MODUL PRAKTIKUM
SI0054 BASIS DATA LANJUT

OLEH

NURAINI PURWANDARI S.T., M.M.S.I.
MIRA ZIVERIA, S.Si, M.T.

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER & ILMU KOMUNIKASI
INSTITUT TEKNOLOGI DAN BISNIS KALBIS
2018

KATA PENGANTAR

Segala puji bagi Allah SWT karena dengan karunia-Nya Modul Praktikum SI0014 Perancangan Basis Data ini dapat diselesaikan.

Modul ini disusun berdasarkan Garis-garis Besar Proses Pembelajaran (GBPP) mata kuliah SI0014 Perancangan Basis Data, khususnya bagian Rencana Pembelajaran Praktikum. Modul ini bertujuan untuk membantu mahasiswa Kalbis Institute yang mengambil mata kuliah SI0014 Perancangan Basis Data khususnya kelas praktikum yaitu mahasiswa prodi Sistem Informasi semester 2 dan mahasiswa Teknik Informatika semester 4.

Kami sangat mengharapkan masukan yang positif dari para pengajar maupun mahasiswa untuk kesempurnaan modul ini, yang dapat menjadi bahan pertimbangan sebagai bahan perbaikan di edisi berikutnya. Semoga modul praktikum ini dapat memberikan manfaat dalam memahami materi perkuliahan praktikum Perancangan Basis Data di Kalbis Institute.

Jakarta, 10 Maret 2018
Penulis

DAFTAR ISI

1. INSTALASI ORACLE

1.1 PENDAHULUAN

Secara umum, panduan ini berlaku untuk semua instalasi Oracle. Di semua OS baik Windows maupun Unix (Sun Solaris, IBM AIX, HP UX, Linux, dan lain-lain) proses instalasi itu sama, hanya sedikit berbeda di pre-installation requisite-nya.

1.2 HAL-HAL YANG PERLU DIPERHATIKAN SEBELUM PROSES INTALASI

Secara umum, berikut ini spesifikasi yang diminta oleh instalasi tipe ini. Lebih detail tentang spesifikasi komputer yang bisa di-install, lihat dokumentasi (installation guide) yang ada di paket software yang telah di download; saya cantumkan juga direferensi. Dalam contoh ini instalasi dilakukan pada windows XP service pack 2.

Hardware

- Physical memory (RAM) : 256 MB minimum, 512 MB recommended
- Virtual memory: dua kali RAM
- Disk space: kira-kira 5 G
- Video (monitor) adapter: 256 colors
- Processor : 550 MHz minimum

Operating system (OS)

- Windows 2000 with service pack 1 or later. All editions, including Terminal
- Services and Microsoft Windows 2000 MultiLanguage Edition (MLE)
- Windows Server 2003 - all editions
- Windows XP Professional
- Windows NT is not supported

Pilihan instalasi Database Oracle 10g

1. Enterprise Editon : Merupakan pilihan instalasi yang membutuhkan lisensi. Selain fitur-fitur standar yang dimiliki oracle, konfigurasi database dan management tool serta fitur untuk data warehousing dan transaction processing disertakan dalam pilihan instalasi ini.

2. Standard Edition : melakukan instalasi management tool, distribusi penuh, replikasi, fitur-fitur untuk web, dan fasilitas-fasilitas untuk membangun aplikasi.

3. Personal Edition : melakukan instalasi fitur yang sama seperti pada Enterprise Edition tetapi hanya mendukung single user development dan deployment environment.

4. Custom : melakukan instalasi dengan cara memilih fitur-fitur yang dibutuhkan secara manual.

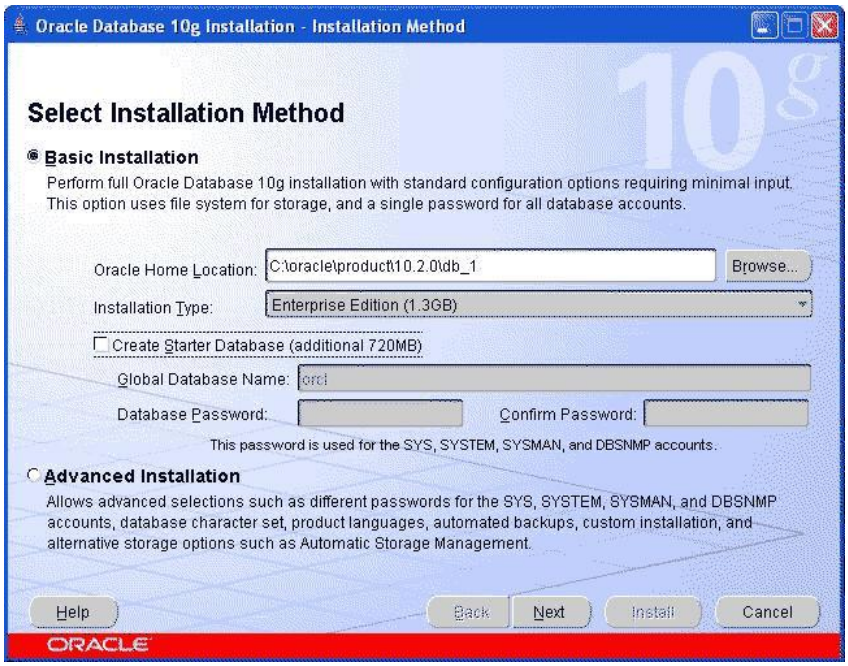
Hal-hal yang perlu diperhatikan pada saat instalasi Oracle 10g

Ketika setup meminta memasukkan password ada beberapa hal yang perlu diperhatikan antara lain:

1. Pastikan panjang password 4 sampai dengan 30 karakter.
2. Sertakan database character set seperti underscore(_), dollar ($), dan pound sign (#) karena akan dapat memperkuat password.
3. Jangan memulai password dengan karakter numeric
4. Jangan menggunakan username untuk password
5. Jangan menggunakan istilah-istilah yang sudah ada di oracle untuk password.
6. Jangan menggunakan password change_on_install untuk account SYS.
7. Jangan mengunakan password manager untuk account SYSTEM.
8. Jangan menggunakan password sysman untuk account SYSMAN.
9. Jangan menggunakan password dbsnmp untuk account DBSNMP.
10. Pastikan password terdiri dari karakter alphabet, numeric, dan punctuation mark character.
11. Jangan menggunakan kata-kata sederhana untuk password seperti welcome, account, database, user, dll.
12. Jangan mengubah-ubah setingan Java Runtime Environment (JRE) yang digunakan oleh oracle.
13. Jika tipe instalasi yang anda pilih mengharuskan untuk melakukan konfigurasi Database Configuration Assistant dan Oracle Net
14. Configuration Assistant secara manual anda harus mendefinisikan secara detail mengenai konfigurasi database dan jaringan.

