4.13 Verify there are no large gaps

The other example is shown in figure 4.14 is to check the database resources, the CRSCTL command is used to check and manage the clusterware resources. The CRSCTL utility is provided by Oracle under the grid home to check the resources status at the operation system prompt. The command '*crsctl status resource'* is used to check all the database resources and the output is dumped by calling another perl logger module.

```
#=============================================================================
#============Check database res status
sub get_crsctl_status
{
        my $self = shift or die 'no self';


        $self->{GRID_DB}->SetEnv ();
        my $ret = Utils::RunProcess ("crsctl status res");
        return (undef) unless defined ($ret);
        #Logger::Info ("CRSCTL STATURS RES output: $ret");

        my $h = {};
        my $cur;
        foreach my $line (split (/\n/, $ret)) {
                if ($line =~ /^\s*NAME=([^\s]+)\s*$/) {
```

```
                        die "duplicate RESOURCE NAME $1" if (exists ($h-
>{$1}));
                        $cur = $h->{$1} = {};
                } elsif ($line =~ /^\s*([A-Z]+)=(.+)$/) {
                        die "Bad Line $line" unless defined ($cur);
                        $cur->{$1} = $2;
                        if ($1 eq "STATE" and $2 =~ /ONLINE on (.+)/) {
                                $cur->{ONLINE_ON} = $1;
                        }
                }
        }
        Logger::Info ("CRS:" . Data::Dumper::Dumper ($h));


        return ($h);
}

#=============================================================================
```

Figure 4.14 Verify database cluster ware resources

4.10.2 Design of PreCheck

The pre check before a switchover role transition is to verify the readiness of the both primary and standby database for a switchover. The executions are done mainly using the SQL queries gathered from Oracle best known practice documents. The queries not only validate but also command to alter the database configuration or jobs based on the DBA selection. There are four checks been elaborated in the design and development of the approach chapter, figure 4.15 shows one of the checks. The incorporated SQL commands in figure 4.15 is basically verify for any scheduler jobs and prompt to the user to perform the suspension of the jobs.

```
#########################################################################
#####============Suspend Grid Control Scheduler Jobs
sub Disable_GC_Scheduler_jobs
{

        my $job_sql_disable = '';
        my $job_sql_enable = '';
        my $sql_suspend_gc_job = 'SQLFILE\Disable_GC_job.sql';
        my $db_unique_name = AppCfg::Get('DB_UNIQUE_NAME_A');
        my  $dbname  =  substr($db_unique_name,0,length($db_unique_name)-
2);

        if ($gc_connect_state){
                my $db_sysman_handle = create_sysman_handle ();
                my $sql_statement = "select job_id, job_name, job_type,
scheduled_time,       round((scheduled_time-sysdate)*60*24)      To_Run,
execution_id, status from sysman.mgmt\$job_execution_history where upper
(target_name) like upper ('$dbname%') and status in ('Scheduled')";
                Logger::Info("SQL Statement $sql_statement");
                my      $r1_GC      =      $db_sysman_handle->GetAllRows
($sql_statement);
                Logger::Warn          ("Failed          to          query
sysman.mgmt\$job_execution_history") unless defined ($r1_GC);
                Logger::Warn          ("Failed          to          query
sysman.mgmt\$job_execution_history") if ($r1_GC =~ /ORA-/);
                my $disable_job_GC = (scalar @$r1_GC);        #To   check
if scheduled jobs found.

                Logger::Info("PRIMARY Node: NO  GC Scheduler Jobs Found
Scheduled") if ($disable_job_GC == 0);


                if ($disable_job_GC > 0) {#log list of GC Scheduler  Jobs
on PRY which are in Scheduled state

                        Logger::Info ("List of GC Scheduler  Jobs which
are having status Scheduled.\n");
                        foreach my $l  (@$r1_GC) {

                                Logger::Info   ("job_id   :   $l->[0]   \t
job_name : $l->[1] \t scheduled_time:  $l->[3] \t To_Run : $l->[4] mins
\t status : $l->[6]");
                                $job_sql_disable         .=         "exec
sysman.mgmt_job_engine.suspend_job('$l->[0]');\n";
                        }
                        $job_sql_disable .= "commit;";

                        Utils::write_file($sql_suspend_gc_job          ,
$job_sql_disable);
                        Logger::Info("Disable  GC  Scheduler  Jobs\n  "  .
$job_sql_disable);


                        #Running  GC  disable  script  to  suspend  the
scheduled jobs
                        my  ($status,  $spool_file)  =  $db_sysman_handle-
>RunSqlScriptEx($sql_suspend_gc_job);
                        my $log_text = Utils::Slurp ($spool_file);
                        Logger::Warn ("Failed to get spool contents from
$spool_file") unless defined ($log_text);
```

```
                        Logger::Warn ("$spool_file has ORA-Errors") if
($log_text =~ m/ORA-/);
                        Logger::Info             ("Output             of
$sql_suspend_gc_job:\n$log_text");

                        #Displaying status of job after running the
disable commands
                        my $sql_statement_GC = "select job_id, job_name,
status from sysman.mgmt\$job_execution_history where upper (target_name)
like     upper     ((select     substr     ('$db_unique_name',1,length
('$db_unique_name')-2) from v\$database) || \'\%\') and status in
('Scheduled')";
                        Logger::Info("SQL Statement $sql_statement");
                        my     $r1_GC=     $db_sysman_handle->GetAllRows
($sql_statement_GC);
                        Logger::Warn       ("Failed       to       query
sysman.mgmt\$job_execution_history") unless defined ($r1_GC);
                        foreach my $l  (@$r1_GC) {

                                Logger::Info   ("job_id   :   $l->[0]   \t
job_name : $l->[1] \t  status : $l->[2]");
                        }
                }
        }
        return (1);
}
#########################################################################
```

Figure 4.15 Suspend scheduler jobs

Clearing a long running session or operation is crucial check before a switchover as it will cause the switchover to be hung and wait for the operation to complete first. This check would help to avoid the longer downtime which one of the main purpose of this research. Figure 4.16 shows the SQL script used to query and kill a long running session or operation particularly Media Management Layer (MML) transaction which is an RMAN session created to communicate with a third-party backup software.

```
#########################################################################
############Log and Kill long running operations

sub log_and_kill_MML_session
{
        my $db_pri_handle = create_dba_primary_handle ();              #
handle to connect to PRY DB
        Logger::Fatal ("Failed to connect to PRIMARY database.") unless
(defined ($db_pri_handle));
        my $MML_transaction_sql = 'SQLFILE\kill_MML_transaction.sql';
        my $sql_kill_MML_transaction = '';
```

```
        my $sql_stmt =  "SELECT s.sid,s.serial#,p.SPID, s.EVENT,
s.SECONDS_IN_WAIT,sw.STATE, s.CLIENT_INFO FROM  V\$SESSION_WAIT sw,
V\$SESSION s, V\$PROCESS p WHERE  sw.EVENT LIKE '%MML%' AND
s.SID=sw.SID AND  s.PADDR=p.ADDR";
        my $r1 = $db_pri_handle->GetAllRows ($sql_stmt);
        return Logger::Warn (" Failed to query v\$SESSION_WAIT and
v\$session") unless defined ($r1);
        my $row_count = (scalar @$r1);

        #Data::Dumper::Dumper ($r1);
        Logger::Info ("MML tape backup is running.") unless ($row_count
> 0);
        Logger::Info ('Number of MML Sessions  : '.$row_count);

        if (-f $MML_transaction_sql) {
                        Logger::Warn ("Failed to remove FILE :
$MML_transaction_sql : $!") unless unlink ($MML_transaction_sql);
                        Logger::Warn ("Failed to remove old PFILE :
$MML_transaction_sql : file found after unlink!") if (-f
$MML_transaction_sql);
        }
        if ($row_count > 0) {
                foreach my $l  (@$r1) {
                        Logger::Info ("SID : $l->[0] \t SERIAL# : $l-
>[1] \t Event :  $l->[3] \t Seconds_IN_Wait : $l->[4]");
                        $sql_kill_MML_transaction .= "alter system kill
session '$l->[0],$l->[1]' immediate;\n";
                }
                Logger::Info ("Long Running MML Transactions :\n
$sql_kill_MML_transaction");

                my $pv = join (",", map { $_->[1]} @$r1);
                Logger::Info ("SID of LONG running MML transactions " .
$pv);

        Utils::write_file($MML_transaction_sql,$sql_kill_MML_transaction
);
                Logger::Info ("SqL Script to kill MML transactions are
created in $MML_transaction_sql");
                Logger::Info ('Running Script to kill MML transaction');

                #running  script to kill long running transaction
                my ($status, $spool_file) = $db_pri_handle-
>RunSqlScriptEx($MML_transaction_sql);
                my $log_text = Utils::Slurp ($spool_file);
                Logger::Info ("Output of
$MML_transaction_sql:\n$log_text");

        }

        return (1);
}
##############################################################################
```

