BAB III

PELAKSANAAN KERJA MAGANG

3.1. Kedudukan dan koordinasi

Dalam proses pelaksanaan kerja magang, penulis bekerja dibawah bimbingan bagian *Workforce Development*. Penulis bekerja sebagai trainee pada bagian WDP dan memiliki beberapa tugas antara lain:

a. Mengikuti pelatihan produk Oracle yang diselenggarakan untuk menambah pengetahuan penulis terkait dengan produk Oracle.

b. Mengikuti dan mengerjakan tugas yang berupa studi kasus analisis.

3.2. Tugas yang dilakukan dalam masa *training*

Selama kegiatan kerja magang sebagian besar adalah mengikuti *training* produk Oracle dan analisis mengenai studi kasus. Berikut ini adalah *gant chart* terkait dengan pelaksanaan kerja magang yang dilakukan oleh penulis selama masa kerja magang.

No. Training	Aktivitas / Waktu Kerja Magang	2013				
		Minggu Ke- 1	Minggu Ke- 2	Minggu Ke- 3	Minggu Ke- 4	Minggu Ke- 5
		29 Juli - 2 Agustus	12 Agustus - 16 Agustus	19 Agustus - 23 Agustus	26 Agustus - 30 Agustus	2 September - 6 September
1	Training Oracle Database 12c: New Features for administrators	1				
2	Studi kasus Develop and Tuning Database CD Store		2			
3	Training Data Integration and ETL with Oracle Warehouse Builder			3		
4	Training Oracle Database 11g: Administration Workshop 1 Release 2				4	
5	Training Oracle Database 11g: Administration Workshop 2 Release 2					5

Gambar 3.1 Pelaksanaan Kerja Magang

Pada minggu pertama, penulis mendapatkan *training* terkait dengan produk Oracle terbaru yaitu *training* Oracle *Database* 12C: *New Features for Administration*. *Training* ini menjelaskan bahwa produk baru oracle ini menggunakan basis *Cloud*.

Pada minggu kedua, penulis mendapatkan sebuah tugas terkait dengan studi kasus *develop* dan *tunning database CD Store*. Penulis hanya diberikan sebuah ERD terkait dengan *CD Store* dan penulis harus bisa menyelesaikan dan memperbaiki ERD yang telah diberikan tersebut. Selanjutnya, membuat dan men*develop database CD Store* tersebut serta melakukan *tunning database*.

Pada minggu ketiga, penulis kembali mendapatkan *training* terkait dengan *Data Integration and ETL Oracle Warehouse Builder*.

Pada minggu keempat, penulis mendapatkan *training Oracle Database 11g: Administration Workshop 1 Release 2*. *Training* ini terkait dengan bagaimana mengenai fitur inti dan bagian dalam dari sebuah *database*.

Pada minggu kelima, penulis mendapatkan training Oracle *Data*base 11g: Administration Workshop 2 Release 2. *Training* ini terkait dengan melakukan *backup and recovery database* dengan menggunakan berbagai metode.

3.3. Uraian Pelaksanaan Kerja Magang

3.3.1. *Training* Oracle *Database* 12c: New Features for Administration

Pada minggu pertama, penulis mengikuti *training* terkait dengan produk Oracle terbaru yaitu *Oracle Database 12C: New Features for Administration*. Selama mengikuti masa *training* ini, penulis harus mengikuti peraturan *training* yaitu mengikuti *training* secara tenang dan tidak mengganggu aktivitas mengajar selama masa *training*. *Training* ini dipandu oleh Bapak Khulung Bursa dan *training* ini dilakukan selama 1 minggu.

Oracle Database 12c adalah produk oracle *database* terbaru yang dirancang khusus untuk memenuhi keinginan perusahaan yang mencari teknologi pendukung yang mampu meningkatkan kinerja, kecepatan dan efektivitas operasional perusahaan. Fitur utama dari Oracle 12c adalah fiturnya yang memiliki arsitektur *multitenant* dan diletakan di *cloud*. Fitur ini mampu menyederhanakan proses konsolidasi *database* ke dalam *server cloud* dan memungkinkan *user* mengelola banyak *database* tanpa harus mengganti aplikasi mereka. Fitur-fitur lainnya yaitu:

1. **Mengotomatisasi dan Optimalkan Siklus In*form*asi**

   Pada pengelolaan siklus hidup in*form*asi, sebuah *data* akan dikompres sesuai dengan umur *data* tersebut. Misalnya *data* sudah tersimpan 5 tahun lebih maka akan dilakukan

kompresi terhadap *data* tersebut sehingga *space* akan tersedia untuk *data* yang muda atau baru saja masuk. Pada Oracle 12c mempunyai fitur otomatisasi *data* secara otomatis, oracle akan me*monitor* usia sebuah *data* dan temperature sebuah *data*. Oracle akan secara otomatis mengkompres *data* dan memindahkan *data* tersebut antara *storage* dengan tingkat kompresi yang sesuai/

2. ***Plug* atau masuk kedalam *cloud* dengan Oracle *Multitenant***

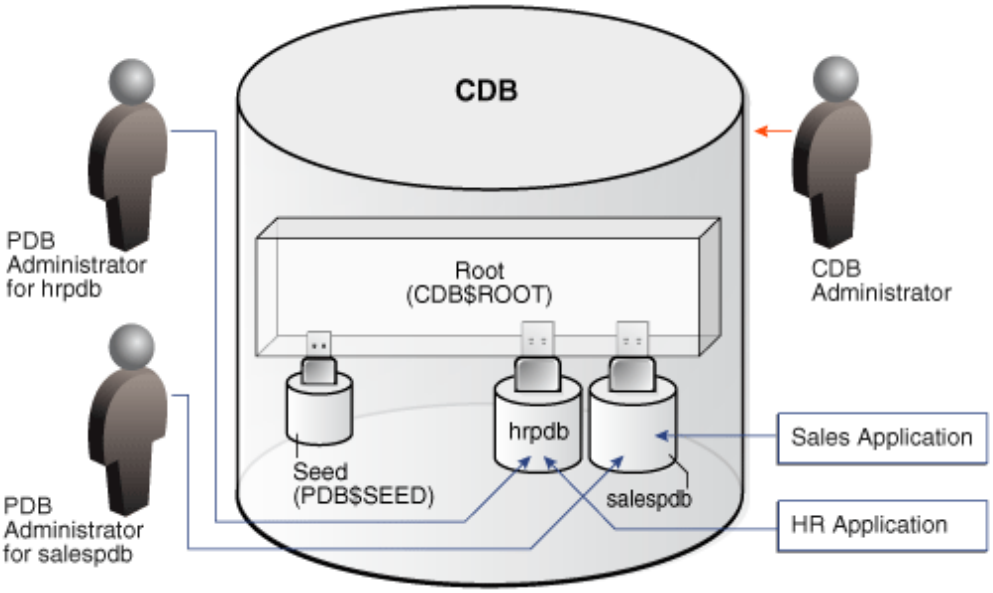
   Oracle 12c mempunyai sebuah fitur yang disebut *database pluggable,* hal ini membuat sebuah *instance* dapat menghandle *database* dengan jumlah yang banyak karena terdapat fitur *container* dan *pluggable database* di dalam oracle 12c. Pada single *instance*, *container database multitenant* dapat berisi banyak *pluggable database*. Setiap *database* akan terhubung dengan *container multitenant*, hal ini akan mempercepat perjalanan ke *cloud* dan konsolidasi juga menyederhanakan manajemen yang merupakan keuntungan yang besar untuk organisasi dengan mengelola banyak *database* sekaligus.

.

