BAB IV

HASIL DAN PEMBAHASAN

4.1. Analisa Permasalahan

Proses bisnis utama perusahaan adalah kegiatan bongkar muat petikemas yang ditangani oleh dua *database* utama yaitu *terminal operation system* (tops) dan *terminal operation backup* (topb), kedua *database* ini terhubung dalam konfigurasi dataguard Oracle 9i yang digunakan perusahaan sebagai solusi *high availability database* dimana jika terjadi *failure* atau diperlukan *maintenance* pada *database* utama tops, *database* topb dapat menggantikan peran dari *database* utama sehingga diharapkan dapat menjamin keberlangsungan proses bisnis perusahaan yang melayani transaksi operasional 24 jam setiap harinya. Permasalahan yang dihadapi adalah jika terjadi *failure* pada tops seorang DBA harus mengaktifkan *database* topb secara *manual* untuk menggantikan peran tops, hal ini menyebabkan waktu *downtime* tidak dapat dihindari dikarenakan seorang dba tidak bisa 24 jam *onsite* memantau *database* secara terus menerus.

Selain itu sistem *database* yang berjalan saat ini telah *running* sejak tahun 2002 dimana perusahaan beresiko mengalami gangguan availabilitas data yang disebabkan oleh kerusakan *database* secara *logical* (*logical block corrupt*), kerusakan secara *logical* ini menyebabkan *user* tidak dapat melakukan akses terhadap suatu data yang terletak pada *block corrupt* tersebut, pada Oracle 9i untuk memperbaiki kerusakan data ini dapat dilakukan dengan mengganti *block corrupt* tersebut dengan

block data yang masih baik yang dapat diambil dari *backup* yang ada. Jika tidak dilakukan *proactive check* pada *database* menyebabkan *block corrupt* tersebut akan ikut terbackup dan hasil *backup* yang baru akan *replace* backup yang lama. Sehingga tidak dapat dilakukan *recovery* pada data yang rusak tersebut karena versi *block* data yang masih baik telah *override* dengan hasil *backup* yang baru.

Avalailabilitas data ini sangat diperlukan dikarenakan di dalam perusahaan terdapat beberapa *database* yang saling terhubung dan terintegrasi (*distributed database*), dimana jika terjadi kerusakan data pada salah satu *database* dapat menyebabkan ketidakkonsistenan data. ketidakkonsistenan ini dapat mempengaruhi laporan data transaksi yang tidak *balance.*

4.2. Implementasi dan Uji Coba Sistem

4.2.1 Konfigurasi Oracle Dataguard

Physical standby database bekerja dengan cara melakukan sinkronisasi *redo* yang di-*generate* oleh *primary database* melalui *media recover*. Sedangkan *redo* dikirimkan melalui *protocol* Oracle net. Dengan menggunakan *media recovery* tersebut dapat dipastikan bahwa *physical standby* akan melakukan proses penyalinan *redo* data secara *block-per-block* dari *primary* ke *standby database*. Sehingga setiap transaksi yang dilakukan pada *primary database* akan selalu tercatat pada *standby database* oleh karena itu keadaan data antara kedua *database* selalu sama.

Standby database dapat dibuat dalam satu komputer *server* yang sama atau dapat menggunakan dua komputer *server* yang berbeda. Yang perlu diperhatikan adalah masing-masing *instance* harus mempunyai DB_UNIQUE_NAME yang

berbeda dikarenakan *physical file* pada *standby database* berasal dari salinan *primary database*. Berikut adalah *environment* yang akan digunakan untuk konfigurasi *dataguard.*

Tabel 4.1 *Environment* Dataguard

No	Environment	Server	
		Production	Standby
1.	IP Address	172.19.152.30	172.19.152.40
2.	Hostname	primary	standby
3.	DB_UNIQUE_NAME	prod	prodb
4.	Service Name	prod	prodb

4.2.1.1 Cek Persyaratan Dataguard

Persyaratan untuk membuat dataguard antara lain versi perangkat lunak Oracle harus sama antara *primary* dan *standby database* dimana dalam implementasi ini menggunakan versi 11.2.0.1 sedangkan untuk sistem operasi juga harus sama tetapi versi dari sistem operasi yang digunakan boleh berbeda. Selain itu di tiap komputer harus memiliki nama *database* / SID yang sama tetapi yang membedakan nanti adalah parameter dari DB_UNIQUE_NAME, untuk membuat *database* baru dapat menggunakan *tool* dari Oracle sendiri yaitu *Database Configuration Assistant* (DBCA).

4.2.1.2 Membuat TNS Alias

Sebelum dilakukan penambahan TNS, pastikan kedua *server* dapat terhubung dalam suatu jaringan :

Dari komputer server primary

```
oracle@primary$ ping 172.19.152.30
```

Dari komputer server standby

```
oracle@standby$ ping 172.19.152.40
```

Pada Unix dan Linux tambahkan line berikut pada /etc/hosts di kedua *server* :

```
172.19.152.30 primary

172.19.152.40 standby
```

Buat TNS Alias pada kedua komputer tersebut karena Oracle akan mengirimkan *log file* (*redo*) melalui protocol Oracle net, untuk menambahkan TNS Alias dapat dilakukan modifikasi pada file TNSNAMES.ORA yang terletak pada $ORACLE_HOME/network/admin

```
PROD  =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = primary)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PROD)
    )
  )


 PRODB =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = standby)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PRODB)
    )
  )
```

Dari komputer *server* standby

```
oracle@standby$ tnsping prod
```

```
oracle@standby:~$ tnsping prod

TNS Ping Utility for Solaris: Version 11.2.0.1.0 - Production on 01-JUL-2013 02:
31:03

Copyright (c) 1997, 2009, Oracle.  All rights reserved.

Used parameter files:


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = primary)
(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = prod)))
OK (20 msec)
```

Gambar 4.1 tnsping prod

Dari komputer *server* primary

```
oracle@primary$ tnsping prodb
```

```
oracle@primary:~$ tnsping prodb

TNS Ping Utility for Solaris: Version 11.2.0.1.0 - Production on 11-SEP-2013 13:
04:01

Copyright (c) 1997, 2009, Oracle.  All rights reserved.

Used parameter files:


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = standby)
(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = prodb)))
OK (30 msec)
```

Gambar 4.2 tnsping prodb

4.2.1.3 *Archivelog* dan *Flasback Mode*

Proses sinkronisasi yang dilakukan antara *primary* dan *standby database* adalah dengan mengirimkan catatan transaksi (*log file*) yang terdapat pada *redo log*, oleh karena itu *history* dari catatan transaksi dibutuhkan agar tidak terjadi *gap*

perbedaan data antar *database*. Pencatatan tersebut hanya dapat dilakukan oleh *database* dalam mode ARCHIVELOG. Dimulai dari versi 10g, Oracle memperkenalkan teknologi *flashback* yaitu mengembalikan keadaan *database* dalam kondisi waktu tertentu (*undo*) karena pada versi Oracle 9i, jika terjadi kegagalan pada *primary database*, seorang DBA harus mengkonfigurasi ulang *primary database* yang mengalami kegagalan tersebut. Dengan teknologi *flashback*, *primary database* yang mengalami kegagalan dapat di *recover* agar dapat di sinkronisasikan kembali dengan konfigurasi *dataguard*. Untuk memeriksa apakah *database* sudah dalam mode *archivelog* dan *flashback*.

```
SQL> select log_mode, flashback_on from v$database;
```

Gambar 4.3 Cek *Archivelog* & *Flashback*

Untuk menjadikan *database* dalam mode *archivelog* dan mengaktifkan *flashback*, lakukan perintah berikut :

```
SQL> conn sys/***** as sysdba
Connected
SQL> shutdown immediate
SQL> startup mount;
SQL> alter database archivelog;
SQL> alter database flashback on;
SQL> alter database open;
```

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  638853120 bytes
Fixed Size                  2213736 bytes
Variable Size             398461080 bytes
Database Buffers          234881024 bytes
Redo Buffers                3297280 bytes
Database mounted.
SQL> alter database archivelog;

Database altered.

SQL> alter database flashback on;

Database altered.

SQL> alter database open;

Database altered.
```

Gambar 4.4 Mode *Archivelog* & *Flashback*

4.2.1.4 Konfigurasi Parameter *File Primary Database*

Sebelum konfigurasi *physical standby database*, terdapat beberapa *parameter* yang perlu ditambahkan pada *primary database*. Buat *parameter file* / pfile agar lebih mudah menambahkan *parameter-parameter* yang dibutuhkan.