Figure 4.16 Check and kill potential long running operations

4.10.3 Design of Switchover

The design for switchover is basically to perform the action of switchover role transition and this is done by using the DGMGRL command. In the tool's switchover design there are two post task is performed after the switchover. Figure 4.17 shows the source of Perl program embedded with DGMGRL command of '*switchover to <StandbyDatabaseName>*' is to perform the role transition where the production database transit to be a standby database meanwhile the current standby database becomes as primary database.

```
#=============================================================================
==================Perform Switchover role transition
sub SwitchOver ($$)
{
my $self = shift or die 'no self';
my $t = shift or die 'no t';
my $to = shift or die 'no to';
die 'bad t $t or to $to' unless ($t =~ /^(N|O)$/ and $to =~ /^(A|B)$/);

my $db = AppCfg::Get("DB_UNIQUE_NAME_${t}");
my $db_to = AppCfg::Get("DB_UNIQUE_NAME_${to}");

my $h;
for (my $retry = 0; $retry < 2; $retry++) {
$h = $self->get_config ($t);
last if (defined ($h) and defined ($h->{CONFIGURATION}));
Logger::Info ("Retrying after 10 seconds.");
sleep(10);
}
Logger::Warn ("Unable to get DGMGRL Config or Config not setup yet.")
unless (defined ($h) and defined ($h->{CONFIGURATION}));
return Logger::Info ("Primary is already $db_to. Skipping switch
over.") if ($h->{PRIMARY} eq $db_to);

Logger::Info ("Performing switch over to $db_to");
my $input = <<"EOF";
connect /\@$db
switchover to $db_to;
quit
EOF
$input =~ s/^\s+//mg; $input =~ s/\s+$//mg;

Logger::Info ("Switchover start time : " . POSIX::strftime ("%Y-%m-%d
%H:%M:%S", localtime()));
AppCfg::Set ('SWITCHOVER_START_TIME', POSIX::strftime ("%Y-%m-%d
%H:%M:%S", localtime()));
```

```
AppCfg::Set ('SWITCHOVER_START_TIME_1', time());
my $switchover_start_time =  time();

my $ret = Utils::CommandPipe ("DGMGRL_$db", "DGMGRL -silent", $input);
Logger::Info ("RET:\n$ret\n");
return Logger::Warn("Failed to perform Switchover to  $db_to") if
($ret =~ /^ORA-/m);
my $switchover_end_time =  time();
AppCfg::Set ('SWITCHOVER_END_TIME', POSIX::strftime ("%Y-%m-%d
%H:%M:%S", localtime()));
Logger::Info ("Switchover End time : " . POSIX::strftime ("%Y-%m-%d
%H:%M:%S", localtime()));
my $switchover_duration = $switchover_end_time -
$switchover_start_time;
AppCfg::Set ('SWITCHOVER_DURATION', $switchover_duration);

Logger::Info ("Switchover Completed in $switchover_duration seconds");
Logger::Info ("Waiting 20 Seconds After Successful Switchover for
datagaurd to sync up");
sleep(20);
return Logger::Warn ("Unable to very clean switch over to $db_to")
unless Utils::WrapAction ("Validating switch over to $db_to",
\&validate_primary_after_switch_over, $self, $t, $db_to);

return Logger::Info ("Switchover successful.");

}
#=================================================================
```

Figure 4.17 Perform switchover

4.11 Implementation of ODaRT

The ODaRT tool was implemented using a vast array of software tools and hardware. Table 4.9 shows the hardware required to develop the ODaRT tool. A powerful hardware were selected to host the Oracle Data Guard databases to replicate a real production database which handle usually a huge load. For an Oracle Data Guard setup, it is at least two systems to install primary and standby database hence two set of system used for this tool implementation.

Table 4.9 Hardware system requirement

Hardware	Description
Processor	Intel(R) Xeon(R) CPU @ 2.00GHz
System Model and Type	VMware Virtual Platform x64-based PC

Installed Memory	10GB
Network Adaptor	WAN Miniport (SSTP)
Storage	HP HSV450 SCSI Disk Device
Storage Capacity	100GB

The table 4.10 shows the software requirement to develop the ODaRT tool. The latest Oracle Data Guard Database version is chosen as of the research writing period, this is to ensure the newer Oracle feature is not supersede the objective of this research.

Table 4.10 Software system requirement

Software	Description
Microsoft Windows Server 2008 R2 Enterprise	Operating system
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0	RDBMS with Data Guard and Automatic Storage Management option
Perl v5.10.0	Programing Language
Padre The Perl IDE v 0.94	For Perl Application Development and Refactoring Environment

We chose Perl 5 programming language to develop the ODaRT tool as its very capable object-oriented and functional programming. The other reason to choose Perl is easily can be integrated and embedded with third-party database integration interface which is Oracle software we used. The freely available open source modules from CPAN (Comprehensive Perl Archive Network) helped very much in developing the tool. We ODaRT tool is modelled, developed, debugged, refactored and optimized through the help of Padre Integrated Development Environment (IDE) version 0.94.

In Figure 4.18, shows the Perl programmed oracle database connection made through Oracle DBI using Oracle TNS entry (Transparent Network Substrate) with the TCP/IP network protocols which resides in the Oracle Home directory.

```
#====================================================================
=pod
$ret = $db->connect ()
return
1 For Success
0 Other unhandled error
-1 Oracle not available
=cut
#====================================
sub connect
{
        my $self = shift or Logger::Fatal "No self";
        my $dbh;

        $self->SetEnv();
        if  (defined  ($self->{TNS_NAME})  &&  (($self->{TNS_NAME}  =~
/odart/i) or  ($self->{TNS_NAME} =~ /^odart_(mydb|dss)$/i))) {
                $dbh     =     DBI->connect     ('dbi:Oracle:',     $self-
>{CONNECT_STRING}, '');
        }
        else {
                $dbh     =     DBI->connect     ('dbi:Oracle:',     $self-
>{CONNECT_STRING}, '', {ora_session_mode => main::ORA_SYSDBA});
        }
        if (!defined($dbh))
        {
                Logger::Info "DBI Error while connecting to Database",
$DBI::errstr;
                if ($DBI::errstr =~ /^ORA-01034:/)#Oracle is not available
                                                        {
                      return -1;}
                Logger::Error  "Failed to Connect to Database using, ",
$self->{CONNECT_STRING};
                return 0;                }
        Logger::Info  "Successfully  Connected  to  database  ",  $self-
>{CONNECT_STRING};
        $self->{CONNECT_HANDLE} = $dbh;
        return 1;
}
#====================================================================
```

Figure 4.18 Database connection through Oracle DBI using TNS

4.12 Summary

An effectual automated database role transition requires a set of accurate and reliable procedures to perform the role transition. In this chapter, the prospective of aim, functional and non-functional of ODaRT tool are assembled in details for the development of the tool. The aiding and illustrative tools of UML use case diagram, analysis model and system design are used to build the automated role transition tool, ODaRT.