3. Ketersediaan Maksimum dan Keamanan *Database*

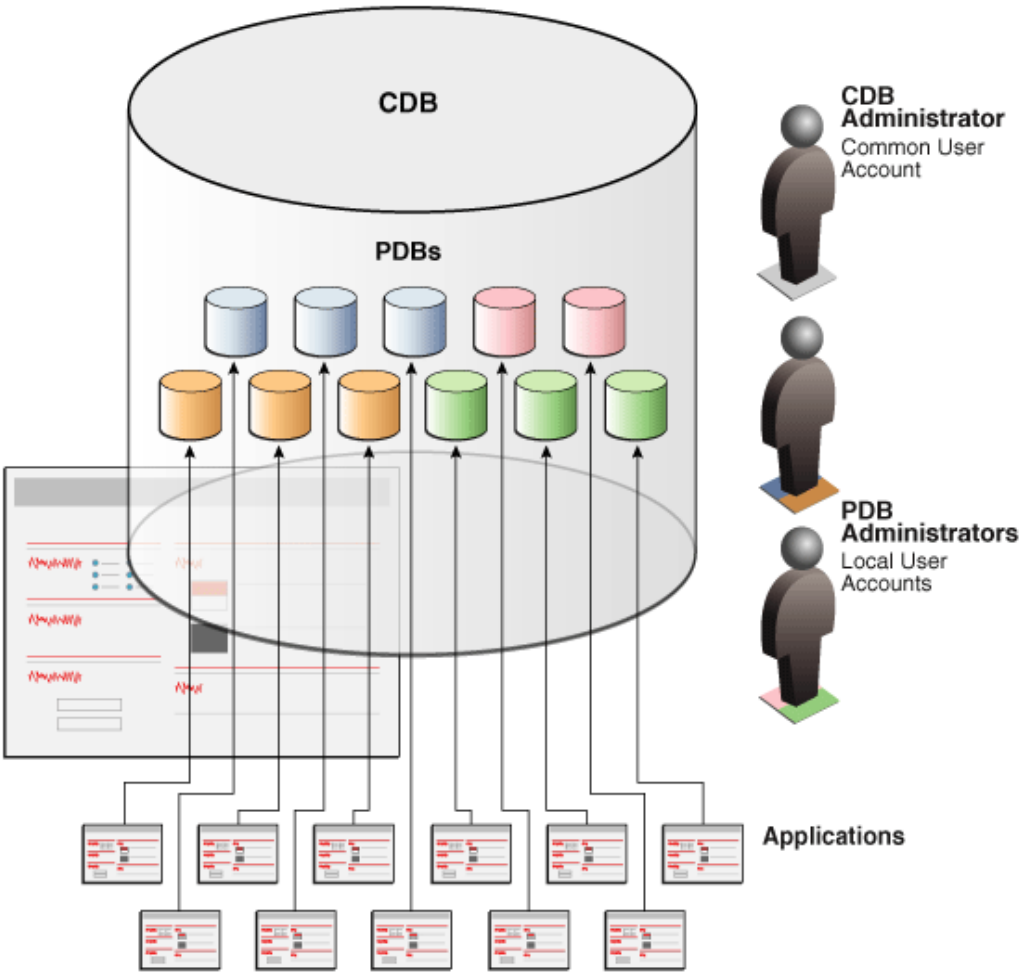
500 fitur baru lainnya membuat oracle *database* 12c memiliki keamanan maksimal dan memiliki ketersediaan yang maksimum.

Container atau CDB merupakan sebuah root yang didalamnnya terdapat PDB atau *pluggable database*, setiap CDB dapat memiliki 1 atau banyak PDB didalamnnya, hal ini memungkinkan *user* bisa menganti *database* mereka tanpa harus menganti aplikasi mereka



Gambar 3.2 CBD *and* PDB

(sumber: Oracle Docs [9])



Gambar 3.3 CDB *and* PDB *architecture*

(sumber: Oracle Docs [9])

Hal ini bisa kita lihat dari gambar di atas ini. 1 CBD dapat berisikan banyak *database* seperti PDB untuk HRD, PDB untuk sales, dan lain-lainnya. Berikut ini adalah cara untuk membuat CDB baru dan melihat status dari CBD:

```
CREATE DATABASE newcdb
USER SYS IDENTIFIED BY sys_password
USER SYSTEM IDENTIFIED BY system_password
EXTENT MANAGEMENT LOCAL
DEFAULT TABLESPACE users
DEFAULT TEMPORARY TABLESPACE temp
UNDO TABLESPACE undotbs1
ENABLE PLUGGABLE DATABASE
   SEED
   SYSTEM DATAFILES SIZE 125M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
   SYSAUX DATAFILES SIZE 100M;
```

Gambar 3.4 *Create new* CBD

(sumber: Oracle Docs [9])

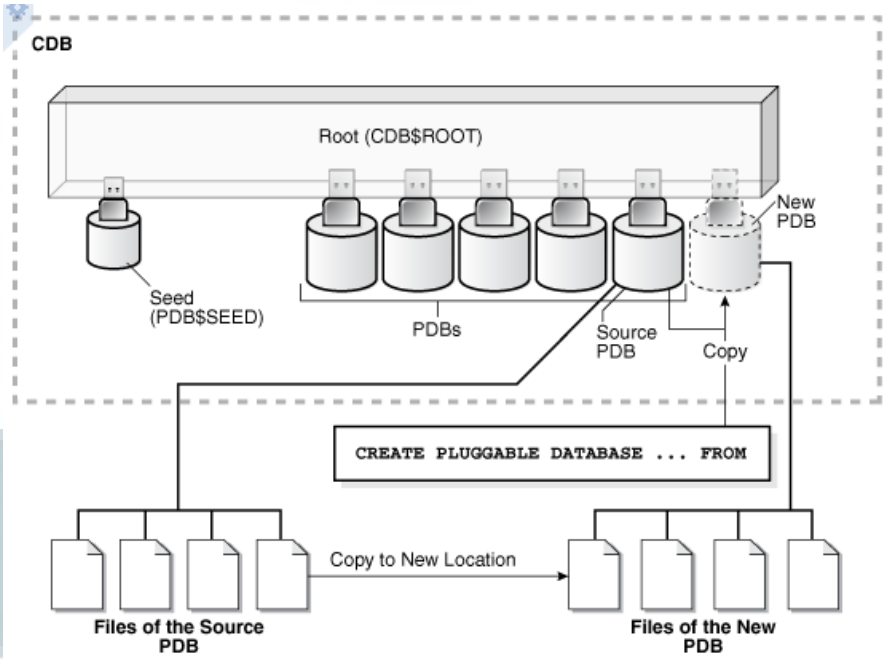
```
SQL> SELECT NAME, CDB, CON_ID FROM V$DATABASE;

NAME      CDB     CON_ID
--------- --- ----------
CDB1      YES          0
```

Gambar 3.5 *Show* CDB *status*

(sumber: Oracle Docs [9])

Ada 3 cara dalam membuat PDB yaitu dengan cara sebagai berikut ini:

- *Creation of a PDB from seed*



Gambar 3.6 *Creation of a PDB from seed*

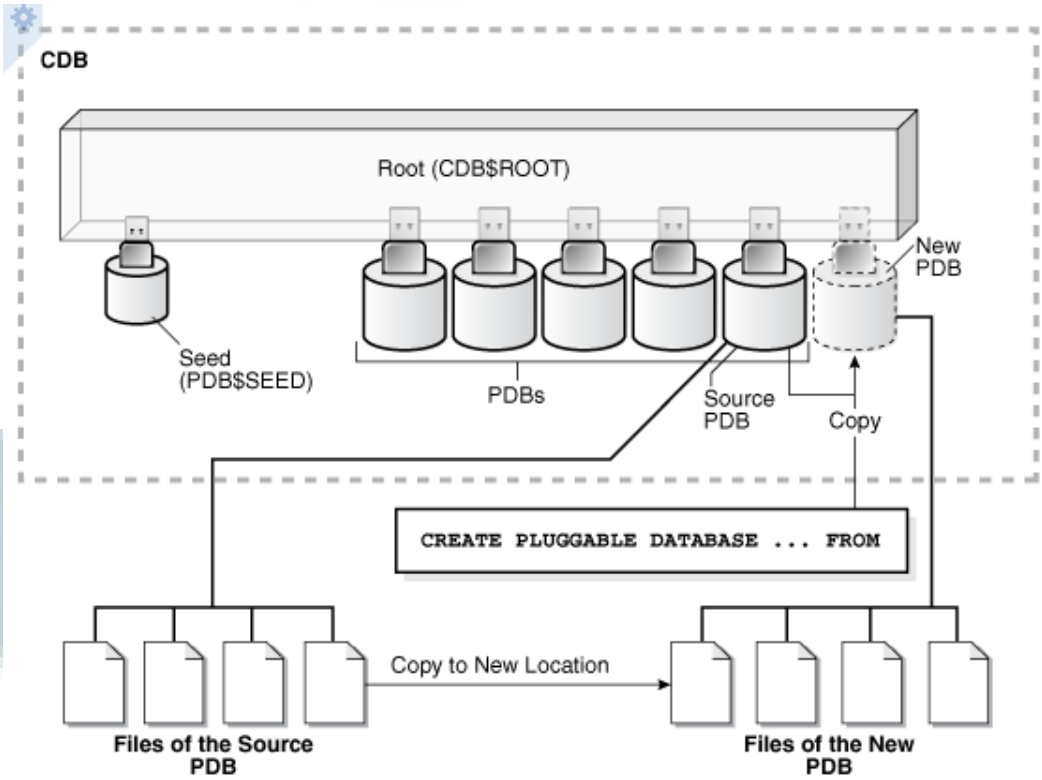
(sumber: Oracle Docs [9])

```
CREATE PLUGGABLE DATABASE hrpdb
 ADMIN USER dba1 IDENTIFIED BY password
```