```
SQL> create pfile='/oracle/pfile_prod' from spfile;
```

Edit hasil *output* file pfile_prod dari perintah diatas menggunakan editor, kemudian tambahkan line berikut :

```
db_unique_name=prod
service_names=prod
log_archive_config='dg_config=(prod,prodb)'
log_archive_dest_1 = 'location=/oracle/oradata/prod/archive
valid_for=(ALL_LOGFILES,ALL_ROLES) db_unique_name=prod'
```

```
log_archive_dest_2 = 'service=prodb
valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_unique_name=prodb lgwr
async=20480'
log_archive_dest_state_1=ENABLE
log_archive_dest_state_2=DEFER
standby_file_management = AUTO
standby_archive_dest = ' location=/oracle/oradata/prodb/archive'
```

Parameter LOG_ARCHIVE_DEST_1 berfungsi untuk meletakkan file hasil *archivelog* pada direktori /oracle/oradata/primary/archive. Pastikan bahwa direktori tujuan ini *valid* sedangkan *parameter* STANDBY_ARCHIVE_DEST mempunyai fungsi yang sama dengan LOG_ARCHIVE_DEST_2 yaitu meletakan file *archivelog* yang dibuat oleh proses *Remote File Server (RFS)* dan *parameter* ini hanya berlaku pada *standby database*.

Simpan *file* pfile_prod tersebut dan juga salin ke host *standby* untuk memudahkan konfigurasi *parameter* pada *standby database*.

4.2.1.5 Konfigurasi *Primary Database*

Pembuatan *physical standby database* membutuhkan semua *datafiles*, *temporary files* serta *standby controlfile*. Oleh karena itu dibutuhkan salinan asli dari *primary database*, salinan asli tersebut dapat berupa hasil *backup* baik secara *cold* maupun menggunakan RMAN. Pada implementasi ini akan digunakan *cold backup*. Sebelumya lihat file-file *primary database* yang akan di salin ke *standby host*.

```
SQL> conn sys/***** as sysdba
Connected
SQL> select name from v$datafile;
```

Dikarenakan menggunakan *cold backup*, sebelum melakukan penyalinan file-file *primary database*, maka harus dilakukan *shutdown database* terlebih dahulu.

```
SQL> shutdown immediate
Database Closed.
Database Dismounted.
Oracle instance shut down.
```

Salin semua file-file dari *primary database* ke komputer *standby*, letakkan pada direktori sesuai konfigurasi pada komputer yang digunakan sebagai *physical standby* yaitu pada direktori /oracle/oradata/standby. Setelah proses penyalinan file selesai, *update* parameter file *primary database* sesuai yang telah di modifikasi pada bagian 4.1.3.

```
SQL> create spfile from pfile='/oracle/pfile_prod';
SQL> startup
```

Agar proses pengiriman *redo* data dari *primary* ke *physical standby database* dapat dilakukan berkelanjutan tanpa menunggu proses *log switch*, maka dibutuhkan *standby redolog* pada konfigurasi Oracle *dataguard* pada kedua *database* agar dapat mendukung jenis proteksi *maximum availability* yang nanti dibutuhkan untuk penerapan *fast-start-failover*. *Standby redolog* yang ditambahkan adalah sebanyak 3 group dengan masing-masing ukuran 100Mb.

```
SQL> alter database add standby logfile group 4
  2  ('/oracle/oradata/prod/stby_redo04.log') size 100M;

SQL> alter database add standby logfile group 5
  2  ('/oracle/oradata/prod/stby_redo05.log') size 100M;

SQL> alter database add standby logfile group 6
  2  ('/oracle/oradata/prod/stby_redo06.log') size 100M;
```

```
SQL> alter database add standby logfile group 4
  2  ('/oracle/oradata/prod/stby_redo04.log') size 100M;

Database altered.

SQL> alter database add standby logfile group 5
  2  ('/oracle/oradata/prod/stby_redo05.log') size 100M;

Database altered.

SQL> alter database add standby logfile group 6
  2  ('/oracle/oradata/prod/stby_redo06.log') size 100M;

Database altered.
```

Gambar 4.5 *Add Standby Redolog*

Standby redolog juga harus di implementasikan pada *physical standby* agar proses *switchover*, *failover* serta *fast-start failover* dari *primary* ke *standby* dapat berjalan dengan lancar. Proses konfigurasi pada *primary database* untuk *dataguard* telah selesai, oleh karena itu buat *standby controlfile* dari *primary database* yang akan digunakan pada *physical standby database*.

```
SQL> alter database create standby controlfile as
  2  '/oracle/stby_controlfile.ctl';
```

```
SQL> alter database create standby controlfile as
  2  '/oracle/stby_controlfile.ctl';

Database altered.
```

Gambar 4.6 *Create Standby Controlfile*

Salin file stby_controlfile.ctl tersebut ke komputer *standby* (172.19.152.40), letakkan pada direktori /oracle/oradata/standby sesuai dengan direktori yang digunakan. Kemudian salin *Password File* pwdprod.ora dari *primary database* yang

terletak pada *directory* $ORACLE_HOME/dbs/ ke komputer *standby* dengan lokasi yang sama lalu ubah nama *password* file tersebut menjadi pwdprodb.ora

4.2.1.6 Konfigurasi Parameter *File Physical Standby Database*

Update beberapa *parameter* file pfile_prod yang telah di salin dari *primary database* untuk digunakan pada *physical standby database*. *Parameter* yang utama adalah CONTROL_FILES karena *parameter* ini menentukan letak file-file fisik yang akan digunakan oleh *physical standby*.

```
control_files=’/oracle/oradata/prodb/stby_controlfile.ctl’
db_unique_name=prodb
service_names=prodb
log_archive_config='dg_config=(prod,prodb)'
log_archive_dest_1 = 'location=/oracle/oradata/prodb/archive
valid_for=(ALL_LOGFILES,ALL_ROLES) db_unique_name=prodb'
log_archive_dest_2 = 'service=prod
valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_unique_name=prod lgwr
async=20480'
log_archive_dest_state_1=ENABLE
log_archive_dest_state_2=ENABLE
standby_file_management = AUTO
standby_archive_dest = ' location=/oracle/oradata/prod/archive'
```

Simpan file hasil modifikasi untuk digunakan pada proses konfigurasi *physical standby database*.

4.2.1.7 Konfigurasi *Physical Standby Database*

Buat SPFILE dari PFILE pfile_prod yang telah di modifikasi sebelumnya dan dilanjutkan dengan *startup mount* pada *physical standby database* dan jangan pernah melakukan *Open Write mode* pada *standby database* karena akan berakibat pada ketidak-sinkronan kedua *database* di karenakan perbedaan nilai SCN (*System Change Number*). Jika hal ini terjadi maka yang bisa dilakukan untuk *recovery* sinkronisasi *database* adalah dengan melakukan flashback ke momen sebelum *database* dilakukan

Open Write mode atau kasus yang lebih buruknya lagi adalah membuat ulang *dataguard.*

```
SQL> create spfile from pfile='/oracle/pfile_prod';
SQL> startup
```

Seperti pada *primay database*, buat *standby redolog* agar *redo* data yang dikirimkan dari *primary* ke *physical standby database* dapat langsung di simpan ke dalam *standby redolog* secara *real time.*

```
SQL> alter database add standby logfile group 4
  2  ('/oracle/oradata/prodb/stby_redo04.log') size 100M;

SQL> alter database add standby logfile group 5
  2  ('/oracle/oradata/prodb/stby_redo05.log') size 100M;

SQL> alter database add standby logfile group 6
  2  ('/oracle/oradata/prodb/stby_redo05.log') size 100M;
```

```
SQL> alter database add standby logfile group 4
  2  ('/oracle/oradata/prodb/stby_redo04.log') size 100M;

Database altered.

SQL> alter database add standby logfile group 5
  2  ('/oracle/oradata/prodb/stby_redo05.log') size 100M;

Database altered.

SQL> alter database add standby logfile group 6
  2  ('/oracle/oradata/prodb/stby_redo06.log') size 100M;

Database altered.
```

Gambar 4.7 *Add Standby Redolog* prodb

4.2.1.8 Mengaktifkan *Physical Standby Database*

Agar *physical standby database* dapat menerima *redo* data untuk sinkronisasi data dengan *primary database*, maka perlu mengaktifkan MANAGED RECOVERY

PROCESS (MRP) sehingga keadaan data antara *primary* dan *physical standby database* akan selalu sama dengan menerapkan *redo* data yang di kirimkan oleh *primary* melalui proses LOG NETWORK SERVER (LNS). Supaya *redo* data dapat segera di proses tanpa menunggu *log switch*, jalankan MRP seperti berikut :

```
SQL> alter database recover managed standby database
using current logfile disconnect from session;
```

4.2.1.9 Konfigurasi *Dataguard Broker*

Dataguard broker dapat digunakan secara lokal maupun secara *remote*, sebelum menggunakan *dataguard broker* terlebih dahulu modifikasi *parameter* pada kedua *database* menjadi DG_BROKER_START=TRUE.