Each of the use case is described in detail with use case description and this will help to model the tool as per the user requirements. In addition, the identified three modules of health check, pre check and switchover are incorporated with the extraction of SQL and DGMGRL commands from Oracle best known methods. Finally we have recognized the hardware and software systems needed for the development of the ODaRT tool.

CHAPTER 5

RESULT AND DISCUSSION

5.1 Main Features

In the following the sections, we have shown how the database administrator can use the ODaRT tool in order to perform one of their standard task of checking the health of the database and role transition of switchover. A health check of database is performed whenever a database administrator is the state of doubtless of the functionality of the database. It's a mandatory task need to be performed by database administrator before a role transition to make sure the primary and standby database healthy and able to support the role transition. The main features of the ODaRT tools is demonstrated in the following segregated paragraphs.

- HealthCheck

The screenshot shown in figure 5.1 is a command prompt invoked by an Administrator privileged DBA. The tool directly change its directory to ODaRT installation directory upon the database administrator invoked the ODaRT execution batch file. The DBA shall type 'h' + tab or in full command of *HEALTHCHECK.cmd* in order to perform the heath check of the database. Also the full instruction are provided to the Database Administrator in user manual document.

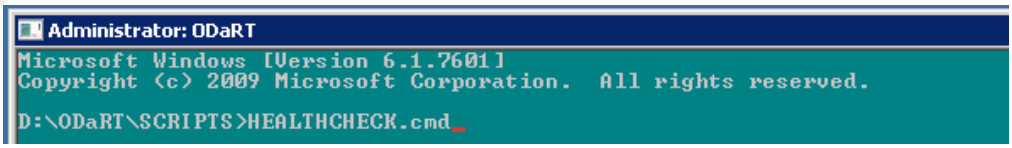

Figure 5.1 Automated Database health check instruction

Upon execution the tool will perform the needed health check passing through all the checks discussed in previous chapters. The checks are adopted from Oracle best known practice to

perform a database health check. The ODaRT tool will print the summary of the execution and verification. Also complete log of the execution will be logged in a text file kept in a folder named LOGS. The figure 5.2 shows the completion and summary of the health execution, and in this screenshot it posted some warnings regarding the database health for the invalid objects in database.

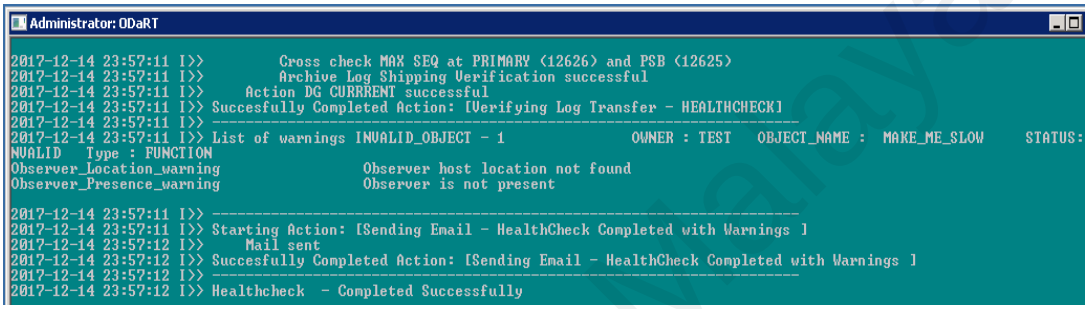

Figure 5.2 Completion of automated database health check

- PreCheck

All the features in ODaRT tool is invoked with the administrator privileged since the database administration task is allowed for privileged DBAs. The figure 5.3 shows the execution of PreCheck command batch file to perform database prerequisites check for a database role transition of switchover. The same method as HealthCheck is applicable for all the execution in ODaRT tool. This is to make sure the execution are simple and easily remembered by the database administrators.

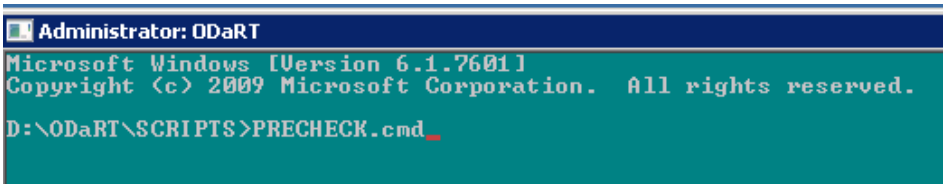

Figure 5.3 Instructing for a database switchover prerequisites check

The figure 5.4 shows a prompt by the ODaRT tool which it's checked the running scheduler jobs and prompting to user for disablement. The user or database administrator can either proceed or skip the disable if he/she not going to perform the switchover after this.

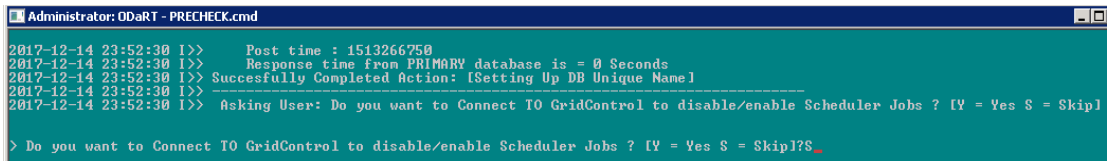

Figure 5.4 Pre check prompt for disabling scheduler jobs

The figure 5.5 shows the completion of the *PRECHECK.cmd* execution with providing the all the warnings to be checked by database administrator. In this figure it thrown two warnings which could led to failure of role transition if this is not taken care before performing database switchover. The warnings in this figure is list of long running operations that were running more than thirty(30) minutes and the other warning is observer location is not found.

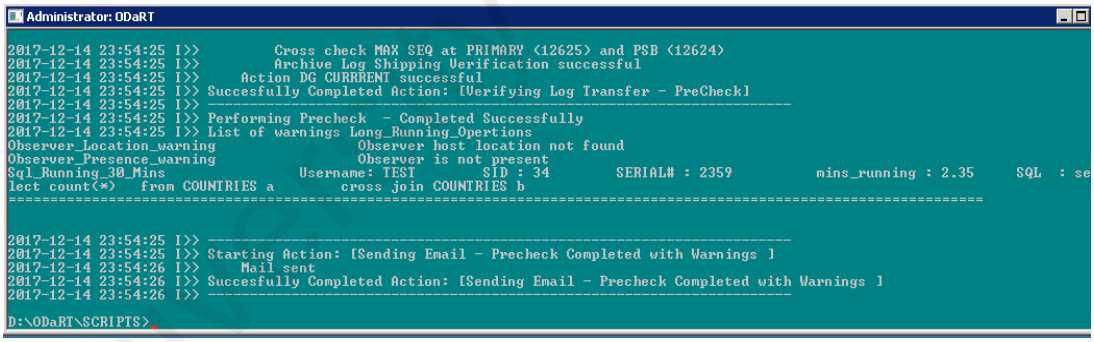

Figure 5.5 Completion of switchover prerequisites check

- Switchover

The switchover feature is the main component in this tool beside the other features. There are several small features are integrated for this switchover component. The figure 5.6 shows the invoking of switchover database role transition. The switchover is invoked either by typing 's' + tab in the keyboard or type the full command of *SWITCHOVER.cmd*.