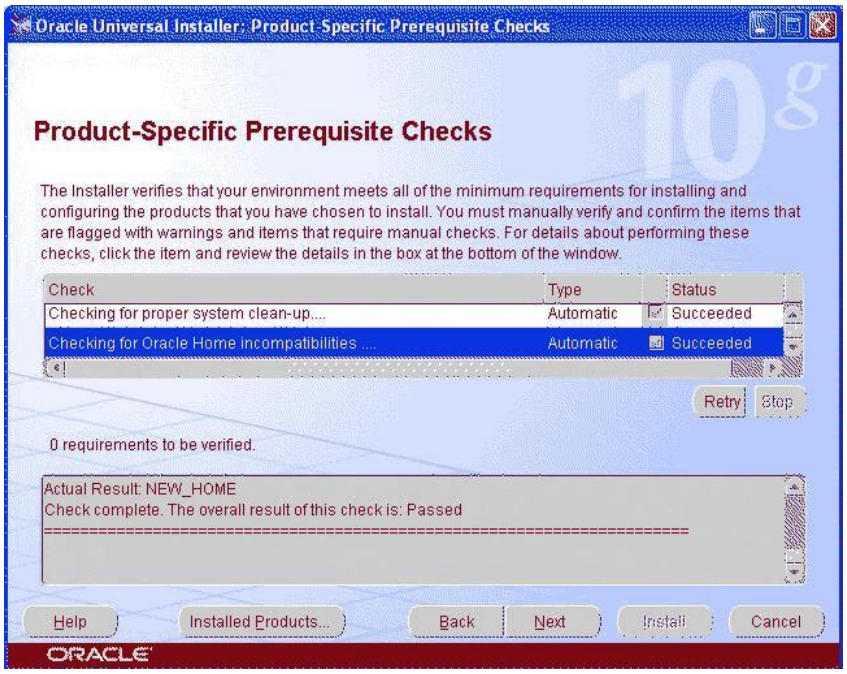
1.3 LANGKAH-LANGKAH INSTALASI

1. Jalankan command “setup.exe” yang ada di paket software yang telah di download. Kemudian muncul Install wizard (GUI). Lihat gambarnya di sini



Pilih option “Basic Installation”
Masukkan directory “Oracle Home Location”
Pilih “Installation Type”
Jangan pilih “Create Starter Database”
Klik button “Next”

2. Oracle installer akan mengecek OS kita, apakah requirement-nya dipenuhi atau tidak. Lihat gambarnya di sini

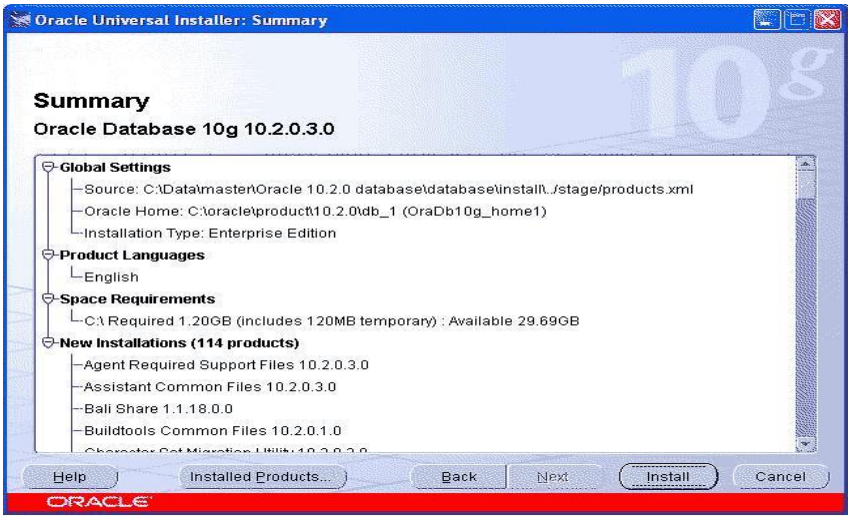


Pastikan semua statusnya “Succeeded”. Kalau ada warning, atau statusnya bukan Succeed, bereskan dulu OS-nya. Kemudian klik button “Next”

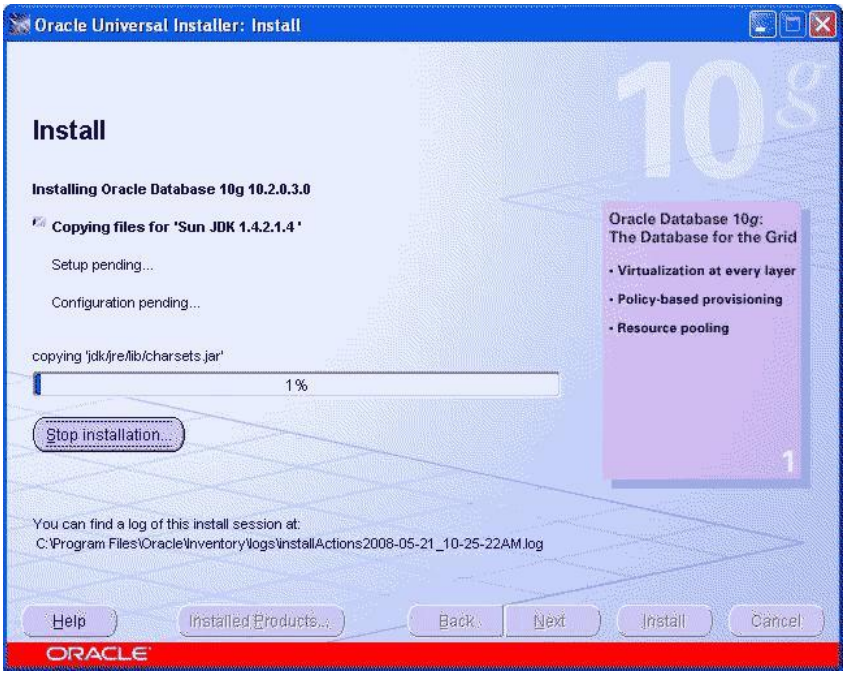
3. Dalam proses installasi, Oracle akan menjalankan program java. Bila firewall PC anda memblock java, dan muncul alert “Windows Security Alert”, klik tombol “Unblock”. Lihat gambarnya di bawah ini