```
SQL> conn sys/*****@prod as sysdba
SQL> alter system set dg_broker_start=true scope=both;
SQL> conn sys/*****@prodb as sysdba
SQL> alter system set dg_broker_start=true scope=both;
```

```
SQL> conn sys/        @prod as sysdba
Connected.
SQL> alter system set dg_broker_start=true scope=both;

System altered.

SQL> conn sys/        @prodb as sysdba
Connected.
SQL> alter system set dg_broker_start=true scope=both;

System altered.
```

Gambar 4.8 *Dataguard Broker*

Kemudian modifikasi isi dari file LISTENER.ORA yang terletak pada direktori $ORACLE_HOME/network/admin pada kedua komputer agar Oracle *dataguard broker* dapat melakukan *startup* pada *database* secara otomatis.

a. *Database* Prod

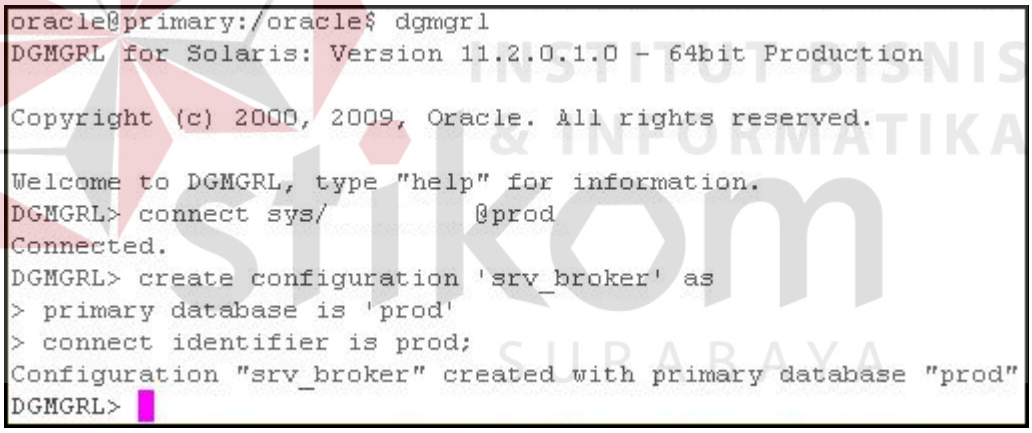
```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = prod)
      (ORACLE_HOME = /oracle/product/11.2.0/db_1)
      (SID_NAME = prod)
    )
   (SID_DESC =
      (GLOBAL_DBNAME = prod_DGMGRL)
      (ORACLE_HOME = /oracle/product/11.2.0/db_1)
      (SID_NAME = prod)
    )
  )
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP)(HOST = primary)(PORT = 1521))
    )
  )
```

b. *Database* prodb

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = prodb)
      (ORACLE_HOME = /oracle/product/11.2.0/db_1)
      (SID_NAME = prodb)
    )
   (SID_DESC =
      (GLOBAL_DBNAME = prodb_DGMGRL)
      (ORACLE_HOME = /oracle/product/11.2.0/db_1)
      (SID_NAME = prodb)
    )
  )
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP)(HOST = standby)(PORT = 1521))
    )
  )
```

Setelah kedua file LISTENER.ORA di atas dimodifikasi , lakukan *restart* pada listener agar efek dari modifikasi file diatas dapat berjalan. Untuk membuat konfigurasi pada *dataguard broker*, *login* terlebih dahulu pada *primary database* kemudian buat konfigurasi *dataguard broker* yang nantinya akan mencatat informasi dari *primary* dan *physical standby database*.

```
oracle@primary$ dgmgrl

DGMGRL> connect sys/******@prod

DGMGRL> create configuration 'srv_broker' as

>primary database is 'prod'

>connect identifier is prod;
```



Gambar 4.9 *Create Configuration Broker*

SRV_BROKER adalah nama konfigurasi *dataguard broker* dan PROD adalah nama *primary database* yaitu nama DB_UNIQUE_NAME pada INIT.ORA sedangkan *connect identifier* menunjukkan koneksi ke *database* menggunakan TNS alias. Berikutnya tambahkan konfigurasi untuk informasi mengenai *physical standby database.*

```
DGMGRL> add database 'prodb' as
> connect identifier is prodb
> maintained as physical;
```

```
DGMGRL> add database 'prodb' as
> connect identifier is 'prodb'
> maintained as physical;
Database "prodb" added
```

Gambar 4.10 *Add Database Broker*

```
DGMGRL> enable configuration;
```

4.2.1.10 *Protection Mode*

Tipe *protection mode* yang akan di implementasikan adalah *maximum availability*, pemilihan tipe proteksi ini berdasarkan pernyataan sesuai landasan teori yang ada dimana *maximum availability* akan menjamin tidak ada data yang hilang dan transaksi tidak akan di-*commit* pada *primary database* sampai telah dipastikan bahwa data transaksi tersedia juga pada *standby database*. *Maximum availability* juga menjaga *primary database* tetap tersedia apabila sewaktu-waktu *standby database* mengalami gangguan dan tipe proteksi ini juga dibutuhkan untuk penerapan *Fast-start failover.*

Modifikasi mode proteksi menjadi *Maximum Availability*

```
DGMGRL> Edit configuration set protection mode as
> maxavailability;
```

```
oracle@primary:~$ dgmgrl
DGMGRL for Solaris: Version 11.2.0.1.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys/          @prod
Connected.
DGMGRL> edit configuration set protection mode as MaxAvailability;
Succeeded.
```

Gambar 4.11 *Protection Mode*

4.2.1.11 *Fast-start Failover*

Fast-start failover dapat di-*enable* dari mana saja yang terhubung dengan konfigurasi *dataguard broker*, termasuk tempat *observer*. Dengan *fast-start failover* memungkinkan *observer* meninjau *primary* dan *physical standby database* dan memulai perubahan peran secara otomatis antara *primary* dan *standby database* saat terjadi kerusakan pada *primary* dimana proses perubahan peran ini tidak memerlukan intervensi *Database Administrator*.

Pastikan *property* LOGXPTMODE bernilai SYNC pada *primary* dan *standby database*

```
DGMGRL> edit database 'prod' set property
LogXptMode='SYNC';
DGMGRL> edit database 'prodb' set property
LogXptMode='SYNC'
```

```
DGMGRL> edit database 'prod'
> set property LogXptMode='SYNC';
Property "logxptmode" updated
DGMGRL> edit database 'prodb'
> set property LogXptMode='SYNC';
Property "logxptmode" updated
DGMGRL>
```

Gambar 4.12 Edit logxptmode

Tentukan *property* FASTSTARTFAILOVERTARGET pada *primary* dan *standby database* untuk saling mengarahkan target ketika terjadi perubahan peran.

```
DGMGRL> edit database 'prod' set property
> FastStartFailoverTarget = 'prodb';
DGMGRL> edit database 'prodb' set property
> FastStartFailoverTarget='prod';
```

```
DGMGRL> edit database 'prod' set property faststartfailovertarget='prodb';
Property "faststartfailovertarget" updated
DGMGRL> edit database 'prodb' set property faststartfailovertarget='prod';
Property "faststartfailovertarget" updated
```

Gambar 4.13 Edit FSFO Target

Enable fast-start failover dapat dilakukan saat sedang terhubung dengan konfigurasi *broker*

```
DGMGRL> enable fast_start failover;
```

Jalankan *observer* dengan DGMGRL, *observer* dapat dijalan dimana saja selama terdapat minimal installasi Oracle *client* dan terhubung dalam satu jaringan.