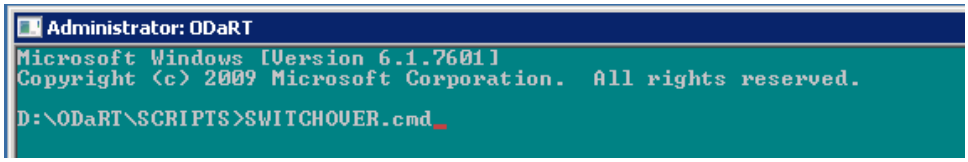

Figure 5.6 Instructing for a database switchover role transition

As per Oracle best known practice a complete pre check of the both primary and standby database is very crucial for a successful role transition. Thus, to call the prerequisites check module before the switchover is important. The figure 5.7 shows the prompt for user to continue for a prerequisites check before the continuing to role transition.

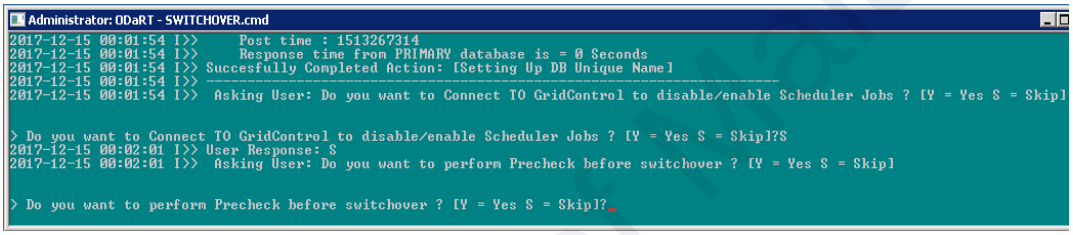

Figure 5.7 Prompt for prerequisites check before switchover

The database administrator is given option to skip prerequisites check if he/she already done prerequisites check against the database. For a successful switchover containment from application/user connection is very important to avoid the hung in switchover and to avoid data loss. A switchover without closing securely a user session could cost data loss. The figure 5.8 shows the prompt for database administrator to kill any long running or user session which running more than thirty (30) minutes.

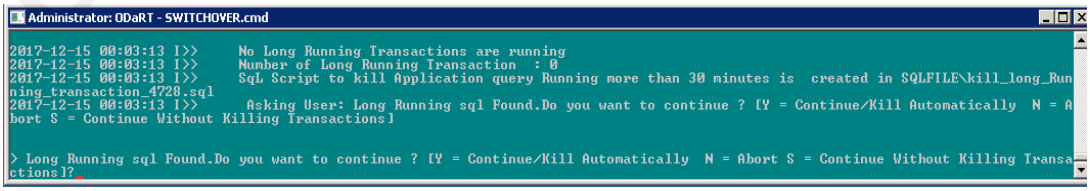

Figure 5.8 Prompt for kill long running session

In Oracle Data Guard database is always safe to disable the fast start failover feature before performing a role transition. This is recommended by Oracle and stated in its best known

practice for preventing the observer initiate a failover back to original primary database if there is an issue in standby database. The figure 5.9 shows an info and prompt to database administrator to commit the switchover to standby database.

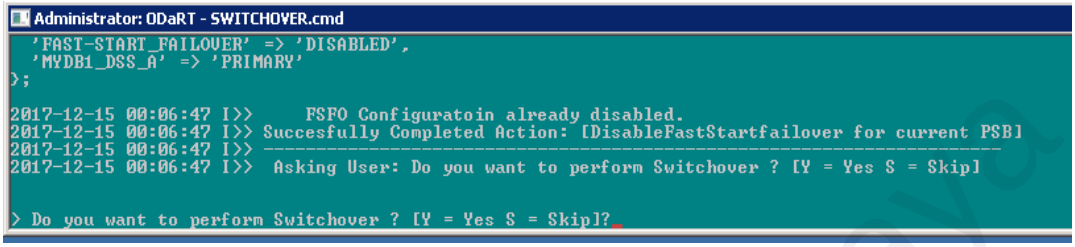

Figure 5.9 Prompt for confirming a switchover

A completed database validation and health check is needed upon a successful role transition before handover the primary database for production serve. The figure 5.10 shows basically a prompt for running post switchover verification which it will invoke again the health check module.

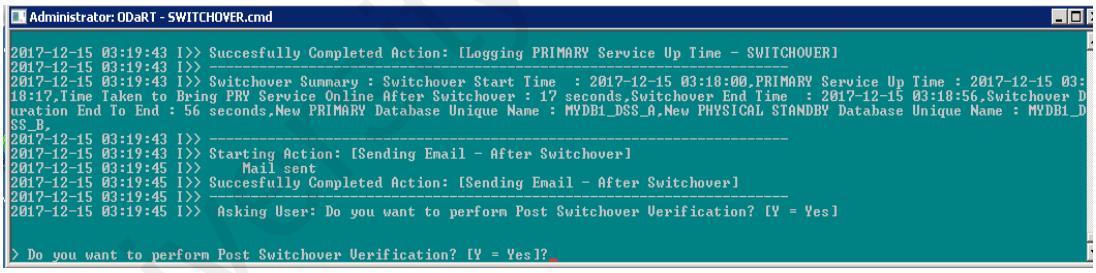

Figure 5.10 Prompt for post switchover verification

The figure 5.11 shows the completion of a database role transition, switchover. The successful switchover will have the summary of the switchover i.e time taken to perform the switchover and information about the new primary database which is serve the production. The information of switchover is also logged in a text file for a future reference.

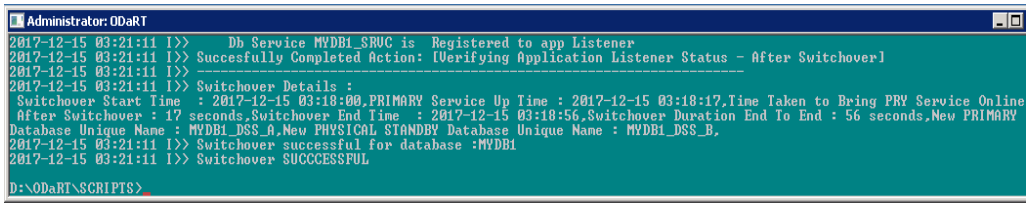

Figure 5.11 Complete of switchover

- Exception Handler

There are few exception handlers been used in ODaRT tool, the figure 5.12 shows an exception handler thrown during the execution of ODaRT. The tool is configured to run from standby database for any activity and only connection should be initiated to primary database for checking and this is to avoid any mishandle in primary database which is servicing the production.

Figure 5.12 Example of exception handler

5.2 Testing

We have conducted several testing in the development of this ODaRT tool. The outlined testing process are Unit Testing, Integration Testing and System Testing. The result of this testing is evaluated and presented in result section. The testing is conducted to validate the interaction of the codes and reduce the faultiness of the tool. Moreover the testing helpful to provide a documentation regarding the solution taken or experienced gained for future reference, this will helpful to secure and maintain the tool in future. The tests are conducted to produce well-structured of program, and to overcome the errors and bugs.