Gambar 3.7 *Syntax Creation of a PDB from seed*

(sumber: Oracle Docs [9])

- *Creation of a PDB by cloning a PDB*



Gambar 3.8 *Creation by cloning a PDB*

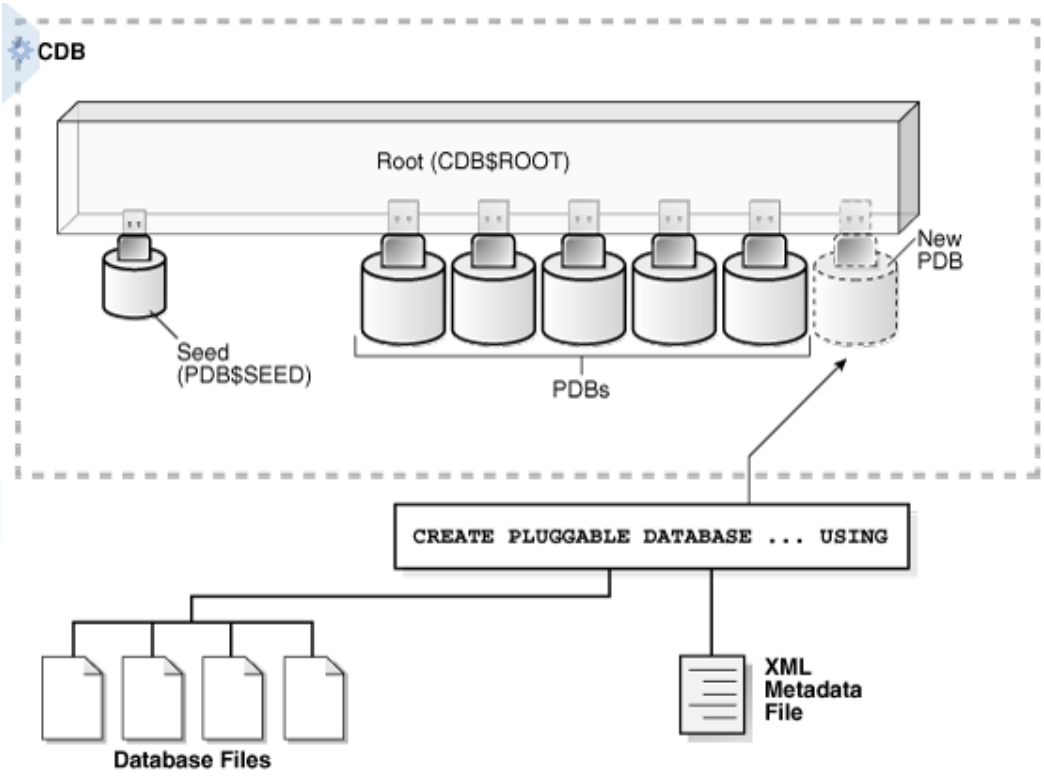
(sumber: Oracle Docs [9])

```
CREATE PLUGGABLE DATABASE salespdb FROM hrpdb
```

Gambar 3.9 *Syntax Creation by cloning a PDB*

(sumber: Oracle Docs [9])

- *Creation of a PDB by plugging in an unplugged PDB*



Gambar 3.*10 Creation of a PDB by plugging in an unplugged PDB*

(sumber: Oracle Docs [9])

```
CREATE PLUGGABLE DATABASE salespdb USING '/disk1/usr/financepdb.xml' NOCOPY
```

Gambar 3.11 *Syntax Creation of a PDB by plugging in an unplugged PDB*

(sumber: Oracle Docs [9])

Hal yang menyebabkan fitur didalam Oracle 12c ini terdapat PDB dan CDB adalah

1. *Database* ini ditaruh pada *server cloud* tentu saja *database* ini digunakan oleh *multitenant* atau *user* yang banyak.
2. Dengan menggunakan PDB dan CDB tidak akan memakan banyak *resource*
3. Tidak perlu adanya *switch user* dan membuat aplikasi harus *switch* aplikasi. Hal ini disebabkan adanya fitur *unplug* dan *plug database*.

3.3.2. Studi kasus *develop* dan *tunning database CD Store*

Pada minggu kedua, penulis mendapatkan sebuah tugas studi kasus terkait dengan men-*develop database* dan melakukan *tunning* terhadap *database* tersebut. Pada tugas ini, penulis mengerjakan bersama 2 orang peserta magang lainnya. Tujuan dari tugas ini adalah untuk melatih analisa dan koordinasi antar anggota dalam menyelesaikan tugas ini.

Hal pertama yang dilakukan oleh penulis yaitu menganalisis ERD yang telah diberikan dan memperbaiki ERD tersebut. Setelah memperbaiki ERD tersebut. Penulis bersama tim mulai men-*develop database* yang diambil dari ERD yang telah dibuat. Berikut ini adalah beberapa *syntax* membuat *database* dan membuat *table* yang dibutuhkan oleh *CD Store*.

```
create database videostore;
```

Gambar 3.12 *Syntax Create New Database*

```
create table video_storebranch(
branchno varchar(15) CONSTRAINT branchno_pk PRIMARY KEY,
br_addr varchar(75),
br_phone varchar(15),
city_code varchar(5),
constraint city_code_fk FOREIGN KEY (city_code) REFERENCES city (city_code)
);

create table employee(
empno varchar(15) CONSTRAINT empno_pk PRIMARY KEY,
empname varchar(50),
empaddr varchar(75),
gender varchar(10),
birthdate date,
branchno varchar(15),
CONSTRAINT gender_1 CHECK (gender IN ('perempuan','lakilaki')),
constraint branchno_fk_1 FOREIGN KEY (branchno) REFERENCES video_storebranch
(branchno)
);
```

Gambar 3.13 *Syntax New Table*

Setelah melakukan pembuatan *database* beserta melakukan penginputan *data* yang diperlukan. Penulis bersama tim mulai melakukan *tunning database*. *Tunning database* ini bertujuan untuk meningkatkan per*form*a dari *database* sehingga *database* dapat bekerja secara optimal dan efisien. Ada beberapa bentuk *tunning* yaitu *tunning hardware* yang berhubungan dengan media penyimpanan, *tunning query* yaitu melakukan pemeriksaan ulang terkait dengan *query* yang diinput apakah sudah efisien atau belum, dan terakhir adalah *tunning database* yang bertujuan untuk mengoptimalkan fungsi dari *database*. *Tunning* yang dilakukan oleh penulis dan tim adalah melakukan partisi *database*. Berikut ini adalah *syntax* partisi yang dilakukan oleh penulis dan tim.

```
create table transaction_header(
transno varchar(15) CONSTRAINT transno_pk PRIMARY KEY,
empno varchar(15),
custno varchar(15),
transactiondate date DEFAULT to_char(sysdate,'dd-month-yyyy'),
late_amt_due int DEFAULT 0,
branchno varchar(15),
v_partition varchar(5) generated always as
        (select a.city_code from video_storebranch a where a.branchno = branchno)
virtual,
constraint branchno_fk_3 FOREIGN KEY (branchno) REFERENCES video_storebranch
(branchno),
constraint empno_fk_1 FOREIGN KEY (empno) REFERENCES employee (empno),
constraint custno_fk_1 FOREIGN KEY (custno) REFERENCES customer (custno)
)
partition by list (v_partition)
subpartition by hash (transactiondate)
subpartition template
(
        subpartition sp_branch1 tablespace ts1,
        subpartition sp_branch2 tablespace ts2,
        subpartition sp_branch3 tablespace ts3,
        subpartition sp_branch4 tablespace ts4,
        subpartition sp_branch5 tablespace ts5
)
(
        partition ts_branch1 values ('JAKARTA'),
        partition ts_branch2 values ('BANDUNG'),
        partition ts_branch3 values ('DEPOK'),
        partition ts_branch4 values ('TANGERANG'),
        partition ts_branch5 values ('BEKASI')
);
```