4. Muncul summary komponen Oracle Database 10g yang siap kita install.

   Kemudian klik tombol “Install”. Lihat gambarnya di bawah ini

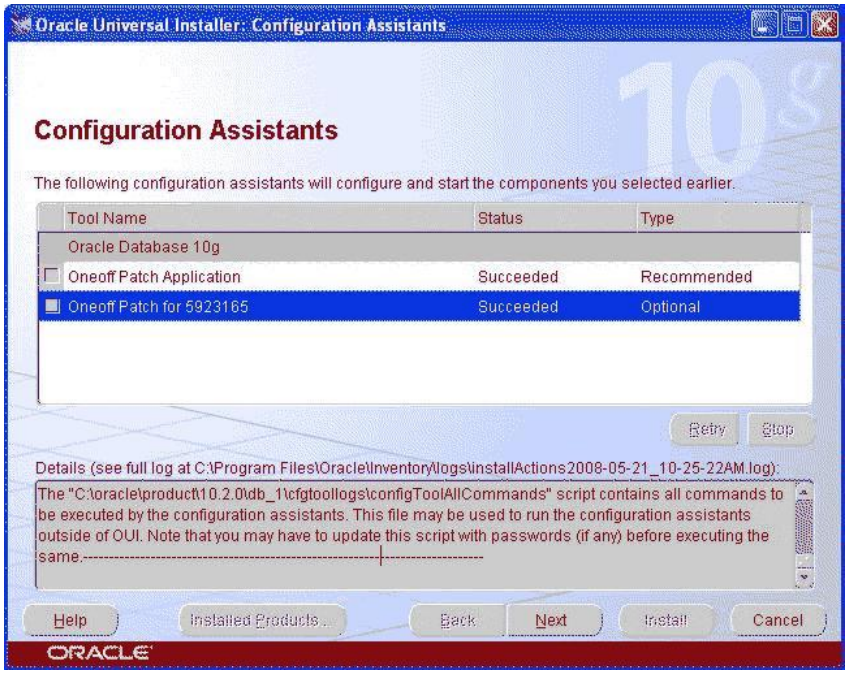


5. Installation progress. Lihat gambarnya di bawah ini



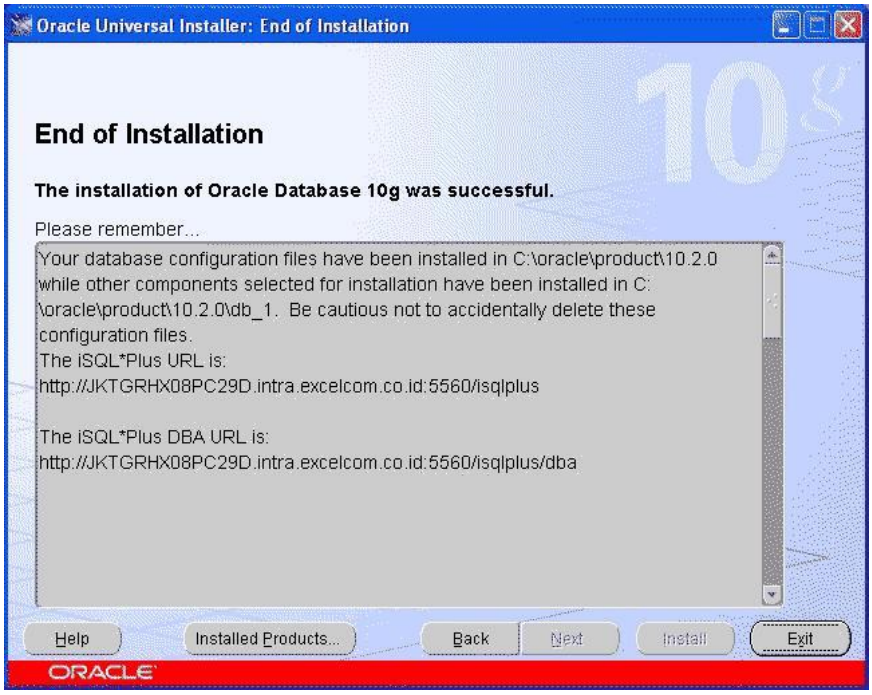
6. Setelah itu akan dilakukan konfigurasi tambahan oleh Oracle Installer. Kita cukup perhatikan saja. Setelah konfigurasi selesai, klik tombol “Next”.

   Kadang-kadang kita tidak perlu kilk tombol Next tersebut karena secara otomatis wizard menuju ke berikutnya. Lihat gambarnya di bawah ini



7. Akhirnya instalasi selesai. Setelah itu klik tombol “Exit”.

   Lihat gambarnya di bawah ini



1.4 LATIHAN

1. Lakukan instalasi Oracle 10g berdasarkan langkah-langkah yang telah dijelaskan diatas, kemudian masuk ke SQL Plus dan login sebagai user SYSTEM!

2. Jelaskan perbedaan antara Oracle 10g dan Oracle 10g XE!

2. SQL ORACLE

Sebuah sistem basis data menyediakan dua macam bahasa yang berbeda. Pertama untuk menspesifikasikan skema basis data (DDL) dan yang kedua untuk melakukan perubahan dalam basis data (DML). Selain itu juga ada select statement (*query*) yang digunakan untuk mengambil data yang ada di dalam database.

2.1 DATA DEFINITION LANGUAGE (DDL)

Data Definition Language ini berfungsi untuk mendefinisikan struktrur sebuah tabel, tipe data setiap field dan constraint yang berlaku pada data yang tersimpan di dalam sebuah tabel. Hasil dari kompilasi DDL adalah kumpulan tabel yang tersimpan pada sebuah file khusus yang disebut dengan kamus data (*data dictionary*) atau *data directory.*

Kamus data merupakan sebuah file yang berupa *metadata*, yaitu data tentang data. Kamus data ini akan selalu diakses pada suatu operasi basis data sebelum suatu file data yang sesungguhnya diakses.