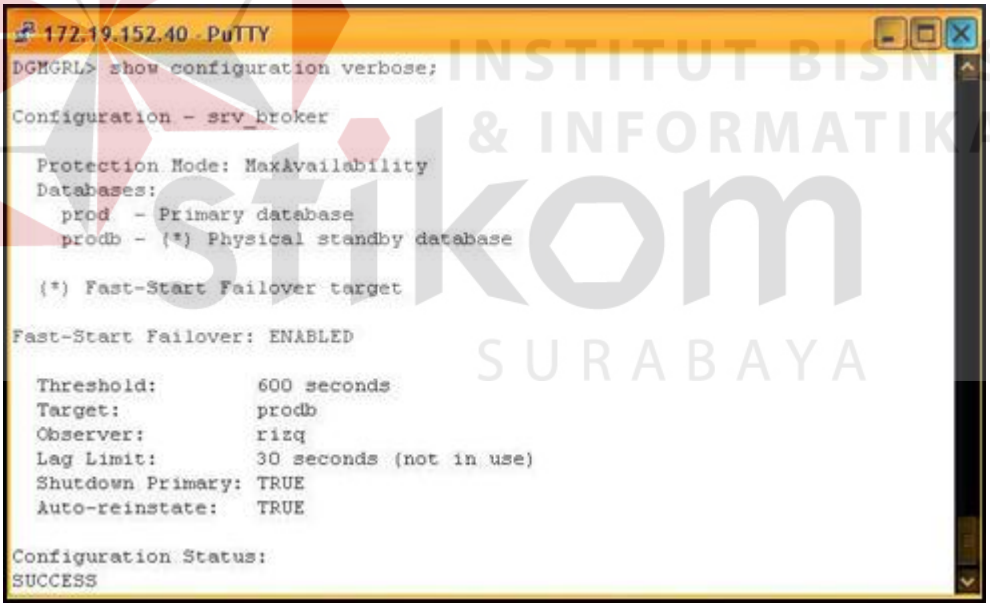
```
DGMGRL> connect sys/*****@prod
DGMGRL> start observer
```

4.2.2 Uji Coba Sistem

4.2.2.1 Reliability

Pengujian sistem *high availability* dilakukan dari kemampuan *reliability database* yang diharapkan dapat memenuhi kebutuhan akan sistem yang dapat mendukung transaksi operasional 24 jam dimana pada sistem sebelumnya jika terjadi kegagalan pada *primary database* membutuhkan intervensi seorang *Database Administrator* (DBA) untuk mengaktifkan peran *standby database* menjadi *primary*. Sebelum dilakukan pengujian, lakukan pengecekan konfigurasi *fast-start failover* dengan *broker*, pastikan statusnya adalah *Success* seperti pada gambar 4.14

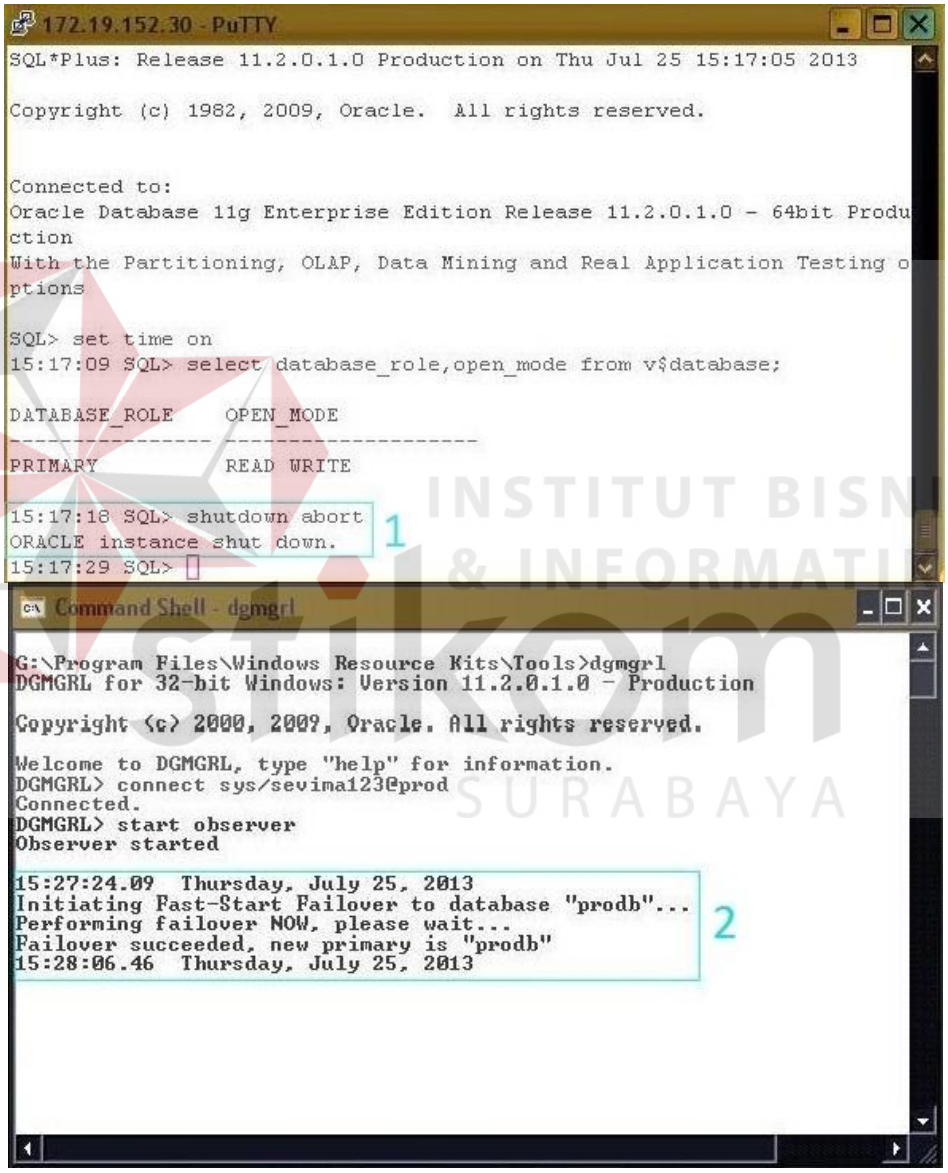
DGMGRL> show configuration verbose;



Gambar 4.14 Status FSFO

Pengujian dilakukan dengan mematikan *database primary* (prod) secara paksa dengan perintah SHUTDOWN ABORT yang akan menyebabkan *database*

mengalami *'hard crash',* pada saat ini *observer* akan mendeteksi adanya masalah pada *primary database* dan berusaha untuk menghubungkan lagi *(reconnect)* dalam jangka waktu yang telah diatur pada properti FASTSTARFAILOVERTHRESHOLD yaitu 10 menit.



Gambar 4.15 Proses *Fast-Start Failover*

Setelah melebihi waktu *threshold* 10 menit, *observer* akan melakukan proses *fast-start failover* dengan menginstruksikan *standby database* (prodb) untuk menggantikan peran dari *primary* seperti pada gambar 4.15. Proses ini berjalan secara otomatis dan tidak memerlukan intervensi seorang DBA seperti yang terjadi pada sistem sebelumnya dengan *database* Oracle 9i dimana jika terjadi kegagalan membutuhkan kehadiran seorang DBA untuk *onsite* ke lapangan yang berarti lamanya sistem *database* dapat kembali melayani transaksi bergantung dari availabilitas seorang DBA.

4.2.2.2 Recoverability

1. RMAN Data Recovery Advisor

Data Recover Advisor secara otomatis mendiagnosa kerusakan atau hilangnya konsistensi struktur dari lokasi *database file* serta menentukan dan memberikan rekomendasi opsi yang terbaik kepada *user* dalam melakukan *recovery* sehingga mengurangi waktu yang dibutuhkan untuk *recovery* ketika terjadi kegagalan

Pengujian dilakukan dengan merusak datafile *users*. Sebelumnya lakukan pengecekan lokasi *datafile users* tersebut.

```
SQL> select name from v$datafile;
```

```
SQL> select name from v$datafile;

NAME
----------------------------------------
/oracle/oradata/prod/system01.dbf
/oracle/oradata/prod/sysaux01.dbf
/oracle/oradata/prod/undotbs01.dbf
/oracle/oradata/prod/users01.dbf
```

Gambar 4.16 *List Datafile*

Simulasi perusakan *datafile* dilakukan menggunakan *command* "echo" dengan menuliskan *null string* pada *datafile users* sehingga seolah-olah *datafile users* tersebut menjadi sebuah text file, lakukan pengecekan ukuran file users01.dbf sebelum dan sesudah perintah "echo" seperti gambar 4.17

```
oracle@primary:~$ cd /oracle/oradata/prod/
oracle@primary:/oracle/oradata/prod$ ls -l users01.dbf
-rw-r-----   1 oracle   oinstall 5251072 Sep  5 11:35 users01.dbf
oracle@primary:/oracle/oradata/prod$ echo > users01.dbf
oracle@primary:/oracle/oradata/prod$ ls -l users01.dbf
-rw-r-----   1 oracle   oinstall       1 Sep  5 12:19 users01.dbf
```

Gambar 4.17 Simulasi *Datafile Users*

Lakukan pengaksesan terhadap *datafile users* dengan membuat sebuah *table* yang diletakkan pada *datafile* tersebut. Berdasarkan gambar 4.18, Oracle tidak dapat melakukan penulisan ke *datafile users* dikarenakan terjadi perubahan file (*convert*) yang disebabkan oleh command "echo" selain itu juga terjadi ketidak konsistenan *System Change Number* (SCN) *Header* antara *datafile users* dengan yang lainnya.

```
SQL> CREATE TABLE SCOTT.TABLE_1 TABLESPACE USERS AS SELECT * FROM ALL_OBJECTS;
CREATE TABLE SCOTT.TABLE_1 TABLESPACE USERS AS SELECT * FROM ALL_OBJECTS
                                                          *
ERROR at line 1:
ORA-01115: IO error reading block from file  (block # )
ORA-01110: data file 4: '/oracle/oradata/prod/users01.dbf'
ORA-27072: File I/O error
Additional information: 4
Additional information: 3
```