Gambar 3.14 *Create New Partition by List and Hash*

Tujuan dari melakukan partisi adalah untuk mempercepat pencarian *data*. Oracle memiliki 4 macam partisi, yaitu *range*, *list*, *hash* dan composite. Pada *range partitioning*, *data* dikelompokan berdasarkan oleh *range* yang telah ditentukan. Berikut ini adalah contoh penggunaan *range partitioning*:

```
CREATE TABLE penyewaan
( notransaksi  NUMBER,
tglpeminjaman    DATE NOT NULL)
PARTITION BY RANGE (tgl_jual)
(
PARTITION peminjaman_cd1 VALUES LESS THAN (TO_DATE('01-JUN-2013','DD-MON-YYYY')) TABLESPACE users,
PARTITION peminjaman_cd2 VALUES LESS THAN (TO_DATE('01-JUL-2013','DD-MON-YYYY')) TABLESPACE users,
PARTITION peminjaman_cd3 VALUES LESS THAN (TO_DATE('01-SEP-2013','DD-MON-YYYY')) TABLESPACE users,
PARTITION peminjaman_cd4 VALUES LESS THAN (TO_DATE('01-DES-2013','DD-MON-YYYY')) TABLESPACE users
);
```

Gambar 3.15 *Syntax Create new range partition*

Partitioning yang kedua adalah *list partitioning*. Pada *partitioning* ini *data* dikelompokan berdasarkan nilai yang telah ditentukan sebelumnya. Misalnya adalah sebagai berikut:

```
CREATE TABLE penyewaan
( notransaksi  NUMBER,
tglpeminjaman    DATE NOT NULL,
daerah varchar(15))
PARTITION BY LIST (tgl_jual)
(
PARTITION Barat VALUES ('JAKARTA','MEDAN','BANDUNG') ,
PARTITION Timur VALUES ('SEMARANG','SURABAYA','MAKASAR')
) TABLESPACE users;
```

Gambar 3.16 *Syntax Create List Partition*

Partitioning yang ketiga adalah *hash partitioning*. Metode *partitioning* ini memberikan sepenuhnya *partition* terhadap oracle dan *internal* oracle akan mengatur nilai akan ditaruh di partisi yang mana. Berikut ini adalah penggunaan dari *hash partitioning*:

```
CREATE TABLE penyewaan
( notransaksi  NUMBER,
tglpeminjaman   DATE NOT NULL,
daerah varchar(15))
PARTITION BY HASH (no_invoice)
PARTITIONS 4 tablespace users;
```

Gambar 3.17 *Syntax Create Hash Partition*

Partitioning selanjutnya adalah composite *range-list* yang merupakan gabungan *range partition* dan *list partition*. *Partitioning* lainnya yaitu composite *range-hash partition* yang merupakan gabungan antara *range* dan *hash partition*.

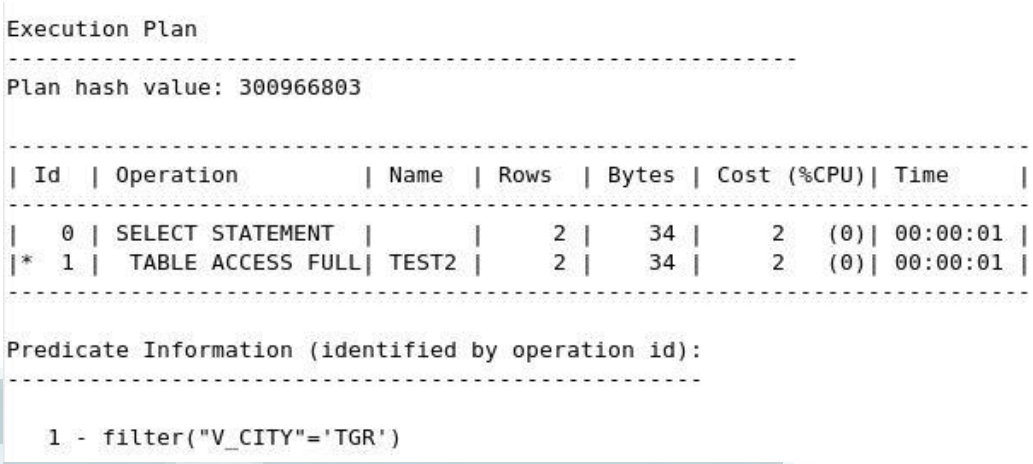
Partitioning bertujuan untuk mempercepat pencarian ketika user melakukan *statement select*. Penulis menggunakan sample data 250 data untuk melihat kecepatan *select* dengan menggunakan partisi dan tidak menggunakan partisi.

```
SQL> select * from test1 where v_city ='TGR';

TRXNO           V_CITY
--------------- ------------
TEST1_20        TGR
TEST2_20        TGR

Elapsed: 00:00:00.03
SQL> select * from test2 where v_city ='TGR';

TRXNO           V_CITY
--------------- ------------
TEST2_20        TGR
TEST2_20        TGR

Elapsed: 00:00:00.08
```

Gambar 3.18 Perbandingan Kecepatan Mencari Data

```
Plan hash value: 2255668434

-------------------------------------------------------------------------------
---------------

| Id  | Operation             | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
Pstart| Pstop |

-------------------------------------------------------------------------------
---------------

|   0 | SELECT STATEMENT      |       |     2 |    34 |     2   (0)| 00:00:01 |
      |       |

|   1 |  PARTITION LIST SINGLE|       |     2 |    34 |     2   (0)| 00:00:01 |
  KEY |   KEY |

|   2 |   TABLE ACCESS FULL   | TEST1 |     2 |    34 |     2   (0)| 00:00:01 |
    4 |     4 |

-------------------------------------------------------------------------------
---------------
```

Gambar 3.19 *Select* Menggunakan *Partition*



Gambar 3.20 *Select* Tidak Menggunakan *Partition*

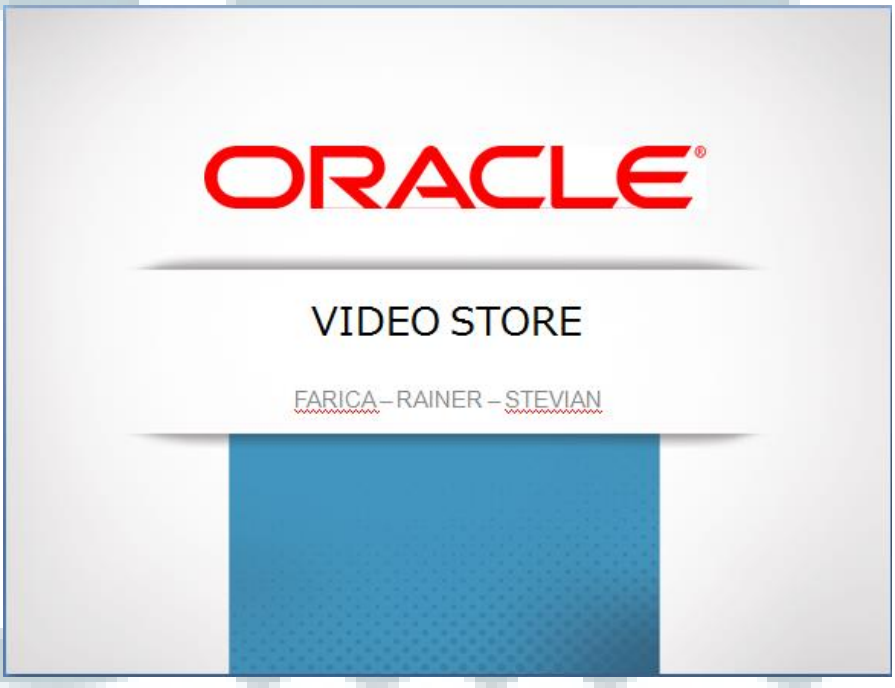
Pada gambar 3.18 dapat dilihat perbandingan antara menggunakan partisi pada bagian select pertama dan non partisi pada select yang kedua. Hal ini dikarenakan ketika melakukan select, database oracle akan mencari partisi yang memiliki data yang user select dan langsung memunculkannya, sedangkan select dengan non partisi oracle akan membaca table secara penuh. Hal ini membuat select dengan non partisi menjadi lebih lambat daripada select dengan menggunakan partisi.