DDL terdiri dari CREATE, ALTER, dan DROP yang akan dijelaskan sebagai berikut :

Membuat tabel (CREATE TABLE)

Query untuk membuat tabel (create table) adalah :

```
CREATE TABLE nama_tabel
          (
 nama field 1 tipe data,
nama field 2 tipe data,
  ……,
             nama_constraint
[ CONSTRAINT  PRIMARY KEY
   (nama_field)
  ]
   [ CONSTRAINT nama_constraint FOREIGN KEY
   (nama_field) REFERENCE nama_table
   (nama_field) ON DELETE CASCADE
   ]
 );
```

Keterangan :
nama_field1, nama_field2 adalah nama kolom (field) yang akan ada di dalam tabel.
tipe_data adalah tipe data yang bisa ditampung oleh field. **tipe_constraint** adalah atasan yang menentukan suatu field. **Tanda [ ]** menyatakan bahwa yang didalamnya boleh ada atau tidak.

Macam – macam tipe constraint yang mungkin adalah :

NULL atau NOT NULL

Constraint NOT NULL ini berfungsi untuk menjamin field harus diisi sedangkan constraint NULL memperbolehkan suatu field untuk dikosongi

UNIQUE

Constraint ini berfungsi untuk menjamin bahwa nilai pada setiap record di dalam sebuah tabel adalah unik

PRIMARY KEY

FOREIGN KEY

CHECK

Constraint ini berfungsi untuk menjamin bahwa nilai yang akan dimasukkan ke dalam sebuah field sudah sesuai dengan aturan yang dibuat

Contoh : pembuatan tabel mahasiswa dengan atribut nim bertipe char(9) dan nama bertipe varchar(20) yang tidak boleh kosong.

```
Sintaks : CREATE TABLE mahasiswa
          (
               Nim char(9),
               Nama varchar(20) not null,
               CONSTRAINT pk_mhs PRIMARY KEY (Nim)
          );
```

Memanipulasi tabel (ALTER TABLE)

Sebuah struktur tabel yang sudah didefinisikan dapat dimodifikasi dengan menggunakan perintah ALTER TABLE.

Ada beberapa jenis modifikasi pada tabel :

1. Menambahkan constraint baru untuk sebuah tabel Query :
   ALTER TABLE nama_tabel ADD CONSTRAINT nama_constraint_baru (definisi_constraint);

   Contoh : (misal pada tabel mahasiswa belum ada primary key)

```
SQL >    ALTER TABLE mahasiswa
         ADD CONSTRAINT pk_mhs PRIMARY KEY (Nim);
```

2. Menghapus sebuah constraint di dalam tabel. Query :

```
ALTER TABLE nama_tabel
DROP CONSTRAINT nama_constraint;
```

Contoh :

```
SQL > ALTER TABLE mahasiswa
      DROP CONSTRAINT pk_mhs;
```

3. Manambahkan field baru pada sebuah tabel. Query :

```
ALTER TABLE nama_tabel
ADD (nama_field  tipe_data  [definisi_konstraint]);
```

Contoh :

```
SQL > ALTER TABLE mahasiswa
ADD (alamat varchar(50) NOT NULL);
```

4. Mengubah definisi field yang telah ada pada sebuah tabel.
   Query :

```
ALTER TABLE nama_tabel
MODIFY(nama_field  tipe_field  definisi_konstraint);
```

Contoh :

```
SQL > ALTER TABLE mahasiswa
MODIFY(alamat varchar(25) not null);
```

Menghapus tabel (DROP TABLE)

Query untuk menghapus tabel (drop table) adalah :

```
DROP TABLE nama_table [CASCADE CONSTRAINT];
```

CASCADE CONSTRAINT :
akan menghapus semua constraint yang terhubung dengan tabel yang dihapus.
Contoh :

```
SQL > DROP TABLE buku;
```

2.2 DATA MANIPULATION LANGUAGE (DML)

Level abstraksi yang telah dibahas sebelumnya tidak hanya berlaku pada definisi atau struktur data tetapi juga pada manipulasi data. Manipulasi data itu sendiri dapat berupa:

- Pemasukan informasi baru ke dalam basis data (insert)
- Penghapusan informasi dari basis data (delete)
- Modifikasi informasi yang tersimpan pada basis data (update)

DML merupakan bahasa yang memungkinkan user untuk mengakses atau memanipulasi data

sebagaimana telah direpresentasikan oleh model data. Terdapat dua macam DML, yaitu:

Prosedural, mengharuskan user untuk menentukan data apa yang dibutuhkan dan bagaimana untuk mendapatkan data tersebut. **Nonprosedural,** mengharuskan pemakai untuk menentukan data apa yang dibutuhkan tanpa menyebutkan bagaimana mendapatkan data tersebut.

Sintaks-sintaks pada DML antara lain : INSERT RECORD
Insert digunakan untuk menyisipkan sebuah record ke dalam sebuah tabel.

Query :

```
INSERT INTO nama_tabel [(nama_field1, nama_field2, ….)]
VALUES (nilai1, nilai2, …);
```

Perlu diperhatikan bahwa nama_field1, nama_field2, … dapat dihilangkan. Hal ini dapat dilakukan bila nilai yang dimasukkan user, urutan sesuai dengan urutan field di dalam tabel tersebut, dan jumlah nilai yang dimasukkan juga harus sama dengan jumlah field pada tabel.

Selain itu, proses penyisipan juga harus mematuhi aturan constraint yang telah didefinisikan pada sebuah tabel
Contoh :

```
insert into mahasiswa values('30108001', 'paijo', 'skp');
                    atau
insert into mahasiswa (nim,nama,alamat) values('30108001', paijo', 'skp');
```

UPDATE RECORD

Setiap nilai record yang telah disimpan di dalam tabel dapat dimodifikasi kembali berdasarkan kondisi tertentu. Jika kondisi tidak didefinisikan, maka nilai semua record akan terupdate

Query :

```
UPDATE nama_tabel
       SET    nama_field1 = nilai_baru1,
              nama_field2 = nilai_baru2,
              ….
       [WHERE kondisi];
```

Contoh :

```
SQL >     update mahasiswa
          set nama  = 'paimin', alamat  = 'skb'
          where nim = '30108001';
```

DELETE RECORD

Record yang telah disimpan di dalam sebuah tabel dapat dihapus berdasarkan kondisi tertentu. Jika kondisi untuk menghapus tidak didefinisikan maka seluruh record pada tabel tersebut akan dihapus.