Gambar 4.18 *Error* Akses *Datafile Users*

List failure pada RMAN akan mendeteksi kegagalan ataupun kerusakan yang terjadi pada *database* pasca dilakukanya perusakan *datafile.* Oleh karena *datafile*

merupakan salah satu *physical file database* maka *priority* dari status kegagalannya adalah *HIGH* atau *mandatory* seperti pada gambar 4.19.

```
RMAN> list failure;

using target database control file instead of recovery catalog
List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
82         HIGH     OPEN      05-SEP-13     One or more non-system datafiles are corrupt
5          HIGH     OPEN      04-JUL-13     One or more non-system datafiles need media recovery
```

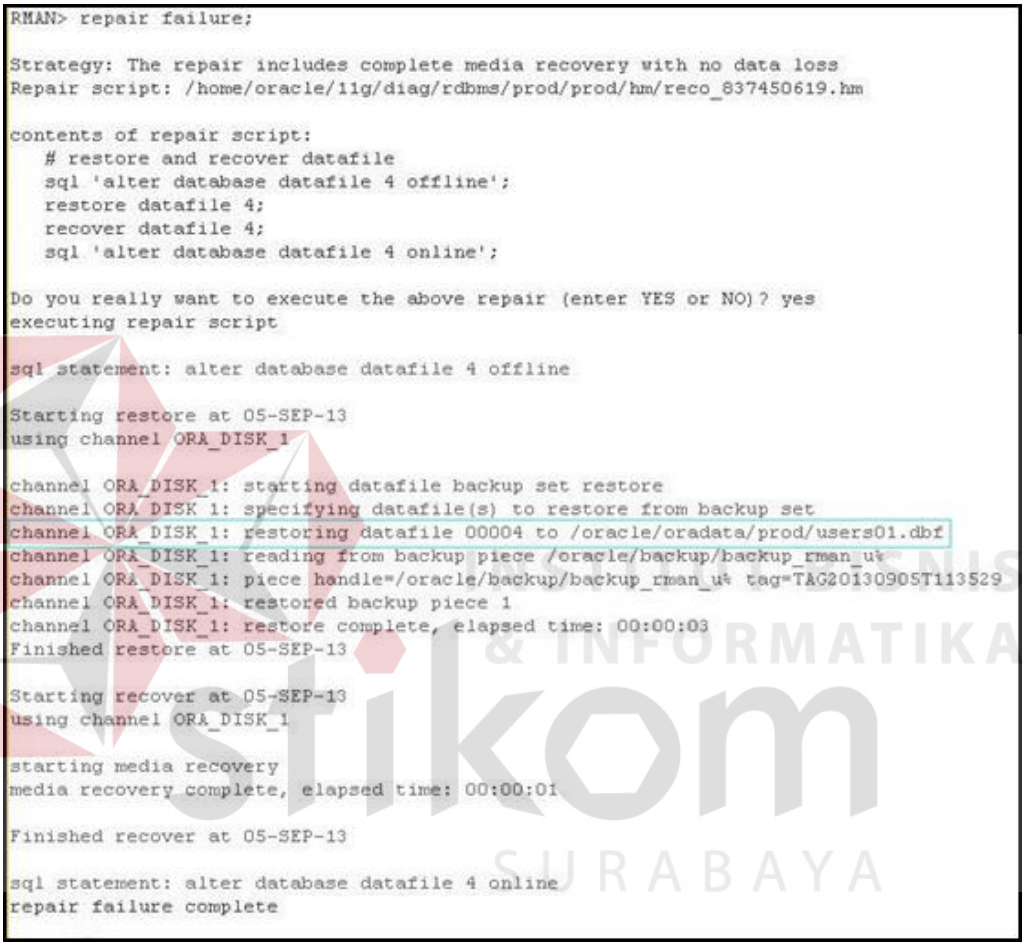
Gambar 4.19 RMAN *List Failure*

Advise failure RMAN menampilkan *list* rekomendasi perbaikan kerusakan pada *database* yang dihasilkan dari proses *List Failure* sebelumnya. Berdasarkan gambar 4.x terdapat hasil rekomendasi yang dapat dilakukan secara *manual* ataupun otomatis.

```
RMAN> advise failure;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
82         HIGH     OPEN      05-SEP-13     One or more non-system datafiles are corrupt
5          HIGH     OPEN      04-JUL-13     One or more non-system datafiles need media recovery

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=63 device type=DISK
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
========================
1. If you restored the wrong version of data file /home/oracle/11g/oradata/prod/users01.dbf, then
 replace it with the correct one
2. Perform a Data Guard role change (using PL/SQL routine DBMS_DG.INITIATE_FS_FAILOVER).

Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Restore and recover datafile 4
  Strategy: The repair includes complete media recovery with no data loss
  Repair script: /home/oracle/11g/diag/rdbms/prod/prod/hm/reco_837450619.hm
```

Gambar 4.20 RMAN *Advise Failure*

Repair failure RMAN akan mengeksekusi secara otomatis hasil rekomendasi dari proses yang dihasilkan oleh *Advise failure*. Proses ini akan menjalankan *script* yang ter-*generate* dari proses sebelumnya seperti pada gambar 4.20.



```
RMAN> repair failure;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /home/oracle/11g/diag/rdbms/prod/prod/hm/reco_837450619.hm

contents of repair script:
   # restore and recover datafile
   sql 'alter database datafile 4 offline';
   restore datafile 4;
   recover datafile 4;
   sql 'alter database datafile 4 online';

Do you really want to execute the above repair (enter YES or NO)? yes
executing repair script

sql statement: alter database datafile 4 offline

Starting restore at 05-SEP-13
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00004 to /oracle/oradata/prod/users01.dbf
channel ORA_DISK_1: reading from backup piece /oracle/backup/backup_rman_u%
channel ORA_DISK_1: piece handle=/oracle/backup/backup_rman_u% tag=TAG20130905T113529
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 05-SEP-13

Starting recover at 05-SEP-13
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:01

Finished recover at 05-SEP-13

sql statement: alter database datafile 4 online
repair failure complete
```

Gambar 4.21 RMAN *Repair Failure*

2. *Auto Block Media Recovery*

Kerusakan pada data (*data corrupt*) dapat terjadi pada setiap perangkat lunak DBMS, semakin besar *load* transaksi pada *database* berbanding lurus dengan kemungkinan terjadinya *data corrupt*, selain itu kejadian yang tidak terduga seperti listrik padam ataupun permasalahan pada *disk storage* juga dapat menyebabkan

kerusakan data.. *Data corrupt* biasanya disebabkan oleh *block corruption* dari sisi OS (*Operating System*) dimana terjadi kegagalan ketika proses penulisan ke dalam *disk*, jika hal ini terjadi *database* tetap dapat berjalan secara normal tetapi *user* tidak dapat membaca data pada *block* data yang terjadi kerusakan tersebut. Pada Oracle jika data yang rusak tersebut di akses maka akan terdapat peringatan pada *alert log.*

```
ORA-01578: Oracle data block corrupted (file # string, block #
string)
```

Sebelum Oracle 11g, seorang DBA harus melakukan *recovery* secara manual dari *backup* yang terdapat pada *Recovery Manager* (RMAN). Dan sejak Oracle 11g permasalahan ini dapat diatasi dengan adanya *standby database* yang terdapat pada konfigurasi *dataguard*, Oracle akan melakukan *recovery* dengan mengganti *block corrupt* tersebut dengan *block* yang terdapat pada *standby database*. Proses ini berjalan otomatis dan tidak diperlukan intervensi DBA dan dilakukan oleh proses *Auto Block Media Recovery* (Auto BMR).