Tunning yang dilakukan oleh penulis yaitu melakukan partisi dengan metode *list* pada *branch* store dan *hash* pada *transaction date*. Hal ini bertujuan untuk membuat partisi setiap *branch CD Store* misalnya *CD Store* yang terletak di wilayah Jakarta, Bandung, Tangerang, Depok, dan Bekasi sehingga ketika *user* ingin mencari *data branch* di wilayah Jakarta akan cepat didapatkan.

Setelah melakukan *tunning database*, penulis dan tim harus mengerjakan beberapa soal terkait dengan *query* untuk memunculkan *data*. Kasus yang diberikan adalah memunculkan *data* terkait dengan judul film sherk seperti dibawah ini:

```
BEGIN
        open cur_movie_search(movie_name);
        DBMS_OUTPUT.PUT_LINE('Branch :');
        DBMS_OUTPUT.PUT_LINE('----------------------------------------');
        loop
                fetch cur_movie_search INTO movies;
                EXIT WHEN cur_movie_search%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('Branch Address : '||movies.Address);
                DBMS_OUTPUT.PUT_LINE('Phone No       : '||movies.Phone);
                DBMS_OUTPUT.PUT_LINE('----------------------------------------');
        end loop;
        close cur_movie_search;
END;
/

exec SEARCH_MOVIE('Shrek');
```

Gambar 3.21 *Syntax Select Data*

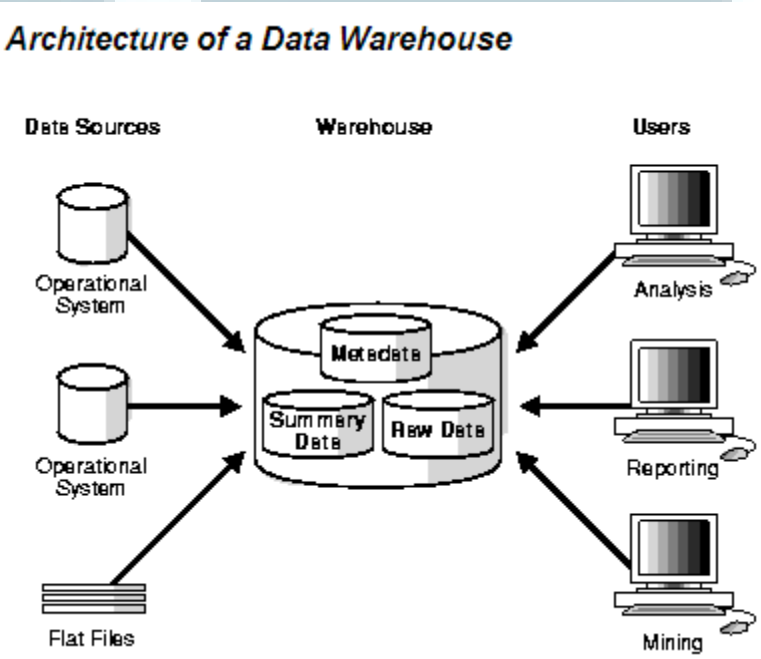


Gambar 3.22 *Slide* Presentasi *Video Store*

3.3.3. *Training Data Integration and ETL Oracle Warehouse Builder*

Pada minggu ketiga, penulis mengikuti *training Data Integration and ETL Oracle Warehouse Builder*. Pada *training* ini penulis diwajibkan mengikuti peraturan yang telah ditentukan sebelumnya. *Training* ini dipimpin oleh Pak Khulung Bursa.

Data Warehouse adalah kumpulan *data* yang dikumpulkan menjadi satu. Hal ini bertujuan untuk analisa *data* yang telah dikumpulkan tersebut dan berguna untuk pengambilan keputusan.

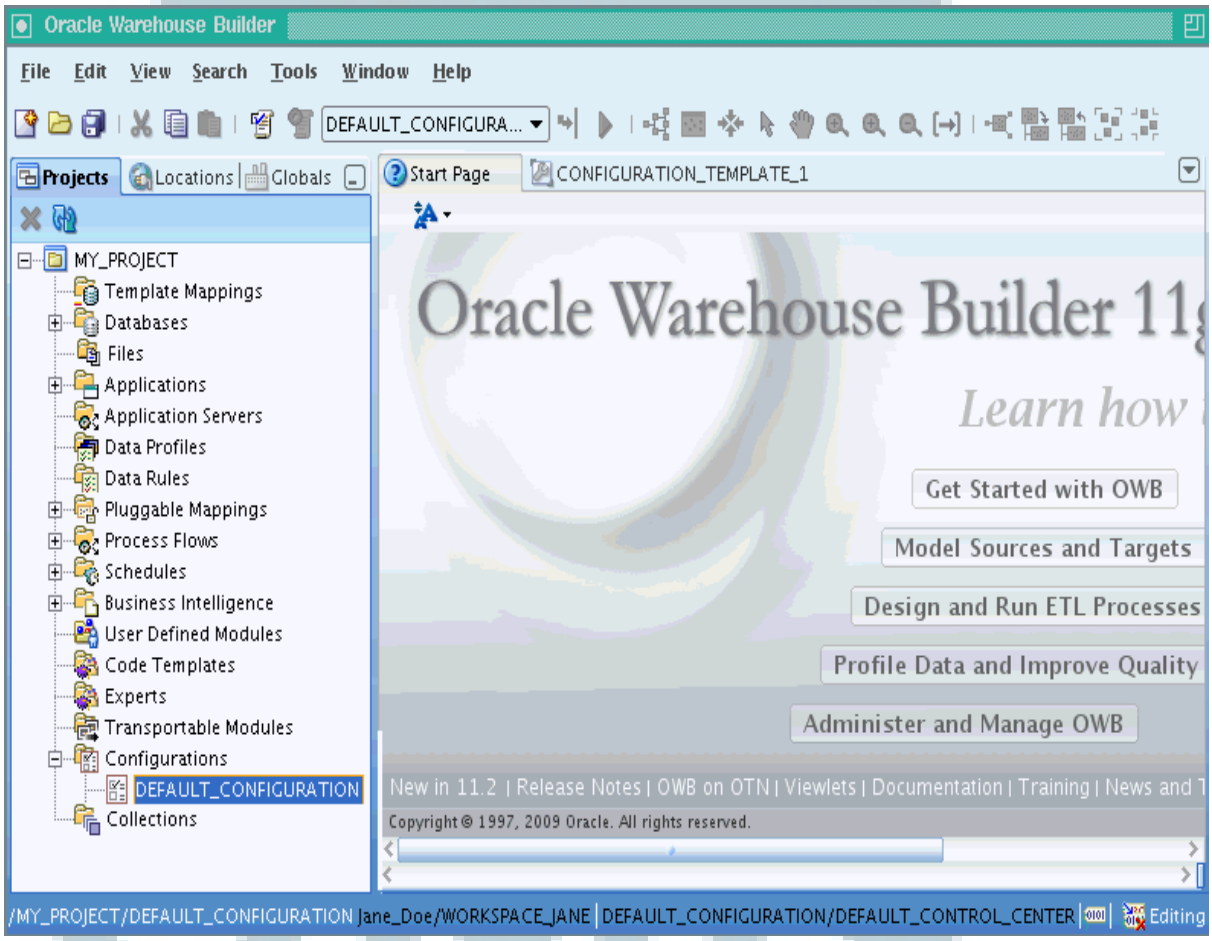


Gambar 3.23 *Architecture of a Data Warehouse*

(sumber: Oracle Docs [12])

Oracle *Warehouse* Builder adalah sebuah tools yang bertujuan untuk mempermudah pekerjaan dalam membuat *data warehouse*. Oracle *Warehouse* Builder juga berfungsi untuk mem-*built*,