The figure 5.13 shows the overall testing methodology conducted in this research. The testing are categorized into two (2) section which are black box testing and white box testing. In black box testing the accessibility and system testing is conducted which are the design and

implementation of ODaRT tool is not known by user. Meanwhile, the white box testing are integration and unit testing which are known to user. The following section shall describe briefly about both the stages of the testing. We performed the test to check both the functionalities of functional and non-functional based on gathered user requirements, also to check the efficiently of the implemented ODaRT tool.

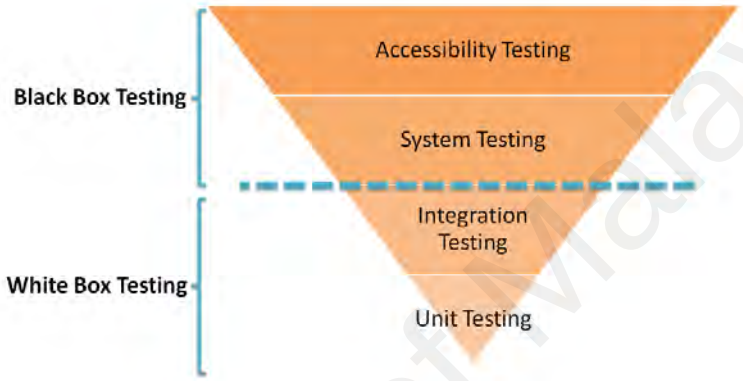

Figure 5.13 The testing methodology used for ODaRT tool

The table 5.1 shows the twenty (20) test cases prepared and tested at the each of the stages of execution. The functionality, performance and installation testing been conducted for the tool based on the twenty scenarios listed in the table. The table also elaborate the action should take by the tool if the expected results are not met. The tests are conducted in both the primary and standby databases.

Table 5.1 ODaRT tool black box test cases

Sl#	Description	Checks on Primary	Checks on PSB	Expected Results	Time to wait	Fail Action - (Command execution Fail)	Tool Action if Expected Results are not met
1	Verify disk groups free space	select name, round((free_mb/total_mb)*100,2) free_perc from v$asm_diskgroup;		>= 5%		FO	Fail Tool - Fix Disk Space issue and re-run tool.
			select name, round((free_mb/total_mb)*100,2) free_perc from v$asm_diskgroup;	>= 5%		Manual Action/Troubleshooting	Fail Tool - Fix Disk Space issue and re-run tool.
		select dbms_flashback.get_system_change_number() SCN from dual;		<scn>		FO	
		select sequence#, status from v$log where <scn> between first_change# and next_change#; --> If no row selected, execute below query		no rows selected or STATUS: CURRENT/ACTIVE		FO	
		select distinct sequence#, deleted, status from v$archived_log		DELETED='NO' and STATUS='A'		FO	Fail Tool - Fix Disk Space issue and re-run tool.

		where <scn> between first_change# and next_change#;					
2	Verify observer location	select fs_failover_observer_present, fs_failover_observer_host from v$database;		<host_name> and observer_present=YES		FO	Information - Logging only.
3	Verify configuration health		show configuration;	configuration status: SUCCESS Fast-Start Failover: ENABLED		Manual Action/Troubleshooting	To fix any warning/ORA-message or errors. Tool log output of error. Manual Action.
4	Verify sufficient number of archiver processes >= 4	show database <PRI-DB_UNIQUE_NAME> 'LogArchiveMaxProcesses';		5		Manual Action/Troubleshooting	Fix to POR and rerun tool.
			show database <PSB-DB_UNIQUE_NAME> 'LogArchiveMaxProcesses';	5		Manual Action/Troubleshooting	Fix to POR and rerun tool.
5	Verify there is no apply delay for the		show database <PSB-	0		Manual Action/Troubleshooting	Fix to POR and rerun tool.

	target standby		DB_UNIQUE_NAME> DelayMins				
6	Ensure online redo log files on the target physical standby have been cleared	show database <PRI-DB_UNIQUE_NAME> 'LogFileNameConvert';		<non-null value> and POR settings.		Manual Action/Troubleshooting	Fix to POR and rerun tool. Check for Valid value.
			show database <PSB-DB_UNIQUE_NAME> 'LogFileNameConvert';	<non-null value> and POR settings.		Manual Action/Troubleshooting	Fix to POR and rerun tool. Check for Valid value.
7	Verify there are no large gaps	select thread#, sequence# from v$thread;		PRI sequence#		FO	
			select thread#, max(sequence#) from v$archived_log where applied = 'YES' and resetlogs_change# = (select resetlogs_change# from v$database_incarnation where status = 'CURRENT') group by thread#;	PSB sequence# should be within 1 or 2 of the PRI current sequence number		Manual Action/Troubleshooting	>2 - Abort and use Manual Process to address Gap.

8	Verify primary and standby tempfiles match	select tmp.name filename, bytes, ts.name tablespace from v$tempfile tmp, v$tablespace ts where tmp.ts#=ts.ts#;		PRI list of tempfiles		FO	
			select tmp.name filename, bytes, ts.name tablespace from v$tempfile tmp, v$tablespace ts where tmp.ts#=ts.ts#;	PSB list of tempfiles		Manual Action/Troubleshooting	Log message and continue with SO. No need for compare if it takes more than 5 seconds to compare.
9	Verify primary and standby disk location	select name from v$tempfile where substr(name,1,1)!='+';		no rows selected		FO	
			select name from v$tempfile where substr(name,1,1)!='+';	no rows selected		Manual Action/Troubleshooting	Log message and continue with SO. No need for compare if it takes more than 5 seconds to compare.
10	Verify all datafiles are ONLINE	select name from v$datafile where status='OFFLINE';		no rows selected		FO	
			select name from v$datafile where status='OFFLINE';	no rows selected		Manual Action/Troubleshooting	Log message and continue with SO. No need for compare if it takes more than 5 seconds to compare.

11	Verify primary and standby datafiles disk location	select name from v$datafile where substr(name,1,1)!='+';		no rows selected		FO	
			select name from v$datafile where substr(name,1,1)!='+';	no rows selected		Manual Action/Troubleshooting	Log message and continue with SO. No need for compare if it takes more than 5 seconds to compare.
12	Verify primary and standby online redo log disk location	select member from v$logfile where substr(member,1,1) not in ('+','M','N');		no rows selected		FO	
			select member from v$logfile where substr(member,1,1) not in ('+','M','N');	no rows selected		Manual Action/Troubleshooting	Log message and continue with SO. No need to compare if it takes more than 5 seconds to compare.
13	Verify Oracle Streams replication	select status, count(*) processes from dba_apply;		This is for troubleshooting purpose only. If query returns no rows, then it is ok. Mainly to identify if this DB is source or target.		FO	No Action needed. Just logging in logfile and at the end provide message to validate Streams if setup on this DB where SO is performed.