Pengujian *Auto* BMR dapat dilakukan dengan *Active Dataguard* dimana *user* tetap dapat melakukan *query select* pada *standby database* dengan data yang *terupdate* secara *real-time* dengan *primary database.* Sebelum pengujian, lakukan perubahan mode pada *standby database* seperti gambar 4.22.

```
SQL> alter database recover managed standby database
cancel;
SQL> alter database open read only;
SQL> alter database recover managed standby database
using current logfile disconnect from session;
```

```
SQL> alter database recover managed standby database cancel;

Database altered.

SQL> alter database open read only;

Database altered.

SQL> alter database recover managed standby database using current logfile disco
nnect from session;

Database altered.
```

Gambar 4.22 *Active Dataguard*

Periksa apakah *standby database* sudah dalam mode *read-only*

```
SQL> select open_mode from v$database;
```

```
SQL> select open_mode from v$database;

OPEN_MODE
--------------------
READ ONLY WITH APPLY
```

Gambar 4.23 Pengecekan Status *Standby Database*

Pengujian akan dilakukan dengan *table* EMP yang dimiliki oleh *user* / *schema* SCOTT, oleh karena itu dibutuhkan informasi salah satu *block id* yang terdapat pada *table* EMP.

```
SQL> select min(dbms_rowid.rowid_block_number(rowid))
from scott.emp;
```

```
SQL> SELECT MIN(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) FROM SCOTT.EMP;

MIN(DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID))
-----------------------------------------
                                      151
```

Gambar 4.24 *Block* ID *Table* Emp

Dari informasi pada gambar 4.24 di dapat informasi salah satu *block id* pada *table* EMP adalah 151, selanjutnya lakukan pengecekan lokasi *datafile* tempat *table* EMP tersimpan seperti berikut:

```
SQL> SELECT OWNER, TABLE_NAME, FILE_NAME
  2  FROM DBA_TABLES TAB, DBA_DATA_FILES DBF
  3  WHERE TABLE_NAME = 'EMP'
  4  AND TAB.TABLESPACE_NAME = DBF.TABLESPACE_NAME;

OWNER      TABLE_NAME FILE_NAME
---------- ---------- -----------------------------------
SCOTT      EMP        /oracle/oradata/prod/users01.dbf
```

Gambar 4.25 *Datafile Table* EMP

Berdasarkan gambar 4.25, lokasi *table* EMP berada pada *datafile* users yang terletak pada /oracle/oradata/prod/users01.dbf. Selanjutnya akan disimulasikan agar *datafile* dengan *block id* 151 tersebut mengalami kerusakan (*corrupt*) dengan menggunakan *command* "dd" yang terdapat pada sistem operasi unix ataupun linux seperti pada gambar 4.26.

```
oracle@primary:~$ dd if=/dev/zero
of=/oracle/oradata/prod/users01.dbf bs=8192 conv=notrunc
count=2 seek=151
```

```
oracle@primary:~$ dd if=/dev/zero of=/oracle/oradata/prod/users01.dbf bs=8192 conv=notrunc count=2 seek=151
2+0 records in
2+0 records out
```

Gambar 4.26 dd *Command*

Selanjutnya gunakan *tool* dbv dari Oracle yang berfungsi untuk memverifikasi ataupun pengecekan struktur *physical database* file dari Oracle.

```
oracle@primary:~$ dbv file=/oracle/oradata/prod/users01.dbf

DBVERIFY: Release 11.2.0.1.0 - Production on Wed Sep 11 14:07:22 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

DBVERIFY - Verification starting : FILE = /oracle/oradata/prod/users01.dbf
Page 151 is marked corrupt
Corrupt block relative dba: 0x01000097 (file 4, block 151)
Completely zero block found during dbv:


DBVERIFY - Verification complete

Total Pages Examined         : 640
Total Pages Processed (Data) : 14
Total Pages Failing   (Data) : 0
Total Pages Processed (Index): 2
Total Pages Failing   (Index): 0
Total Pages Processed (Other): 589
Total Pages Processed (Seg)  : 0
Total Pages Failing   (Seg)  : 0
Total Pages Empty            : 34
Total Pages Marked Corrupt   : 1
Total Pages Influx           : 0
Total Pages Encrypted        : 0
Highest block SCN            : 884077 (0.884077)
```

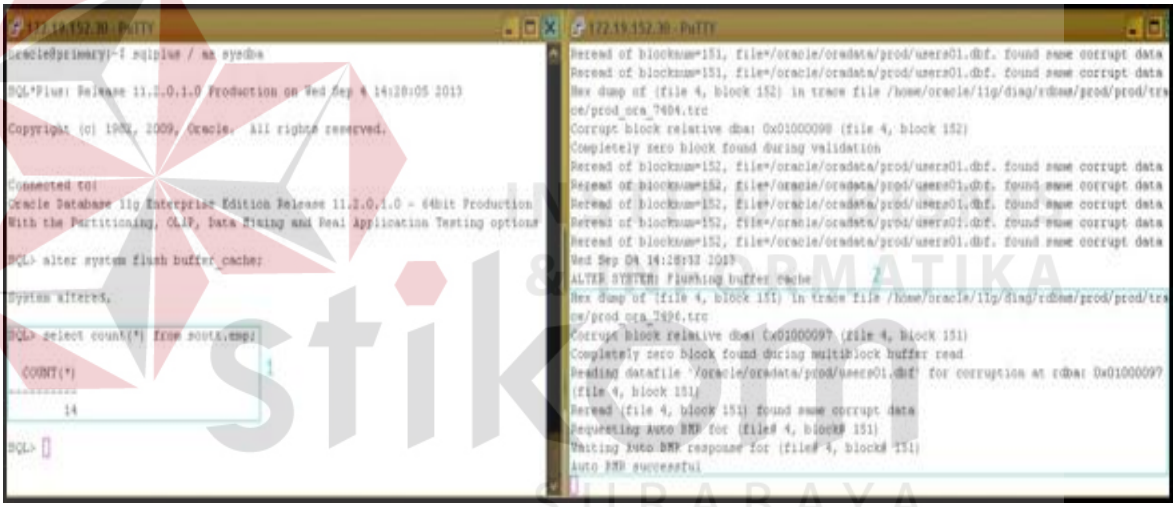
Gambar 4.27 Verifikasi *Database File*

Berdasarkan gambar 4.27 di dapat hasil informasi dari verifikasi terhadap *datafile* /oracle/oradata/prod/users01.dbf dimana terjadi kerusakan data (*corrupt*) pada *block* 151 yang disebabkan dari hasil simulasi menggunakan *command* "dd" sesuai dengan gambar 4.26. Selain itu pengecekan *block corrupt* dari Oracle juga dapat dilihat dari *data dictionary* v$database_block_corruption seperti gambar 4.28.

```
SQL> select * from v$database_block_corruption;

     FILE#     BLOCK#     BLOCKS CORRUPTION_CHANGE# CORRUPTIO
---------- ---------- ---------- ------------------ ---------
         4        151          2                  0 ALL ZERO
```

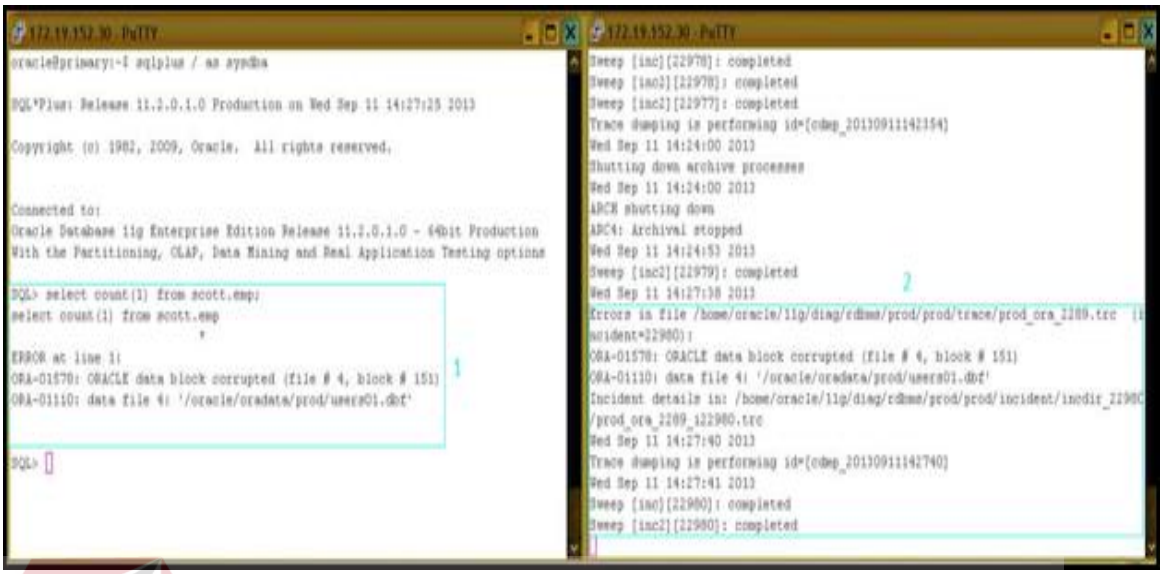
Gambar 4.28 *List Database Block Corrupt*

Setelah *block* 151 yang terdapat pada *table* EMP mengalami kerusakan, selanjutnya dilakukan pengaksesan terhadap *table* EMP secara keseluruhan yang diharapkan terjadi pembacaan terhadap data yang terdapat pada *block* 151. Berdasarkan gambar 4.29 ketika terjadi pengaksesan pada *table* EMP, Oracle mendeteksi adanya kerusakan data (*corrupt*) pada *file* 4 *block* 151 dan *Auto Block Media Recovery* (Auto BMR) akan otomatis melakukan *recovery* pada *block* yang mengalami kerusakan tersebut dengan menyalin *block* yang terdapat pada *standby database*.