Query :

```
DELETE nama_tabel [WHERE kondisi];
```

Contoh :

```
SQL >     delete mahasiswa
          where nim = '30108001';
```

(menghapus sebuah record pada tabel mahasiswa dimana nimnya „30108001")

```
SQL >     delete mahasiswa;
          (menghapus semua record pada table mahasiswa)
```

Select Statement

Statement SELECT digunakan untuk mengambil record dari sebuah tabel atau lebih. Record yang diambil dengan SELECT dapat disaring dengan menggunakan kondisi yang terdefinisi.

Statement SELECT mempunyai format sebagai berikut :

```
SELECT [DISTINCT | ALL]  * | nama_field 1, nama_field 2,……
       FROM nama_tabel [WHERE kondisi1]
       [GROUP BY nama_field] [HAVING kondisi2]
       [ORDER BY nama_field [ASC | DSC]];
```

DISTINCT digunakan untuk mengambil record yang nilainya tidak ganda. **FROM** digunakan

untuk mendefinisikan tabel yang menjadi sumber data dari suatu perintah SELECT.

WHERE digunakan untuk mendefinisikan kondisi pengambilan data pada tabel yang disebutkan di klausa FROM.

GROUP BY digunakan untuk mengelompokkan baris – baris data berdasarkan ekspresi group – group tertentu yaitu untuk field tertentu.

HAVING digunakan untuk memilih / mendefinisikan kriteria kelompok group yang akan ditampilkan berdasarkan group yang akan dibuat. Penggunaa klausa HAVING dipakai sebagai pengganti klausa WHERE pada GROUP BY.

ORDER BY digunakan untuk mengurutkan seleksi berdasarkan kondisi yang diinginkan.

ASC (ascending) berarti pengurutan menaik – 1, 2, 3, 4 …. DSC (descending) berarti pengurutan menurun – 10, 7, 4, 2, ….

Untuk mengambil semua field pada setiap record, maka ganti semua nama field menjadi tanda bintang „*“ (tanpa tanda kutip)

```
SELECT *
FROM nama_tabel

[WHERE kondisi];
```

Contoh : SELECT * FROM mahasiswa;

Untuk mengambil record yang memenuhi kondisi tertentu maka statement SELECT digabung dengan klausa WHERE. Format WHERE :

```
WHERE  nama_field operator_kondisional nilai
```

Operator kondisional yang diganakan adalah :

Operator	Arti
=	Sama dengan
<>, !=, ^=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan
BETWEEN … AND …	Mengambil nilai yang berada pada antara 2 buah bilangan.
IN	Mengetes nilai - nilai pada list yang disepsifikasikan
LIKE	Menyocokkan sebuah pola karekter
IS NULL	Apakah sebuah nilai null?

Untuk menggabungkan beberapa kondisi didalam klausa WHERE, maka digunakan operator boolean **AND, OR, dan NOT**.

AND : apabila setiap kondisi harus dipenuhi.
OR : apabila cukup salah satu kondisi yang terpenuhi.
NOT : apabila merupakan kebalikan / negasi dari kondisi yang dipenuhi.

Contoh :

Select * from mahasiswa where nama not „paimen“;

BETWEEN … AND …

Untuk mengambil record dengan nilai field berada pada batas tertentu, maka digunakan statement BETWEEN atau NOT BETWEEN.
Statement ini bergabung menjadi satu dengan klausa WHERE.

```
WHERE  nama_field BETWEEN nilai_awal AND nilai_akhir

WHERE nama_field NOT BETWEEN nilai_awal AND nilai_akhir
```

IN dan NOT IN

Untuk menampilkan record yang nilai fieldnya berada pada suatu kelompok nilai tertentu maka gunakan statement : IN atau NOT IN yang dipakai bersama klausa WHERE

```
WHERE nama_field IN (nilai1, nilai2, nilai3, …)

WHERE nama_field NOT IN (nilai1, nilai2, nilai3, …)
```

LIKE dan NOT LIKE

```
WHERE nama_field LIKE pola;
WHERE nama_field NOT LIKE pola;
```

Untuk menampilkan record yang nilai fieldnya mengandung nilai tertentu, digunakan statement LIKE atau NOT LIKE yang diletakkan pada klausa WHERE.

Pola adalah karekter atau string, yang dapat dipadukan dengan dua macam wild character, yaitu :
% : mewakili 0,1 atau beberapa karakter
_ : mewakili tempat sebuah karakter.

Contoh :
1. %JO%
   String yang sesuai dengan %JO% adalah : **JO**HAN, I**JO**, PAI**JO**, JO**JO**N
2. _ELO_
   String yang sesuai dengan _ELO_ adalah : B**ELO**K, K**ELO**K, B**ELO**N

Field yang dapat menggunakan statement LIKE atau NOT LIKE adalah field yang bertipe Char, Varchar, atau Varchar2.

IS NULL dan IS NOT NULL

```
WHERE nama_field IS NULL;
WHERE nama_field IS NOT NULL;
```

Salah satu kegunaan nilai NULL adalah untuk merepresentasikan sebuah nilai yang belum ada.

Dalam hal ini nilai field untuk sebuah record belum terisi. Sebuah nilai NULL tidak identik dengan spasi atau 0.

Untuk menampilkan record yang mempunyai nilai field NULL atau tidak, gunakan statement IS NULL atau IS NOT NULL yang dimasukkan dalam statement WHERE

OPERATOR ARITMATIKA

Ada kemungkinan, pada saat menampilkan nilai, kita ingin memodifikasi nilai yang ingin ditampilkan, misalnya kita ingin menampilkan harga barang, namun harga barang tersebut ditambah 500.