		select status, count(*) processes from dba_capture;		This is for troubleshooting purpose only. If query returns no rows, then it is ok. Mainly to identify if this DB is source or target.		FO	No Action needed. Just logging in logfile and at the end provide message to validate Streams if setup on this DB where SO is performed.
14	Suspend Scheduler jobs	select owner, job_name from dba_scheduler_running_jobs and owner <> 'SYS';		no rows selected		FO	Manual Action
		select owner, job_name, next_run_date, enabled from dba_scheduler_jobs where enabled='TRUE' and owner <> 'SYS' order by next_run_date;		list of scheduler jobs		FO	Manual Action
15	Check for potential long running operation	select s.inst_id, o.sid, client_info ch, context, sofar, totalwork, round(sofar/totalwork*100,2) "% complete" from gv$session_longops o, gv$session s where o.sid=s.sid and totalwork != 0		no row selected		FO	Manual Action - Pre-work

		and sofar <> totalwork and opname not like '%aggregate%'					
16	Suspend GC backup jobs	select job_id, job_name, job_type, scheduled_time, execution_id, status from sysman.mgmt$job_execution_history@to_emrep where upper (target_name) like upper ((select substr (db_unique_name,1, length (db_unique_name)-2) from v$database) || '%') and status in ('Scheduled');		4 rows expected: BCKP_<APP>_<SITE>_L0, BCKP_<APP>_<SITE>_L1, BCKP_<APP>_<SITE>_MONTHLY, PURGE_ARCH_LOG_<APP>_<SITE>		FO	Manual Action - Pre-work
17	Clear potential blocking parameters	show parameter job_queue_processes		<value>		FO	Log message and continue with SO.
			show parameter job_queue_processes	<value>		Manual Action/Troubleshooting	Log message and continue with SO.

18	Perform switchover		switchover to <PSB-DB_UNIQUE_NAME>;			Manual Action/Troubleshooting	
19	Resume scheduler jobs	execute dbms_scheduler.enable('<owner>.<job_name>');		User to take Manual Action.		Manual Action/Troubleshooting	
20	Resume GC jobs	select job_name, job_id, target_name from sysman.mgmt$job_targets@to_emrep where upper (target_name) like upper ((select substr (db_unique_name,1, length (db_unique_name)-2) from v$database) || '%')		User to take Manual Action. 4 rows expected: BCKP_<APP>_<SITE>_L0, BCKP_<APP>_<SITE>_L1, BCKP_<APP>_<SITE>_MONTHLY, PURGE_ARCH_LOG_<APP>_<SITE>		Manual Action/Troubleshooting	

The table 5.2 elaborate the integration and unit testing conducted for the ODaRT tool. The test is known to user and the result of the tests are exported to SPSS software to produce a graphical presentation. The presentation are based on t-test data and stem-leaf plot which elaborated in the next section of result. We have conducted the white box text with the selected sample of respondents. The test case scenarios are based on the circumstances of for a planned activity or the database goes unhealthy. The result of the test cases are provided in the Appendix B. The combination of the primary and standby states define the preferred and alternated role transition. For an instance, the test case scenario TS1 explains during a data base rolling upgrade or patch a switchover is preferred role transition and failover as an alternate option. There should be no data loss expected and the system should recovered within two (2) minutes.

Table 5.2 ODaRT white box test cases

Sl#	Scenario	Primary State	PSB DB	Planned Activity with Quiet time or no Quiet time?	Preferred option to perform Role change?	Alternate option to perform Role change?	Data Loss?	System recovery <2mins?	Reinstate Original-Pri as Standby Automatically?	Comment
TC#	***Sample***	***Healthy/Unhealthy***	***Healthy/Unhealthy***	***DT/No-DT***	***Switchover/Failover***	***Switchover/Failover***	***Yes/No/MayBe***	***Yes/No/May be***	***Yes/No/Maybe***	
TS1	Rolling Upgrade	Healthy	Healthy	No-DT	Switchover	Failover (Shutdown Abort)	No	Yes	NA	Any upgrades without downtime.
TS2	Rolling Upgrade	Unhealthy	Healthy	No-DT	Failover	NA	No	Yes	Maybe	
TS3	Rolling Upgrade	Healthy	Unhealthy	No-DT	NA	NA	NA	NA	NA	Fix PSB
TS4	DB/Node Sick	Healthy	Unhealthy	No-DT	NA	NA	NA	NA	NA	Fix PSB
TS5	DB/Node Sick	Unhealthy	Healthy	No-DT	Failover	NA	No	Yes	Maybe	
TS6	DB/Node Sick	Unhealthy	Unhealthy	No-DT	NA	NA	NA	NA	NA	True Disaster Recovery Scenario
TS7	DB/Node	Healthy	Healthy	DT	Switchover	Failover (Shutdown Abort)	No	NA	NA	

*Healthy – Primary and standby in synch/ stack working as expected

*Unhealthy – Primary and standby out of synch/Sick

*DT - Downtime

5.3 Result

The ODaRT tool has been tested and presented the result with both the functionality test by the experts and the tool efficiency measurement. Two criteria are defined for finding the tools efficiently by the expert's survey and with test cases. The criteria for expert's observation are set as prerequisite and post tasks time taken in minutes and the ratio of successful switchover.

Meanwhile, the criteria for tool test cases are switchover time in seconds and failure of role transitions.

5.3.1 Expert's Observation

Table 5.3 Result for expert's observation

One-Sample Test

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Respondant	4.899	6	.003	4.000	2.00	6.00
Pre-Post Tasks Time Taken (Min)	16.460	6	.000	12.14286	10.3377	13.9480
Ratio of successful	41.243	6	.000	9.00000	8.4660	9.5340

Table 5.3 data shows that, the average minutes taken for a responded to perform pre and post tasks are 16.4 minutes meanwhile the ratio of successful is 9 out of 10 execution with only a single failure.

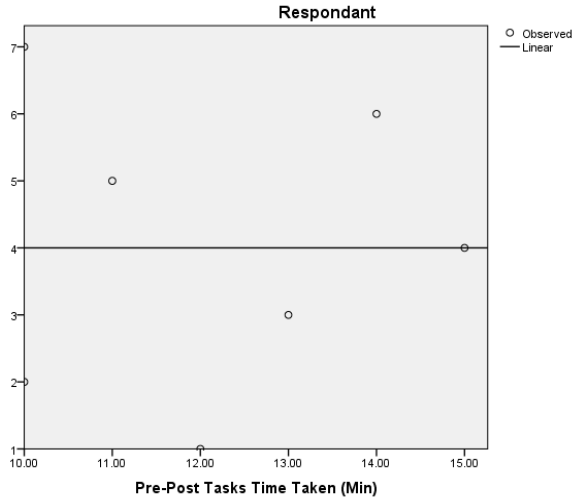

Figure 5.14 Linear Chart of Pre-Post Tasks Time (Minutes)

Figure 5.14 is the linear chart that shows the observation of pre and post tasks time taken in minutes by the respondents (Experts). The chart clearly shows that the observed circle is closer to the value of ~15minutes crossed by the linear line.

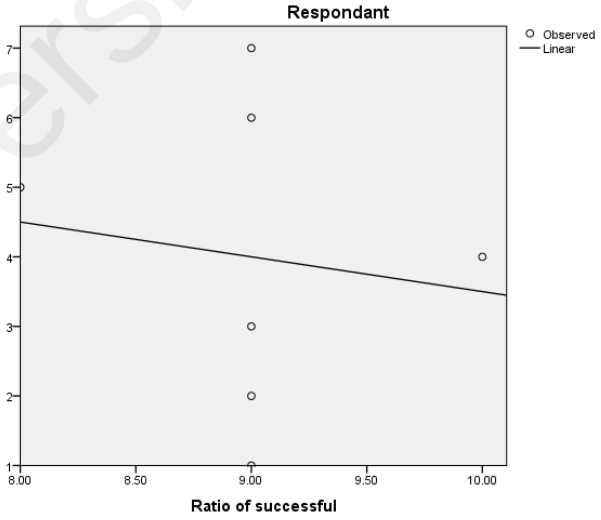

Figure 5.15 Linear Chart of Ratio of Successful Switchover

The linear chart in Figure 5.15 shows the ration of successful switchover out of 10 execution. The results shows the observed circle of value ~10 is closer to the liner line. This indicate that the average successful of execution is almost 100%.