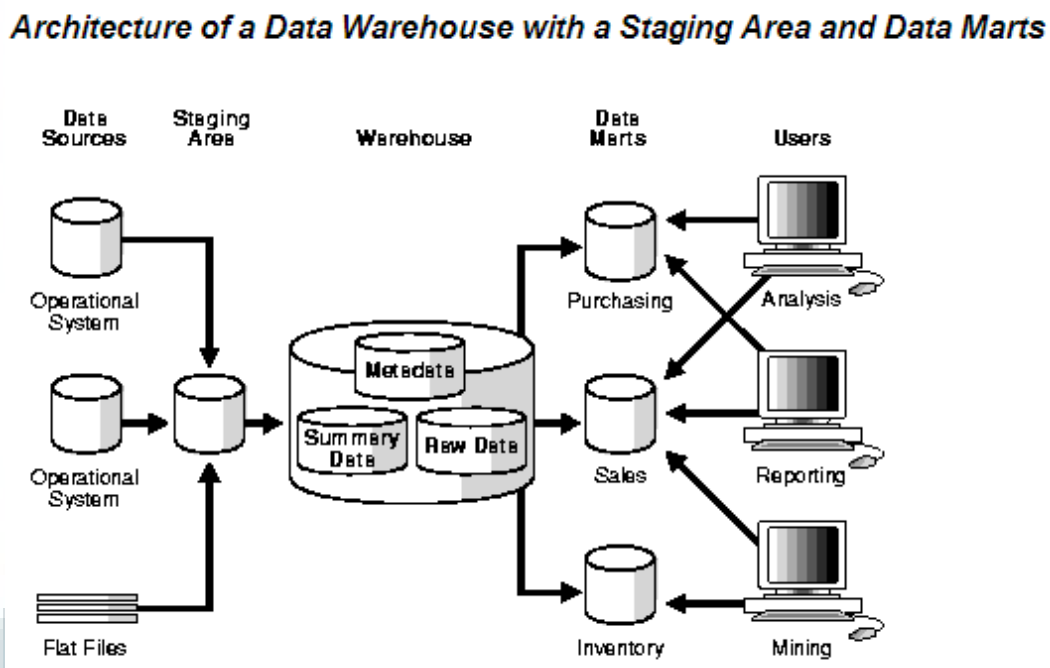
men-*deploy* dan me-*run* ETL atau *Extract*, *Transform*, *Load*. ETL adalah fasilitas yang digunakan untuk membangun *data warehouse*. ETL bekerja dengan beberapa cara, pertama ETL Melakukan ekstrasi *data* dari sumber *internal* dan eksternal. Kedua ETL akan melakukan *transform*asi *data* ke struktur yang dibutuhkan dan melakukan *sorting data*. Ketiga maka *data* tersebut akan di-*loading* dan dimunculkan ke dalam *data warehouse*.



Gambar 3.24 *Oracle Warehouse Builder 11g*

Sebelum menjadi *data* yang bersih maka *data* yang di*transform* akan melewati *staging area*. *Staging area* adalah sebuah tempat bagi *data*

untuk melakukan *cleansing data* dan men*transform data* sebelum menjadi *data warehouse*. Selanjutnya, *data* yang sudah melewati *staging area* akan dikumpulkan menjadi satu di dalam *data warehouse* sebelum menjadi *data marts*. *Data marts* adalah sebuah tempat penyimpanan *data* yang beorientasi pada departemen tertentu saja sedangkan *data warehouse* adalah sebuah tempat penyimpanan yang digunakan oleh *corporate* perusahaan untuk dianalisa.



Gambar 3.25 *Architecture of a Data Warehouse with a Staging Area and Data Marts*

(sumber: Oracle Docs [12])

Data warehouse ini memiliki 2 dimension model yaitu:

1. ROLAP (*Relational* OLAP) adalah OLAP yang menggunakan *relational database* untuk menyimpan *detail* dari sebuah *data*.
2. MOLAP (*Multi dimensional* OLAP) adalah OLAP yang memiliki *storage* tersendiri, yang isinya merupakan *precomputed agregasi data* seperti *sum*, *count*, *min*, dan *max*.

3.3.4. *Training* Oracle *Database* 11g: Administration Workshop 1 & 2 Release 2

Pada minggu keempat dan kelima, penulis mengikuti *training Oracle Database 11g: Administration Workshop 1 & 2 Release 2.* Pada *training* ini penulis diwajibkan mengikuti peraturan yang telah ditentukan sebelumnya. *Training* ini dipimpin oleh Pak Tri.

Training ini dimulai dengan perkenalan *instance* dan *database*. Oracle *database server* memiliki 2 struktur penting yaitu *instance* dan *database*. *Instance* merupakan sebuah program yang akan menjalankan *database* contohnya seperti *Microsoft Word* dan *database* dari *Microsoft word* tersebut berupa *.docx*.

Gambar 3.26 *Instance and Database*

Oracle *instance* terdiri dari beberapa *Memory structure* dan *background* proses. *Memory structure* tersebut berupa *physical files* yang terdiri dari 3 *Memory* mandatory yang wajib ada, apabila *file* atau *Memory* tersebut di*kill* atau di*delete* maka akan membuat *database* menjadi tidak terbaca. *Physical files* lainnya yaitu 2 *Memory optional* yaitu *java pool* and *large pool*. *Background process* pun dibagi menjadi

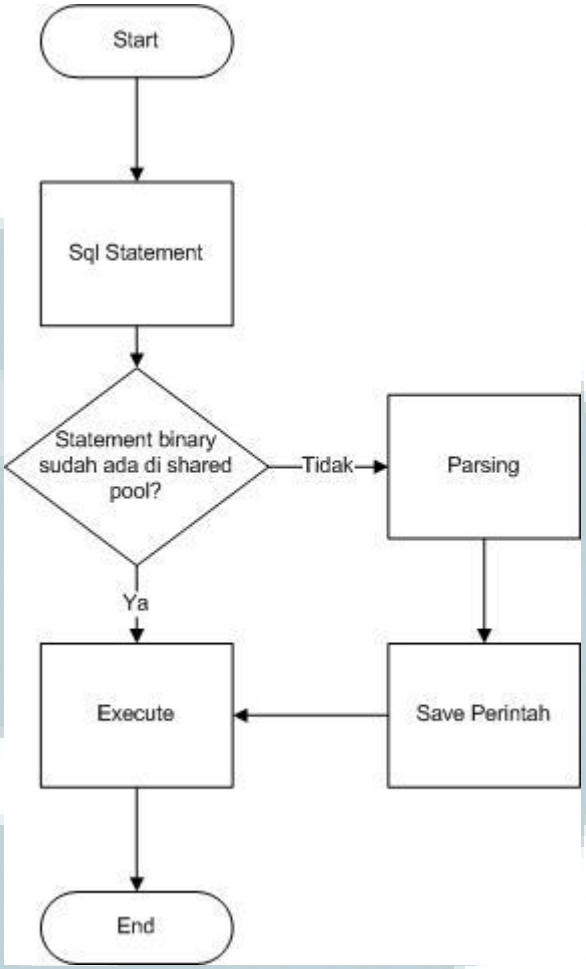
beberapa bagian yaitu DBWR (*Database Writter*), CKPT (*Check Pointer*), LGWR (Log*Writter Process*), SMON (*System Monitor*), PMON (*Process Monitor*), dan RECO (*Recovery*).

<table>
<tr><td rowspan="2">Shared Pool</td><td rowspan="2">DB Buffer Cache</td><td rowspan="2">Redo Log Buffer</td><td>Java pool</td></tr>
<tr><td>Large pool</td></tr>
</table>