Untuk memodifikasi nilai tersebut kita gunakan operator aritmatika yang diletakkan pada klausa SELECT.

```
SELECT nama_field1 operator nilai_tambahan, …
       FROM nama_tabel [WHERE kondisi];
```

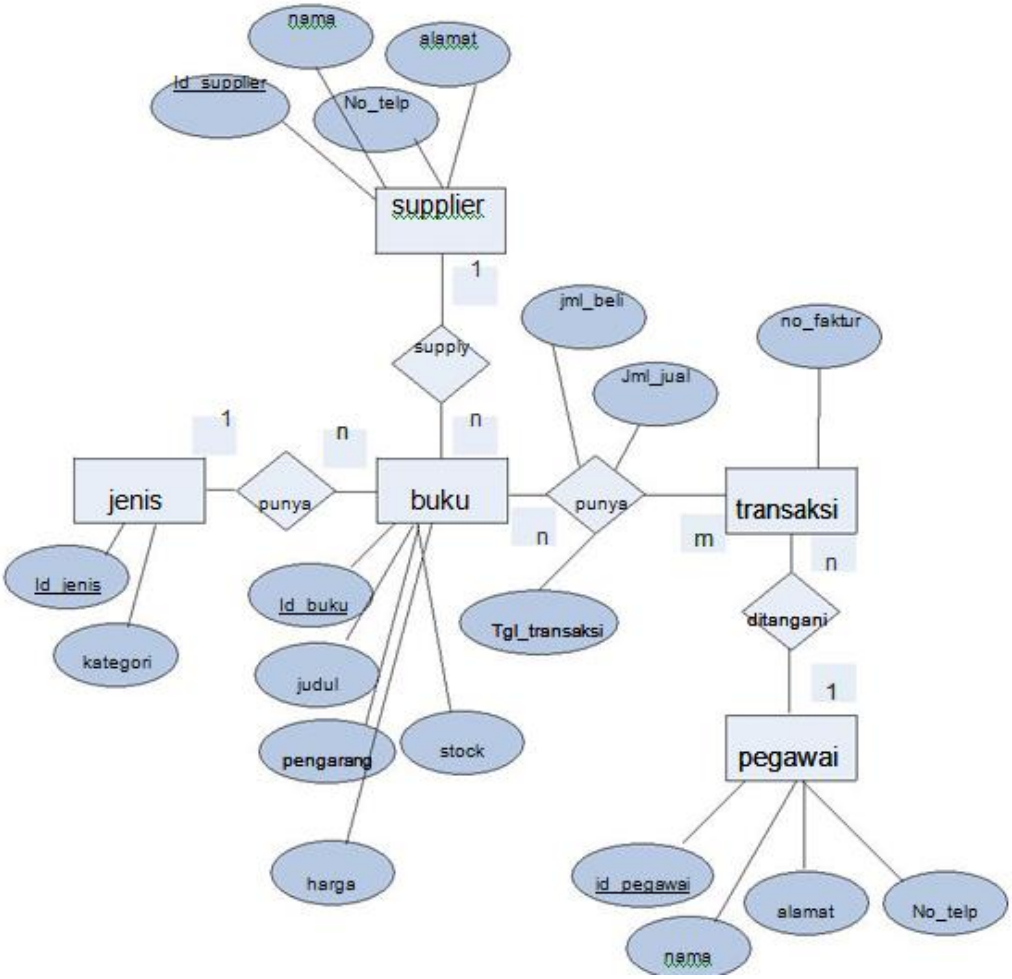
Operator yang digunakan adalah : *, /, +, dan –

Field yang bisa menggunakan operator ini harus bertipe numerik, date?

Contoh :

```
SQL > SELECT nim, spp + 50000 FROM keuangan;
```

2.3 LATIHAN STUDI KASUS TOKO BUKU



1. Berdasarkan ER Diagram diatas buatlah Data Definition Language (DDL)?
2. Masukkan data-data berikut ke tabel jenis !

ID JENIS	KATEGORI
AG	Agama
AK	Akuntansi
BH	Bahasa
BI	Biologi
EK	Ekonomi
FL	Filsafat
FI	Fisika
IT	Komputer dan Internet
MN	Manajemen
UM	Umum

3. Masukkan data-data berikut ke tabel supplier !

ID SUPPLIER	NAMA	NO TELEPON	ALAMAT
SP-001	PT Sidogiri	022-8768509	JL Ruwet Gg Buntet No.34 Bandung
SP-002	PT Moroseneng	021-8695465	JL Badak Jawa No. 6 Jakarta
SP-003	PT Suka-suka	031-4568698	JL Darmo No.7 Surabaya
SP-004	PT Sukabaca	022-9797966	JL Sukabirus no.9 Bandung

4. Masukkan data-data berikut ini ke dalam table buku!

ID BUKU	ID JENIS	ID SUPP-LIER	JUDUL	PENGARANG	HARGA	STOCK
BK-001	IT	SP-002	10 Langkah Belajar Logika & Algoritma Meng. Bahasa C & C++	Ema Utami	Rp27.375	10
BK-002	IT	SP-001	10 Mp.Lotus Notes 4.5	Jane Calabria	Rp9.180	34
BK-003	BH	SP-004	Kamus Bahasa Inggris	Jubilee Enterprise	Rp15.980	65
BK-004	IT	SP-001	101 Tip & Trik Adobe Photoshop Cs	Gregorius Agung	Rp16.830	26
BK-005	BI	SP-001	101 Tip & Trik Belajar Anatomi Manusia	Gregorius Agung	Rp19.380	84
BK-006	IT	SP-002	101 Tip & Trik Ms Access Project 2003 & Sql Server 2000	Hengky Alexander M.	Rp16.830	36
BK-007	MN	SP-002	Konsep-konsep Marketing	Anwar Khaidir	Rp12.580	26
BK-008	AG	SP-004	RENUNGAN JUM'AT	ABDURRAHMAN ARROISI	Rp31.280	6
BK-009	UM	SP-003	11 Langkah Menjadi Sahabat Ana	V. Dwiyani	Rp19.380	2
BK-010	IT	SP-002	12 Kreasi Logo Dengan Adobe Illustrator Cs2	Jubilee Enterprise	Rp27.880	88
BK-011	UM	SP-004	13 Cara Praktis Memetik Sukses	R. Herry Prasetyo	Rp16.830	25

3. FUNGSI-FUNGSI PADA SQL ORACLE

3.1 DUAL DAN GROUP BY

DUAL adalah sebuah built-in tabel yang disediakan oleh Oracle untuk memproses ekspresi aritmatika, logika, tanggal dan lain sebagainya.