5.3.2 ODaRT Tool Observation

Table 5.4 Test cases result for tool observation

One-Sample Test

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Test Cases	5.196	7	.001	4.500	2.45	6.55
Swith Over time (Seconds)	7.878	7	.000	53.12500	37.1783	69.0717
Failure of RT	5.292	7	.001	1.00000	.5531	1.4469

Table 5.4 data shows that, the test cases result for the tool execution results. It shows average it takes 53 seconds to complete the switchover. There are only one failure happen out of 5 role transition.

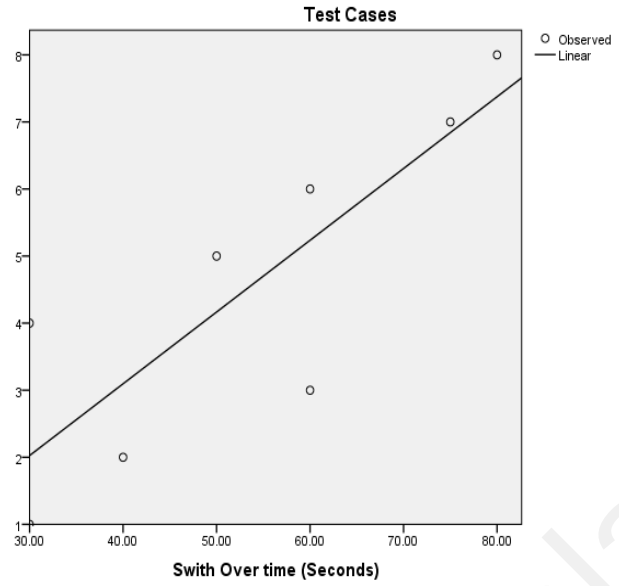

Figure 5.16 Linear Chart of Switchover Time (Minutes)

The linear chart in Figure 5.16 shows the switchover time more leaning to seconds between 60-75 seconds.

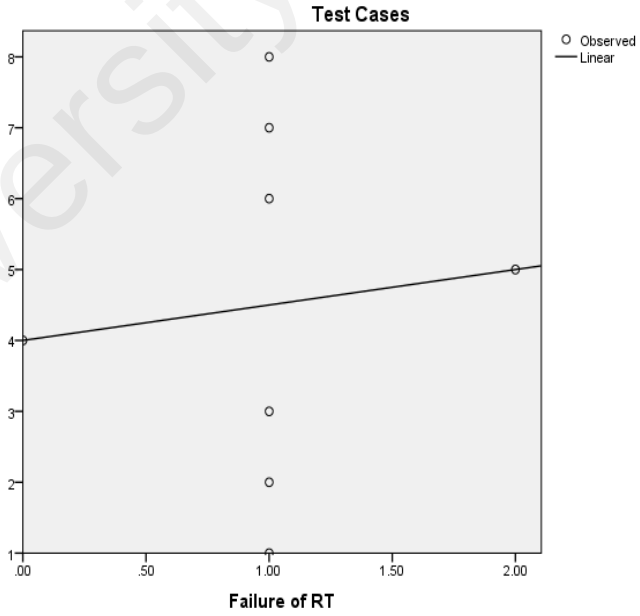

Figure 5.17 Linear Chart of Failure of Role Transition

The Figure 5.17 shows the test cases conduct to observe the frequency of role transition failure using the ODaRT tool. And it shows almost 0(1) cases for conducted 5 test cases.

5.3.3 Comparison Result

To prove the ODaRT have successfully achieve the objective, the test result were compared between both the traditional method of only Data Broker and with the automated custom tool ODaRT.

Table 5.5 Data Comparison of Traditional Method and ODaRT

One-Sample Test

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Traditional Method (DataBroker)	8.885	6	.000	32.71429	23.7045	41.7240
ODaRT	23.904	6	.000	21.14286	18.9786	23.3071

The table 5.5 shows the test cases result comparison between the Traditional method of using Data Broker and the ODaRT tool. The ODaRT tool have average of 21 seconds of switchover time meanwhile the traditional method took more in average 32 seconds. The maximum seconds taken for traditional method is 41 seconds whereas the ODaRT is just 23 seconds. This prove that the ODaRT tool give better efficiency than the traditional method in term of time taken to complete a switchover.

5.4 Evaluation

The table 5.6 describe the comparison between the Oracle Data Guard database role transitions comparison. The comparison done between the third-party tool, oracle provided Data Guard Broker and the developed tool ODaRT. The DBvisit standby and DG Toolkit are the tools available in market to perform role transitions. Oracle Data Guard Broker by the Oracle itself is the most popular tool used to perform a switchover.

Table 5.6 Oracle Data Guard database role transition tools evaluation

Tools Feature	**1. Dbvisit Standby**	**2. DG Toolkit**	**3. Oracle Data Guard Broker**	**4. ODaRT**
Oracle Best Known Practice			☆	☆
Automated Check				☆
Perform database switchover	☆	☆	☆	☆
Perform database health check				☆
Support varied operating system	☆		☆	
Switchover time < 2mintues			☆	☆

But now with the comparison from the results, it's proven the customized ODaRT tool give more efficiently in performing a role transition. Most importantly the ODaRT tool has incorporated the Oracle best known practice and automated each of the checks. The automated check really help to streamline the process and time taken to perform role transition by database administrators. Only the ODaRT tool is providing the feature of database health check based on oracle best known practices. Only the ODaRT tool support an automated feature to perform prerequisites check, it means the database administrators don't need to remember and run any SQL queries or DGMGRL commands.

5.6 Summary

The ODaRT's simple and easy execution enables a database administrator to perform Oracle Data Guard database role transition efficieantly. In this chapter we have covered the whole process of ODaRT tool implementation, the process been illustrated and outlined with recognized core capabilities of the tool and with the supported technology.

The robustness of the tool is improved with the detailed white and black box testing. The testing has made the tool functions more accurately without bugs or errors. The comparative results have proven the tool has achieve its objective with taking a lesser seconds to complete a role transition compare to the traditional Oracle Data Broker method. Finally the evaluation of the comparison result has made the participant from the expert database administrators sensed that this automated tool is excellent and superb in term of the capabilities to do health check of a database and perform pre and post verification check before the role transition, switchover.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

In today's world, efficiency and streamlining processes are much anticipated in performing any task. There are lots of scope in streamlining the process involved database role change. The ODaRT tool have successfully identified the incurring problems in Oracle Data Guard role transition and provided a solution that given an excellent result.

In this research, we have accessed the role transitions performed by database administrators in one of the complex database system, Oracle Data Guard. The Oracle Data Guard systems are widely used in the sector of manufacturing, finance and human resource, etc since it's ensure the enterprise data are protected, highly available and recoverable for a disaster. The Oracle Data Guard is mostly favors multiple types of systems such as Data Warehouse, Online Transaction Processing and Decision Support Systems. The Oracle Data Guard offload the resource intensive backups to standby database to improve the productivity of the primary database. Since planned and unplanned outage is very common for Data Guard database, the efficiency of the role transitions is important to make sure the availability of database is maintained and minimized the downtime.