Gambar 4.29 *Auto Block Media Recovery*

Jika tidak terdapat active dataguard, ketika dilakukan pembacaan pada *table* EMP dimana terdapat *block corrupt,* akan menghasilkan *output* seperti gambar 4.30



Gambar 4.30 Akses Data *Block Corrupt*

3. Flashback

Teknologi *Flashback Database* mengembalikan keadaan *database* dalam kondisi waktu tertentu *(undo),* dimana kondisi yang dapat dilakukan oleh *flashback* dapat berupa mengembalikan data yang hilang ataupun mengembalikan keadaan *database* ke waktu tertentu (*Timestamp*) dimana jangka waktu yang dapat di *handle* oleh *flashback* bergantung dari parameter *undo_retention* yang digunakan serta besarnya kapasitas *flash_recovery_area.*

a. *Flashback Drop*

Flashback dapat digunakan untuk mengembalikan *table* yang terhapus baik itu yang disengaja ataupun tidak, penghapusan *table* yang tidak disengaja biasanya disebabkan oleh *user* yang memiliki akses terhadap *schema* ataupun dikarenakan *design* dari aplikasi dimana untuk *login* ke dalam aplikasi membutuhkan akses terhadap *schema* tersebut.

Untuk uji coba mengembalikan *table* yang terhapus dengan *flashback*, pada awalnya buat suatu *table* yang akan digunakan untuk pengujian.

```
SQL> create table scott.table_demo as select * from
all_objects;
```

```
22:12:48 SQL> CREATE TABLE SCOTT.TABLE_DEMO AS SELECT * FROM ALL_OBJECTS;

Table created.
```

Gambar 4.31 *Create Table* Demo

Isi data dari TABLE_DEMO merupakan hasil salinan dari *view* ALL_OBJECTS yang berisi 71346 data seperti yang ditunjukkan pada gambar 4.32

```
SQL> select count(1) from scott.table_demo;
```

```
22:13:08 SQL> SELECT COUNT(1) FROM SCOTT.TABLE_DEMO;

  COUNT(1)
----------
     71346
```

Gambar 4.32 Pengecekan Data *Table* Demo

Lakukan simulasi dengan menghapus Table_Demo seperti pada gambar 4.33

```
SQL> drop table scott.table_demo;
```

```
22:13:24 SQL> DROP TABLE SCOTT.TABLE_DEMO;

Table dropped.
```

Gambar 4.33 *Drop Table* Demo

Lakukan pengaksesan terhadap Table_Demo yang telah terhapus pada proses sebelumnya yang ditunjukkan pada gambar 4.34.

```
22:13:33 SQL> SELECT COUNT(1) FROM SCOTT.TABLE_DEMO;
SELECT COUNT(1) FROM SCOTT.TABLE_DEMO
                        *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Gambar 4.34 Pengecekan Data *Table* Demo Setelah *Drop*

Berdasarkan gambar 4.34, ketika dilakukan akses terhadap Table_Demo Oracle akan mengeluarkan error *ORA-00942 : table or view does not exist* dimana *table* tersebut tidak ada dalam *database* dikarenakan telah dilakukan penghapusan.

```
22:13:38 SQL> FLASHBACK TABLE SCOTT.TABLE_DEMO TO BEFORE DROP;

Flashback complete.

22:14:06 SQL> SELECT COUNT(1) FROM SCOTT.TABLE_DEMO;

  COUNT(1)
----------
     71346
```

Gambar 4.35 Operasi *Flashback Table* Demo

Untuk mengembalikan Table_Demo yang telah terhapus sebelumnya, lakukan perintah *flashback* seperti pada gambar 4.35 untuk kemudian lakukan akses kembali pada *table* tersebut. Setelah operasi *flashback*, Table_Demo memiliki keadaan data yang sama seperti sebelum dihapus.

b. *Flashback* Transaksi

Flashback transaksi berfungsi untuk mengembalikan data yang telah dimodifikasi oleh *statement Data Manipulation Language* (DML). Secara umum banyak digunakan untuk mengembalikan keadaan data yang telah dilakukan DML *Update* dan *Delete.*

Untuk uji coba mengembalikan data pada *table* yang dimodifikasi dengan *flashback*, pada awalnya buat suatu *table* yang akan digunakan untuk pengujian.

```
SQL> create table scott.emp_demo as select * from
scott.emp;
```

```
22:34:21 SQL> CREATE TABLE SCOTT.EMP_DEMO AS SELECT * FROM SCOTT.EMP;

Table created.
```

Gambar 4.36 *Create Table* emp_demo

Isi data dari emp_demo merupakan hasil salinan dari *table* emp milik *schema* scott. Kemudian lakukan pengecekan data dengan kondisi seperti pada gambar 4.37.

```
22:44:29 SQL> SELECT ENAME,SAL FROM SCOTT.EMP_DEMO WHERE SAL < 1500;

ENAME             SAL
---------- ----------
SMITH             800
WARD             1250
MARTIN           1250
ADAMS            1100
JAMES             950
MILLER           1300

6 rows selected.
```

Gambar 4.37 Cek Data *table* emp_demo

Lakukan perintah DML *update* untuk modifikasi data pada *table* emp_demo dilanjutkan dengan perintah *commit* seperti gambar 4.38.

```
22:44:39 SQL> UPDATE SCOTT.EMP_DEMO SET SAL = SAL * 1.5 WHERE SAL < 1500;

6 rows updated.

22:45:17 SQL> COMMIT;

Commit complete.
```

Gambar 4.38 *Update* Data *table* emp_demo

Periksa kondisi data setelah dilakukan modifikasi sesuai dengan proses *query* sebelumnya dengan kondisi SAL < 1500.

```
22:45:21 SQL> SELECT ENAME,SAL FROM SCOTT.EMP_DEMO WHERE SAL < 1500;

ENAME             SAL
---------- ----------
SMITH             1200
JAMES             1425
```

Gambar 4.39 Cek Data *Table* emp_demo Setelah *Update*

Dari hasil query pada gambar 4.39 dapat dilihat bahwa sebelum proses DML *update* menghasilkan 6 baris data dan setelahnya menghasilkan 2 data.

```
22:45:37 SQL> FLASHBACK TABLE SCOTT.EMP_DEMO TO TIMESTAMP (SYSTIMESTAMP - INTERVAL '3' minute);

Flashback complete.

22:47:33 SQL> SELECT ENAME,SAL FROM SCOTT.EMP_DEMO WHERE SAL < 1500;

ENAME             SAL
---------- ----------
SMITH              800
WARD              1250
MARTIN            1250
ADAMS             1100
JAMES              950
MILLER            1300

6 rows selected.
```

Gambar 4.40 Operasi *Flashback* Transaksi

Untuk mengembalikan data pada *table* emp_demo yang telah dimodifikasi sebelumnya, lakukan perintah *flashback* seperti pada gambar 4.40 untuk kemudian lakukan akses kembali pada *table* tersebut. Setelah operasi *flashback*, *table* emp_demo memiliki keadaan data yang sama seperti 3 menit sebelumya.

4.2.2.3 Timely Error Detection

Setiap terjadi permasalahan ataupun peringatan dari sisi *database*, Oracle akan selalu mencatat *event* tersebut pada direktori *diagnostic* dalam yang berupa adump, bdump, dan cdump yang tercatat pada file alert.log. File alert.log merupakan acuan pertama seorang DBA ketika dideteksi adanya permasalahan dari sisi *database*, seperti pada gambar 4.29 dan 4.30 yang membahas *block corrupt* dimana alert.log menginformasikan adanya suatu *event* yang terjadi dan dari tiap *error* ataupun peringatan yang terjadi Oracle akan men-*generate* suatu file berektensi .trc yang menjabarkan lebih detail tentang suatu *event* tersebut seperti pada lampiran 2.

4.2.2.4 Continous Operation

Switchover dilakukan ketika adanya kebutuhan untuk *maintenance* ataupun *upgrade* pada *primary database* dengan merubah peran *standby database* menjadi *primary* sehingga ketersediaan dan keberlanjutan transaksi dapat tetap terjaga. Untuk melakukan *switchover*, lakukan *query* untuk memastikan bahwa *primary database* dapat melakukan *switchover* seperti gambar 4.41.