Contoh :
SQL > SELECT 10 * 12 – 2 FROM DUAL;
Menampilkan hasil operasi aritmatika 10 * 12 – 2

SQL > SELECT sysdate FROM DUAL;
Menampilkan tanggal hari ini

GROUP BY … HAVING ….

GROUP BY …. HAVING … digunakan untuk menyeleksi himpunan yang dihitung berdasarkan sebuah fungsi agregasi (agregate function) sesuai dengan kondisi yang didefinisikan setel
ah HAVING.

SELECT fungsi_agregasi, ..
FROM nama_tabel

GROUP BY nama_field
[HAVING kondisi];

3.2 AGGREGATE FUNCTION

AVG (Average / rata - rata)

AVG berfungsi untuk mengembalikan nilai rata – rata dari sekumpulan nilai yang terdapat pada sekumpulan nilai record pada sebuah field.

Contoh :

SQL > SELECT AVG(donasi) Donasi FROM keuangan;

Donasi : merupakan alias dari hasil fungsi AVG(donasi)

Queri diatas digunakan untuk menampilkan donasi rata – rata mahasiswa yang terdapat di dalam tabel keuangan.

MAX (Maximum)

MAX berfungsi untuk mengembalikan nilai maksimum yang terdapat pada sekumpulan nilai pada sebuah field.

Contoh :

```
SQL >   SELECT MAX(donasi) “Donasi Terbesar”
        FROM keuangan;
```

Donasi Terbesar : merupakan alias dari hasil fungsi MAX(donasi)

Queri diatas digunakan untuk menampilkan nilai donasi yang paling besar di dalam tabel keuangan.

MIN (Minimum)

MIN berfungsi untuk mengambil nilai minimum yang terdapat pada sekumpulan nilai pada sebuah field.

Contoh :

```
SQL >   SELECT MIN(donasi) “Donasi Terkecil”
        FROM keuangan;
```

Donasi Terkecil : merupakan alias dari hasil fungsi MIN(donasi) Queri diatas digunakan untuk menampilkan nilai donasi yang paling kecil di dalam tabel keuangan.

SUM (jumlah / sigma)

SUM berfungsi untuk menghitung jumlah total nilai pada sebuah kolom tertentu.

Contoh :

```
SQL > SELECT SUM(donasi) “Total donasi” FROM keuangan;
```

Queri diatas digunakan untuk menampilkan total jumlah donasi untuk seluruh mahasiswa pada tabel keuangan.

COUNT (banyak record)

COUNT berfungsi untuk menghitung jumlah record yang memenuhi kondisi tertentu.
Contoh :

SQL > SELECT COUNT(*) “Donasi Manajemen Informatika” FROM keuangan
WHERE nim like „3010%“;

Queri diatas digunakan untuk menampilkan jumlah mahasiswa Manajemen Informatika yang memberikan donasi.

3.3 BUILT-IN FUNCTION

Built-In Function adalah fungsi yang disediakan oleh Oracle agar dapat digunakan di dalam SQL *Plus maupun oleh program eksternal.
Fungsi ini terdiri dari beberapa kelompok, yaitu :

Fungsi Aritmatika

Fungsi	Kegunaan	Contoh
ABS	Mengembalikan nilai absolute	ABS(-10) = 10
CEIL	Mengembalikan bilangan bulat terbesar	CEIL(4.2) = 5 CEIL(-4.2) = -4
EXP	Mengembalikan nilai pemangkatan bilangan natural (e^x)	EXP(0) = 1 EXP(1) = 2.718
FLOOR	Mengembalikan bilangan bulat terkecil	FLOOR(4.2) = 4 FLOOR(-4.2) = -5
LN	Mengembalikan nilai logaritma natural	LN(2) = 0.693147181
LOG	Mengembalikan nilai logaritma	LOG(10,10) =1
MOD	Menghasilkan sisa pembagian	MOD(4,2) = 0 MOD(7,2) = 1
POWER	Mengembalikan nilai pangkat	POWER(2,3) = 8
ROUND	Mengembalikan bilangan pembulatan	ROUND(5.56,1) = 5.6
SIGN	Mengembalikan nilai positif, negative, atau nol. Nilai balikan sign	SIGN(-6) = -1 SIGN(3) = 1

	ada tiga yaitu 1 jika x>0 0 jika x=0 -1 jika x<0	

SIN, COS, TAN, SINH, COSH,TANH	Fungsi trigonometri	SIN(0) = 0 COS(90) = 0
SQRT	Mengembalikan nilai akar	SQRT(16) = 4
TRUNC	Mengembalikan nilai yang telah dipotong	TRUNC(5.671,2) = 5.67

Fungsi Karakter / String

Fungsi	Kegunaan	Contoh
ASCII(karakter)	Memgembalikan nilai ASCII dari karakter	ASCI(„A") = 65
CHR(nilai_ASCII)	Memgembalikan karakter dari sebuah nilai ASCII	CHR(65) = „A"
CONCAT(teks1, teks2)	Menggabungkan teks1 dan teks2	CONCAT (“sate”, ”ayam”) = “sate ayam”
DECODE(field, kode_decode)	Menggantikan nilai yang terdapat di dalam field dengan nilai lain.	DECODE(bulan,1, „Januari", 2,"Februari", 3,"Maret", 4,"April", 5,"Mei", 6,"Juni", „Bulan Lain") bulan
GREATEST(nilai1, nilai2, nilai3, …)	Mengembalikan nilai terbesar dari sederetan nilai	GREATEST (3,10,15,4) = 15
INITCAP(String)	Mengembalikan string yang terdiri dari huruf kapital pada setiap kata.	INITCAP(„pOLItek Nik tElkoM") = Politeknik Telkom
INSTR (teks1, teks2)	Mencari posisi teks2 yang terdapat di dalam teks1.	INSTR ("SQL n PL/SQL", "SQL") = 1