With the accessed database role transition approach using the Oracle best known practice in the ODaRT tool, we have demonstrated the role transition can be performed efficiently and effectively by database administrators without failures. The research study will be supportive for the Oracle Corporation in enhancing their Oracle Data Guard product and probably introduce the research contribution in their future releases.

6.1 Strengths of ODaRT tool

After the assessment of implementation and evaluation process done in previous chapter, we have successfully achieved the following strengths for ODaRT tool.

- Automated role transition

The automated feature enable the database administrator to perform the role transition without referring to any commands or Oracle best known practice, the tool is do it for them. The process of health check, pre check and role transition are streamlined and automated as much as possible.

- A error-free role transition is achievable

The tool proved there is no longer failure due to human error since the tool is automated to perform the role transition and able to prepare both the primary and standby database before the role transition. The industries will be beneficial in reducing the cost of downtime.

- Speed

With the ability to perform health check and prerequisites check, the database administrator can perform the switchover without spending time on checking the health and prepare the database to do role transition.

- Performance

The ODaRT have no performance issue since it is not a resource intensive tool, it use no GUI which will usually slows the performance. The tool is developed with simple Perl program gives excellent reliability.

6.2 Limitation of ODaRT tool

Although we achieved the objective of this research, the ODaRT tool still give some room for improvements. The ODaRT tool is not developed for the use of multi operating systems. Currently the ODaRT tool only can be run on Oracle Data Guard which installed in Microsoft Windows Servers. Although the execution of the tool is very simple, but it slightly out of current generation trend where mostly the applications or tools are based on web or multimedia with a nice Graphical User Interface.

6.3 Future Work

Oracle releases its new technology rapidly, while writing this research paper there is already Oracle Database new version of Oracle 12c is announced. The research study should expand its boundary till the new the newer version of Oracle Database 12c. We should also focused on expanding the development of this tool into multiple platform especially Linux which is Oracle's preferred platform. We have also received feedback for the tools human computer interactions in term of user interface. Some of the participants suggested to have the tool in Graphical User Interface with some buttons capabilities. In term of functionality, there are chances to enhance it with more capabilities like autonomous troubleshooting for an error in database and etc.

REFERENCES

11.2 Data Guard Physical Standby Switchover Best Practices using SQL*Plus. . Retrieved from https://support.oracle.com/epmos/faces/DocumentDisplay?id=1304939.1

11.2 Data Guard Physical Standby Switchover Best Practices using the Broker. Retrieved from https://support.oracle.com/epmos/faces/DocumentDisplay?id=1305019.1

Alapati, S., Kuhn, D., & Nanda, A. (2007). *RMAN Recipes for Oracle Database 11g: A Problem-Solution Approach*: Apress.

Alapati, S. R., Kuhn, D., & Padfield, B. (2011). *Oracle Database 11g performance tuning recipes : a problem-solution approach*: Apress.

Ashdown, L., Kyte, T., Creighton, J., Engsig, B., Fogel, S., Habeck, B., . . . Huey, P. (2011). Oracle® Database Concepts 11g Release 2 (11.2).

Bryla, B. (2007). *Oracle database 11g DBA handbook*: Tata McGraw-Hill Education.

Burleson, D. (2015). Partitioning an Oracle table Tips. Retrieved from http://www.dba-oracle.com/t_partitioning_tables.htm

Carpenter, L. (2009). *Oracle data guard 11g handbook*: Oracle Press/McGraw-Hill.

Carpenter, L., Meeks, J., Kim, C., Burke, B., Carothers, S., Kundu, J., . . . Vengurlekar, N. (2009). *Oracle data guard 11g handbook*: McGraw-Hill, Inc.

Charles, K. (2014). Automate Data Guard Best Practices. Retrieved from http://www.oracle.com/technetwork/database/features/availability/fnf-casestudy- 082608.html

Chaudhuri, S., & Weikum, G. (2000, 2000 / 01 / 01 /). *Rethinking database system architecture: Towards a self-tuning RISC-style database system.* Paper presented at the 26th International Conference on Very Large Data Bases, Cairo, Egypt.

Crosby, N., Hughes, C., Lizieri, C., & Oughton, M. J. J. o. p. r. (2005). A Message from the Oracle: the Land Use Impact of a Major In-town Shopping Centre on Local Retailing. *22*(2-3), 245-265.

Fast-Start Failover Best Practices. (2010). Oracle Data Guard 10g Release 2, Oracle MAA White Paper.

Greenwald, R., Stackowiak, R., & Stern, J. (2013). *Oracle essentials: Oracle database 12c*: " O'Reilly Media, Inc.".

Hayes, S., Rinkevich, D., & Lowrey, B. (2007). Auditing database end user activity in one to multi-tier web application and local environments. In: Google Patents.

High Availability Data Guard Broker. Page 17 of 21. Retrieved from https://docs.oracle.com/cd/B28359_01/server.111/b28295/troubleshooting.htm#

Jim, C. (2009). *Performing Database Failover with Oracle 11g Data Guard*

Kuhn, D., Alapati, S. R., & Padfield, B. (2016). *Expert Oracle Indexing and Access Paths: Maximum Performance for Your Database*: Apress.

Kumar, S. (2003). Oracle database 10g: The self-managing database. In: November.

Kyte, T. (2010). *Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g Programming Techniques and Solutions*: Apress.

Larry, M. C. (2011). *Oracle OpenWorld Active Data Guard Hands on Lab.*

Metalink. (n.d.). How to use Oracle Expert. Retrieved from www.metalink.oracle.com

Nawaz, R., & Soomro, T. R. (2013). Role of Oracle Active Data Guard in High Availability Database Operations. *International Journal of Applied Information System, Foundation of Computer Science, New York, USA, Volume 5(No. 5)* doi:978-0- 07-162148-9

Oracle10g: Data Guard Switchover and Failover Best Practices. Retrieved from https://support.oracle.com/epmos/faces/DocumentDisplay?id=387266.1

Oracle-Base. (2013). Retrieved from http://www.oraclebase.com/articles/11g/dataguard-setup-11gr2.php

Oracle 9i Database Manageability. Retrieved from http://www.oracle.com/technology/products/manageability/database/pdf/Oracle9iManageabilityBWP.pdf

Oracle Active Data Guard. (2010). Oracle Data Guard 11g Release 1, Oracle MAA White Paper.

Oracle® Database VLDB and Partitioning Guide 11g Release 2 (11.2). oracle.com. (2015). Retrieved from http://docs.oracle.com/cd/E18283_01/server.112/e16541/intro.htm

Palinkas, L. A., Horwitz, S. M., Green, C. A., Wisdom, J. P., Duan, N., Hoagwood, K. J. A., . . . Research, M. H. S. (2015). Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. *42*(5), 533-544.

Partitioning Concepts. oracle.com. (2015). Retrieved from http: //docs.oracle.com/cd/B28359_01/server.111/b32024/partition.html

Patton, M. Q. (1990). *Qualitative evaluation and research methods*: SAGE Publications, inc.

Singh, G. (2013). Maintaining Client Connectivity And Zero Failover Using Oracle Dataguard Grid Computing. *Journal of Global Research in Computer Science (JGRCS), 4*(7), 5-9.

Tuomas, N. (2010). Combining high-availability and disaster recovery: Implementing Oracle Maximum Availability Architecture (MAA) on Oracle 10gR2 RDBMS. 24.

Wynekoop, J. L. (1991). A review of computer aided software engineering research methods. *Information Systems Research: Contemporary approaches and emergent traditions*, 129-154.