Table 3.1 *Memory Mandatory and Physical files*

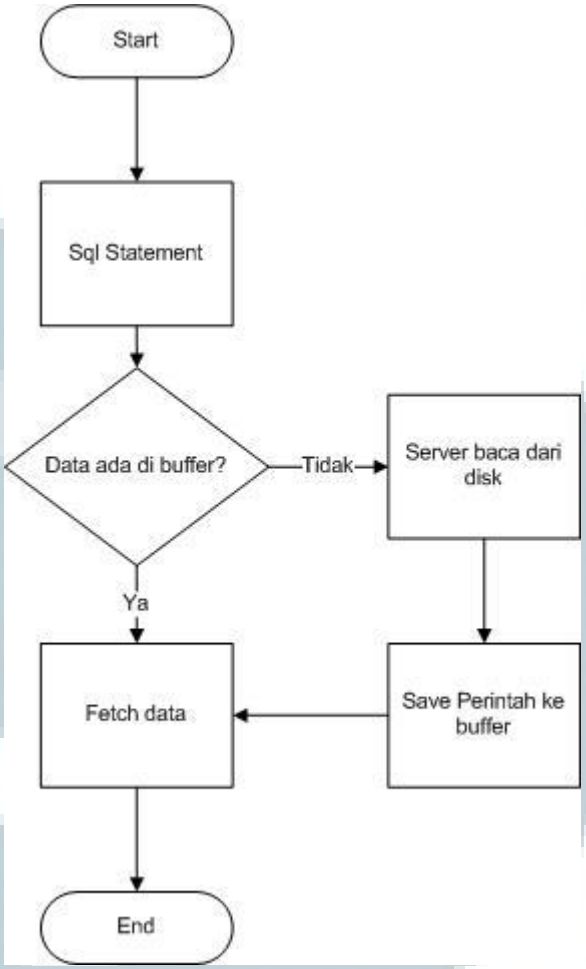
Shared Pool berfungsi untuk menyimpan *sql* versi *binary* yang pernah diakses oleh *user*. Tujuannya untuk *performance tunning* agar oracle tidak sering melakukan *hard parsing*. *Parsing* dibagi menjadi 2 yaitu:

1. *Hard parsing* yaitu meng*convert* dari bahasa manusia menjadi bahasa *binary*.
2. *Soft parsing* yaitu sebuah proses pengambilan *sql* versi *binary* dari *server* proses untuk dieksekusi.



Gambar 3.27 *Flowchart Shared Pool*

DB Buffer cache berfungsi untuk menyimpan *data* yang pernah diakses oleh *user*. Tujuannya untuk *tunning* dalam hal mengurangi frekuensi pembacaan *disk*. Permasalahan yang sering terjadi dalam *DB Buffer cache* adalah sering kali *data* tidak dicache ke DB *Buffer*. Hal ini disebabkan *select* melibatkan *large object* dan sering menggunakan *parallel query* yang digunakan untuk menampilkan *data* yang sangat besar misalnya *select* /*+ *parallel* 16 */.



Gambar 3.28 *Flowchart DB Buffer Cache*

Redo Log Buffer berfungsi untuk menampung sementara *redo entries* sebelum dipindahkan ke dalam *redo log files*. Hal ini bertujuan untuk *tunning* agar penulisan ke*disk* dapat dikurangi. *Redo entries* adalah statement *sql* versi *binary* yang menyebabkan perubahan. *Redo Log Buffer* akan dihapus ketika kondisi dibawah ini dipenuhi.

1. Setiap 3 detik
2. Jika *redo entries* mencapai 1 MB

3. Jika 1/3 *redo log buffer full*
4. Ketika *user* melakukan *commit*.

Pada saat *user* melakukan *commit* maka *background process* akan bekerja secara otomatis. Berikut ini adalah *step* yang dilakukan oleh *background process* ketika *user* melakukan *commit*:

1. CKPT akan menulis SCN ke *System Data File*
2. LGWR akan memindahkan isi *Redo Log Buffer* kedalam *redo log file*
3. Suatu saat DBWR akan memindahkan *data* dari DB *Buffer* kedalam *datafile*.
4. *Checkpoint* akan terjadi ketika DBWR melakukan penulisan kedalam *datafile*.

Control file adalah *file* biner yang diperlukan ketika *database* sedang *startup* dan beroperasi dengan sukses. *Control file* juga berisi in*form*asi tentang konsistensi *database* yang digunakan pada saat melakukan *recovery*. *Control file* akan dibaca pertama kali ketika *instance* dalam posisi *mount*. *Control file* ini akan selalu diupdate ketika ada perubahan dalam hal pemindahan *redo log file*, terjadi penambahan *datafile* dan penghapusan *datafile*. Untuk melihat *control file* diletakkan di *folder* mana dapat menggunakan perintah.

"Select name from V$CONTROLFILE"

Redo log files adalah *file* yang berisi *redo entries* atau berisi statement *sql* versi *binary* yang menyebabkan perubahan. *Redo log files* memiliki beberapa status yaitu:

1. *Current, group redolog files* sedang dipakai
2. *Active, group* yang sekarang digunakan sedang tidak dipakai dan isinya belum disinkronkan dengan isi *datafile*
3. *Inactive, group* yang sekarang digunakan sedang tidak dipakai dan isinya sudah disinkronkan dengan isi *datafile*
4. *Unused, group* yang baru saja di*create* dan beli digunakan.

DBWR atau *Database Writer* bertugas untuk menulis semua *data* yang telah berubah dari *database buffer cache* kedalam *datafile* dan menyimpan *data* yang sering digunakan dengan menggunakan metode LRU atau Least Recently Used.

SMON atau *System Monitor* bertugas untuk melakukan *automatic instance recovery*.

PMON atau *Process Monitor* bertugas untuk me*rollback* semua transaksi yang belum di *commit* oleh *user*.

LGWR atau Log *Writer* bertugas untuk melakukan penulisan isi *redo log file buffer* kedalam *online redo log file* ketika terjadi beberapa hal seperti berikut ini:

1. Setiap 3 detik
2. Jika *redo entries* mencapai 1 MB
3. Jika 1/3 *redo*log *buffer full*
4. Ketika *user* melakukan *commit*.

CKPT atau *Check Point* bertugas untuk memberitahu DBWR untuk melakukan penulisan kedalam *disk*.

Listener adalah sebuah fitur oracle yang dapat digunakan untuk *user remote* atau kita dapat membuka *database* yang berada pada kantor dalam jarak yang jauh misalnya membuka *database* kantor di rumah. *Listener* dapat dibuat dengan menggunakan GUI atau *wizard* dan juga dapat menjalankannya dengan menggunakan *command line*. *Listener* dibagi menjadi dua yaitu:

1. *Listener Default* adalah *listener* yang telah disediakan oleh oracle dan di*create* secara otomatis ketika membuat *database*. *Port* ini akan selalu 1521.
2. *Listener* Non *Default* adalah *listener* yang dibuat manual oleh *user* dengan *port* yang bebas.

Cara membuat *listener* adalah sebagai berikut ini:

1. Ketikkan Netca pada *command line* dan akan muncul sebuah tampilan sebagai berikut ini:

Gambar 3.29 *Create Lisetener Step 1*

2. Lalu pilih *listener configuration* lalu *next* dan pilih add lalu tekan *next* di *form* tersebut

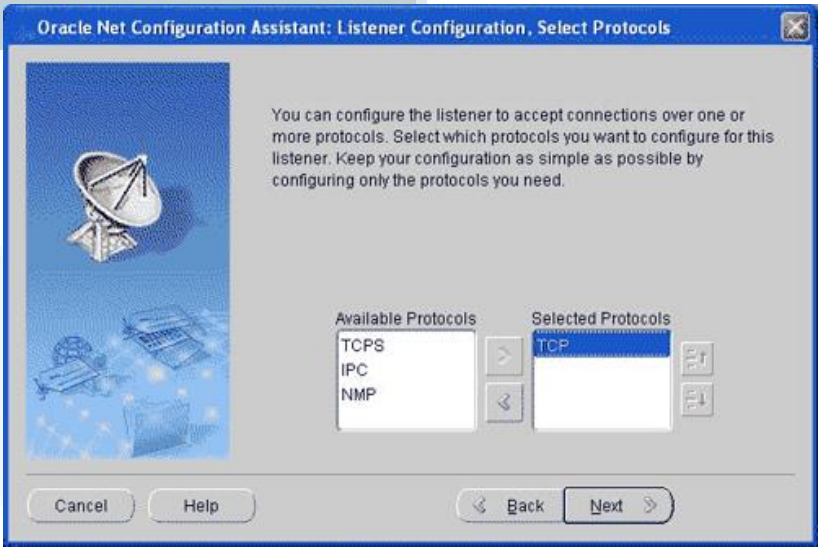
Gambar 3.30 *Create Lisetener Step 2*

3.