```
SQL> select name, database_role, switchover_status from v$database;
```

```
SQL> SELECT NAME, DATABASE_ROLE, SWITCHOVER_STATUS FROM V$DATABASE;

NAME      DATABASE_ROLE    SWITCHOVER_STATUS
--------- ---------------- --------------------
PROD      PRIMARY          TO STANDBY
```

Gambar 4.41 Status *Switchover*

Status TO_STANDBY berarti *primary database* dapat melakukan pergantian *role* menjadi *standby database.* Jika hasilnya SESSION_ACTIVE berarti masih terdapat *session* yang sedang mengakses *database* pada saat itu, terdapat beberapa hasil *output* dari status *switchover* antara lain :

- *Not Allowed* : Hal ini menandakan baik *standby* maupun *primary database* belum dilakukan *switchover* atau juga dapat menunjukkan tidak adanya *standby database* yang aktif.
- *Switchover Pending* : Terdapat *request* untuk melakukan *switchover* tetapi belum dilakukan, biasanya hal ini terjadi saat kedua *database* sedang melakukan sinkronisasi data sebelum dilakukan switchover.

- *Switchover Latent* : Proses *switchover* berada dalam kondisi *pending*, tetapi tidak dapat terselesaikan dan akhirnya statusnya kembali ke bentuk semula, biasanya hal ini terjadi ketika *primary database* tidak dapat melakukan kontak dengan *standby database*.
- *To Primary* : Menandakan bahwa *database* ini adalah *standby database* yang dapat melakukan *switchover* ke *primary database*.
- *Recovery Needed* : Menandakan bahwa *database* ini adalah *standby database* dan belum menerima *request* untuk dilakukan *switchover*.
- *Preparing Switchover* : Terdapat 2 kondisi yang bisa terjadi. Kondisi pertama, *primary database* sedang menerima *redo* data dari sebuah *logical standby database* dalam persiapannya untuk melakukan *switchover* ke *logical standby database role*. Kondisi kedua, *database* ini adalah *logical standby database role* yang sedang mengirimkan *redo* data ke sebuah *primary database* dan ada *standby database* lain yang sedang bersiap untuk melakukan *switchover* ke *primary database role*.
- *Preparing Dictionary* : Menunjukkan bahwa ini adalah sebuah *logical standby database* yang sedang mengirimkan *redo* data ke sebuah *primary database* dan terdapat *standby database* lain yang sedang bersiap melakukan *switchover* ke *primary database role*.
- *To Logical Standby* : Menandakan bahwa *database* ini adalah *primary database* yang dapat melakukan *switchover* ke *logical standby database*.

Proses *switchover* dapat dilakukan secara manual melalui perintah sql ataupun terpusat melalui *dataguard broker*, gambar 4.42 menunjukkan proses *switchover* yang dilakukan melalui *dataguard broker.*

```
DGMGRL> connect sys/sevima123@prod
Connected.
DGMGRL> switchover to 'prodb';
Performing switchover NOW, please wait...
New primary database "prodb" is opening...
Operation requires shutdown of instance "prod" on database "prod"
Shutting down instance "prod"...
ORA-01109: database not open

Database dismounted.
ORACLE instance shut down.
Operation requires startup of instance "prod" on database "prod"
Starting instance "prod"...
ORA-32004: obsolete or deprecated parameter(s) specified for RDBMS instance
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "prodb"
```

Gambar 4.42 Proses *Switchover Broker*

Dari hasil proses *switchover* pada gambar 4.42 dapat dilihat perubahan *role*, dimana *database* prodb yang sebelumnya berperan sebagai *standby database* menjadi *primary database.* Dari hasil catatan *log* ketika terjadi *switchover* yang terdapat pada lampiran 1 dapat terlihat bahwa proses perubahan *role* ini hanya menghabiskan waktu kurang lebih 2 menit. Lamanya proses *switchover* ini bervariasi bergantung dari banyaknya transaksi, *user* ataupun *session* yang mengakses *database* serta juga sedikit dipengaruhi oleh spesifikasi *resource hardware* dari *database.*

4.3 Analisa Penerapan Oracle *Dataguard*

4.3.1 Perbandingan Sistem

Perbandingan antara sistem *availability* PT. TPS yang sedang berjalan dengan sistem yang diimplementasikan berdasarkan penanganan dan waktu yang dibutuhkan untuk *recovery* adalah sebagai berikut :

Tabel 4.2 Perbandingan Sistem

No	Tipe Gangguan	Penanganan & Waktu Recovery	
		Sistem Sebelumnya	**Sistem yang diterapkan**
1.	*Database Server Down*	DBA harus *onsite* ke lokasi dan melakukan manual *failover*. **Waktu Recovery : 1 – 2 Jam**	*Fast-Start Failover* **Waktu Recovery : 10 – 15 Menit**
2.	*Block Corrupt*	Recovery dari backup / Konfigurasi ulang dataguard **Waktu Recovery : 1 jam**	*Automatic Block Corrupt* **Waktu Recovery : Instant**
3.	*Human Error*	Manual *Recovery* oleh DBA **Waktu Recovery : Terkondisi berdasarkan besar data dan jenis backup yang digunakan, dengan estimasi 10 – 30 Menit.**	Teknologi *Flashback* **Waktu *Recovery* : Instant**

4.	*System Upgrade / Maintenance*	Manual *Switchover* dengan melakukan *remote* ke tiap *database.* **Lama Proses : 3 – 15 Menit**	*Switchover* tersentral dengan *broker,* tanpa *remote* ke tiap *database.* **Lama Proses : 3 – 15 Menit**

Tabel 4.2 menunjukkan perbandingan antara sistem *high availability* yang sebelumnya dengan sistem *high availability* yang diterapkan. Dari tabel tersebut dapat disimpulkan bahwa sistem *high availability* yang diterapkan lebih baik dalam menjaga kelangsungan proses bisnis serta keutuhan dan keamanan data perusahaan ketika terjadi gangguan sistem *database* baik terencana maupun tidak terencana dengan lama waktu *recovery* yang lebih singkat dari sistem sebelumnya.

4.3.2 Analisis Benefit

Analisis *benefit* dilakukan dengan menghitung kemungkinan kerugian yang dialami perusahaan jika terjadi *system down* pada *database* berdasarkan standard produktivitas perusahaan. Standard target produktivitas kecepatan bongkar muat PT.TPS adalah 25 box perjam untuk crane dan 45 box perjam untuk kapal dalam satu group kerja. Salah satu komponen biaya jasa di TPS untuk *standard / dry cargo container* dengan ukuran box 40ft sebesar Rp.805.000/box, jika terjadi ketidaktersediaan transaksi maka estimasi kerugian dengan standard produktivitas 45 box/jam adalah Rp 805.000 x 45 = Rp 36.225.000 per jam untuk satu group. Pada PT. TPS terdapat 4 group kerja yaitu :

1. A Force,
2. Brave Corps,
3. Cigma Stars
4. Delta Force

Kemungkinan terburuknya adalah jika terjadi *system down* pada *database* saat *peak transaction* dimana minimal terdapat 3 group yang beroperasi, maka estimasi kerugian per jamnya adalah :

Rp 36.225.000 x 3 = Rp 108.675.000; -per jam.

Berdasarkan tabel 4.2, jika terjadi *down* pada *database system* sebelumnya memerlukan estimasi waktu *recovery* 1 sampai 2 jam dengan estimasi kerugian :

Rp 108.675.000 s/d 217.350.000.

Dengan penerapan *system database* yang baru, jika terjadi *down* memerlukan estimasi waktu *recovery* kurang dari 15 menit dengan estimasi kerugian yang dapat di *minimalisasi* menjadi :

Rp. 108.675.000 x 15/60 = Rp. 27.168.750.

Dapat dilihat bahwa dengan adanya *fast-start failover* pada sistem *high availability* yang baru dapat meminimalkan kerugian yang dialami perusahaan dibandingkan dengan sistem *high availability* sebelumnya jika terjadi *failure* pada *database* utama.

4.4 Evaluasi

Dari hasil ujicoba *high availability* dengan dataguard Oracle 11g, maka sistem yang diterapkan dapat memenuhi kebutuhan perusahaan akan sistem yang dapat mendukung keberlangsungan proses bisnis perusahaan yang melayani transaksi operasional 24 jam setiap harinya serta menjamin tidak adanya kerusakan data pada saat terjadi gangguan yang tidak terduga pada *primary database.*

Dari hasil analisis implementasi ini, PT. TPS telah memiliki sistem *availability* yang lebih baik dengan benefit yang didapat dari Oracle Dataguard 11g dibandingkan dengan sistem sebelumnya yang menggunakan Oracle Dataguard 9i.