3. Lalu masukkan nama *listener* yang diinginkan

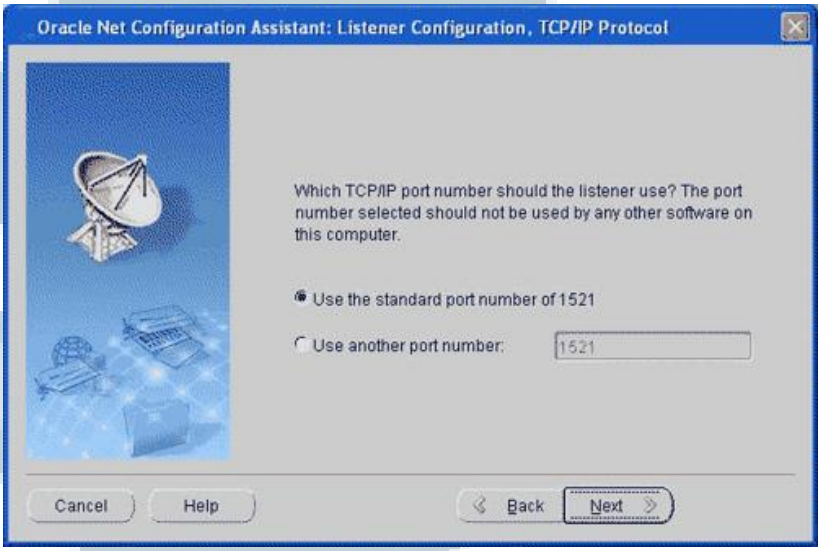
Gambar 3.31 *Create Lisetener Step 3*

4. Setelah itu akan muncul sebuah *form select protocol* dan pilih TCP/IP



Gambar 3.32 *Create Lisetener Step 4*

5. Setelah itu akan muncul sebuah *form* terkait dengan *port* number. Hal ini akan secara otomati terisi dengan *port* 1521. Apabila ingin menggunakan *port* lainnya masukkan *IP address* yang sesuai dengan *port* dari komputer yang dingin dikoneksikan.



Gambar 3.33 *Create Lisetener Step 5*

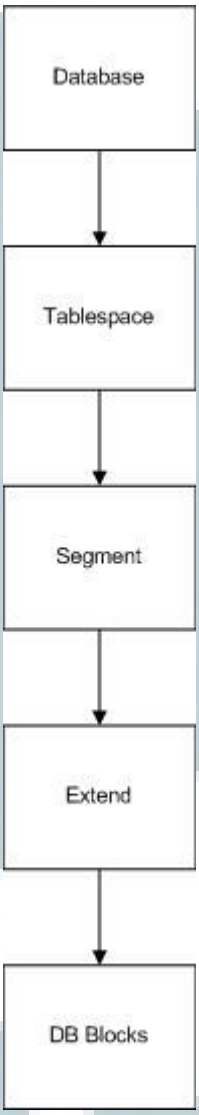
6. Lalu pilih *next*, apabila ingin membuat *listener* lainnya pilih *yes* dan jika tidak pilih *no*.

Gambar 3.34 *Create Lisetener Step 6*

7. Tekan *next* maka proses pembuatan *listener* sudah selesai.

Gambar 3.35 *Create Lisetener Step 7*

Tablespace merupakan bagian dari arsitektur *database* oracle. *Database* oracle memiliki struktur sebagai berikut ini:



Gambar 3.36 *Architecture Database*

Tablespace adalah *space* yang digunakan untuk menyimpan *object* dari *database*.

Segment adalah *object* dari *database* tersebut.

Extent adalah kumpulan *database* block yang berdekatan dan membentuk sebuah *segment*.

DB Blocks adalah satuan penyimpanan terkecil dalam *database*.

Secara fisik, *tablespace* terdiri atas satu atau lebih *datafile*. In*form*asi tentang *tablespace* ada di *view* v$*tablespace*, *dba_tablespace*s, *dba_data_files*, *dba*_temp*_files*, dll. *Tablespace* dibagi menjadi 3 tipe yaitu:

1. *Undo Tablespace* yang digunakan untuk menyimpan *rollback segment*. *Data* akan dihapus otomatis jika;
   - Sudah tidak dipakai
   - *Retention* sudah habis
   - Oracle membutuhkan *space*
2. *Temporary Tablespace* yang digunakan utnuk menyimpan *temporary segment*.
3. Permanent *Table*scape yang digunakan untuk menyimpan *segment* seperti *table* dan *index*.

Berdasarkan statusnya *tablespace* dibagi menjadi 3 yaitu

1. *Online* adalah kondisi dimana *tablespace* dapat dibaca dan ditulis.

2. Off*line* adalah kondisi dimana *tablespace* tidak dapat dibaca dan ditulis

3. *Read only* adalah kondisi dimana *tablespace* hanya dapat dibaca.

Berdasarkan jumlah *file*nya *tablespace* dibagi menjadi dua yaitu:

1. *Small File*, *tablespace* yang terdiri lebih dari 1 *datafile* dan ukuran maksimal dibatasi oleh OS rata-rata 32 GB per*file*, biasanya digunakan untuk OLTP.

2. *Big File*, *tablespace* terdiri dan maksimalnya hanya terdapat 1 *datafile* dengan ukuran yang bisa mencapai 20GB, biasanya digunakan untuk OLAP.

Cara membuat *tablespace*:

- *Undo Tablespace*

```
SQL>create undo tablespace undotbs2
datafile'/oradata/oracle/ts_bak/und
otbs201.dbf' size 10m;
```

- *Temporary Tablespace*

```
SQL> create temporary tablespace
temp2
tempfile '/oradata/oracle/ts/temp21
.dbf' size 10m;
```

- Permanent *Tablespace*

```
SQL> create tablespace DATA
datafile
'/oradata/oracle/ts_bak/data01.dbf'
size 10m;
```

Cara menambah size dari *table space*:

- *Undo Tablespace*

```
SQL> alter database
datafile
'/oradata/oracle/ts_bak/undotbs201.
dbf' resize 20m;
```

- *Temporary Tablespace*

```
SQL> alter database
tempfile
'/oradata/oracle/ts/temp21.dbf'
resize 20m;
```

- Permanent *Tablespace*

```
SQL> alter database
datafile
'/oradata/oracle/ts_bak/data01.dbf'
resize 20m;
```

Cara melihat *datafile* dan size dari *tablespace*

```
SQL> select file_name,bytes from
dba_data_files
where tablespace_name='UNDOTBS2';
```

Cara menghapus *tablespace*:

```
SQL> drop tablespace DATA;
```

Scheduler adalah pekerjaan yang dilakukan secara berulang kali secara otomatis dalam jangka waktu tertentu. *Scheduler* miliki 3 *job* yaitu:

1. *PLSQL_BLOCK* adalah untuk menjalankan *sql* dan *anonymous block*
2. *EXECUTABLES* adalah *job* yang akan menjalankan *script* .
3. *STORED_PROCEDURE* adalah *job* untuk menjalankan *procedure*, *function*, dan *package.*

Cara membuat *Scheduler* adalah sebagai berikut ini:\

```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB
    (
        job_name=>'INSERT_PER_3_DETIK',
        job_type=>'PLSQL_BLOCK',
        job_action=>'INSERT INTO test_sched
values(systimestamp);commit;',
```

```
repeat_interval=>'FREQ=SECONDLY;INTERVAL=3',
  start_date=>sysdate
 );
end;
/
exec dbms_scheduler.enable ('INSERT_PER_3_DETIK');
select * from test_sched;
exec dbms_scheduler.disable ('INSERT_PER_3_DETIK');
```

3.3.5. Kendala yang Ditemukan

Kendala yang ditemui penulis pada saat melakukan kerja magang dan *training* pada PT. Oracle Indonesia adalah:

- Penulis sering kali tertinggal materi *training*.
- Kesulitan dalam mengikuti materi *training* karena ada beberapa pengertian yang tidak dimengerti oleh penulis.
- Kesulitan pengaturan waktu dalam hal mendengarkan, mencatat, dan mengikuti praktikum selama kegiatan training.
- *Trainer* terkadang menjelaskan materi secara cepat. Hal ini membuat penulis kesulitan dalam mengikuti materi yang diajarkan.

3.3.6. Solusi atas Kendala yang Ditemukan

Solusi atas kendala yang dialami penulis selama masa kerja magang dan *training* adalah:

- Penulis datang lebih awal ke kantor untuk dapat mempelajari materi yang tertinggal.
- Penulis mencatat istilah yang tidak dimengerti dan melakukan pencarian istilah tersebut di mesin pencarian.

- Penulis menyesuaikan diri selama mengikuti masa *training* agar dapat melakukan *multitasking*.
- Kesulitan mengikuti *trainer* dapat diatasi dengan melihat dan membuka materi terkait dengan *training* yang diajarkan.