INSTR(teks1, teks2, i) INSTR(teks1, teks2, 1, n)	Pencarian dapat dimulai dari posisi ke-i. n menyatakan pengulangan yang ke-n kali dari teks2 yang terdapat di dalam teks1.	INSTR ("SQL n PL/SQL", "SQL", 4) = 10 INSTR ("SQL n PL/SQL", "SQL",1,2) = 10
LEAST(nila1, nilai2, nilai3,…)	Mengembalikan nilai terkecil dari sederetan nilai	LEAST(2,2,5,1,6) = 1
LENGTH(String)	Mengembalikan panjang String	LENGTH("1233") = 4
LOWER(String)	Mengubah String menjadi huruf kecil semuanya	LOWER ("PoLTeK") = poltek
LPAD (teks1, n , teks2)	Menyisipkan karakter teks2 ke dalam teks1 untuk karakter kosong sepanjang n disebelah kiri teks1	LPAD („Oracle", 10 , „/") = ////Oracle
LTRIM (teks1, teks2)	Menghapus karakter-karakter pada bagian kiri teks1 sehingga tidak diawali dengan sembarang karakter pada teks2. Default teks2 adalah spasi	LTRIM („xxxOracle","x") = Oracle
REPLACE (teks1, teks2, teks3)	Menggantikan kemunculan karakter teks2 di dalam teks1 dengan teks3. Bila teks3 tidak disebutkan maka teks2 pada teks1 akan dihapus karena default teks3 null	REPLACE („siswa","a","i") = siswi

RPAD (teks1, n, teks2)	Menyisipkan karakter teks2 ke dalam teks1 untuk karakter kosong sepanjang n disebelah kanan teks1	RPAD („Oracle", 10 , „/") = Oracle////
RTRIM (teks1, teks2)	Menghapus karakter-karakter pada bagian kanan teks1 sehingga tidak diakhiri dengan sembarang karakter pada teks2. Default teks2 adalah spasi	RTRIM ('123000', '0') = 123

SUBSTR(Teks,i) SUBSTR(Teks, i, n)	Mengambil karakter pada string teks dimulai dari posisi ke-i (dari kiri ke kanan) sebanyak n buah. Jika i bernilai negatif, maka posisi ke-i dimulai dari kanan ke kiri	SUBSTR("ABCD", 2) = "BCD" SUBSTR ("ABCD", -3, 2) = "BC"
TRANSLATE (teks1, teks2, teks3)	Menggantikan kemunculan karakter teks2 di dalam teks1 dengan teks3. perbedaannya dengan REPLACE adalah karakter yang digantikan dilakukan secara individual	
UPER(String)	Mengubah String menjadi huruf besar semuanya	UPER("PoLTeK") = POLTEK

Fungsi Tanggal

Fungsi	Kegunaan	Contoh
ADD_MONTHS (tanggal, n)	Menambah atau mengurangi tanggal terhadap n	sysdate =06-JUL-05 ADD_MONTHS (sysdate, 2) = 08-JUL-05 ADD_MONTHS (sysdate, -2) = 04-JUL-05
LAST_DAY (tanggal)	Menghasilkan tanggal terakhir pada sebuah bulan	LAST_DAY (sysdate) = 31-JUL-05
MONTHS_BETWEEN (tanggal 2, tanggal1)	Mengahsilkan selisih tanggal2 dan tanggal1 dalam satuan bulan	MONTHS_BETWEEN („01-JUL-05", „14-MAR-05") = 3.5806451612903 2
NEXT_DAY()	Mengembalikan tanggal berikutnya	NEXT_DAY() = 07-JUL-05
TO_CHAR (tanggal, format)	Mengubah tanggal menjadi bentuk karakter sesuai dengan format. Sehingga dapat ditampilkan sebagai string	TO_CHAR (sysdate, „DD-MM-YYYY") = 06-07-2006

3.4 JOIN TABEL

JOIN adalah menggabungkan beberapa tabel dangan cara mengakses setiap tabel secara individu berdasarkan kondisi yang diberikan, kemudian hasilnya digabungkan.

Syarat penggunaan JOIN adalah harus terdapat hubungan / keterkaitan diantara tabel – tabel yang dijadikan sumber dari kolom – kolom yang hendak di join dan ditampilkan. Keterkaitan diatara tabel – tabel berupa kolom – kolom yang memiliki nilai dan tipe data yang sama.

Join dimungkinkan dikarenakan oleh model relasional, begitu juga sebaliknya, join dibutuhkan dikarenakan model relational. Join sebetulnya secara umum terbagi dalam 3 jenis: cartesian product, join condition dan outer join.

Cartessian Product

Cartessian product merupakan himpunan dari hasil kombinasi yang memungkinkan dari baris-baris data dari 2 tabel atau lebih. Cartessian product merupakan join tanpa menggunakan join condition. Dengan demikian, sebuah baris pada tabel yang satu akan digabungkan dengan semua baris pada tabel yang lain, seterusnya sehingga jumlah baris hasil query menjadi (n_1* n_2 *...*n_n) dengan n_i adalah jumlah baris pada tabel ke-i. Jumlah baris hasil query yang merupakan hasil perkalian dari jumlah baris dari tabel-tabel inilah yang disebut dengan produk kartesian (*cartesian product*).

Contoh : Misalkan ilustrasi sebagai berikut

Tabel A

A1	**A2**
1	**10**
2	**100**

Tabel B

B1	**B2**
2	**4**
3	**9**

Jika kita melakukan operasi seperti ini:
SELECT * FROM A, B;
maka akan dihasilkan *Cartesian Product.*

Tabel Produk Cartesian

A1	A2	B1	**B2**
1	10	2	**4**
1	10	3	**9**
2	100	2	**4**
2	**100**	**3**	**9**

Untuk menghindari hal seperti ini, paling tidak terdapat sebuah *join condition* pada query yang melakukan join terhadap kedua tabel tersebut dengan catatan *join condition* tersebut valid.

Join Condition

Join condition menspesifikasikan kondisi join dari beberapa tabel.

Seperti telah disinggung sebelumnya, join biasanya selalu melibatkan kolom-kolom yang terdapat pada tabel yang terlibat join yang memiliki kesamaan dan keserupaan maksud representasi dari kolom tersebut.

Equjoin/simple join/inner join adalah join yang menggunakan operator sama dengan (=) pada *join condition*-nya.

Contoh : Misalnya jika kita melakukan operasi SELECT * FROM A,
B WHERE **A.A1 = B.B1**

Maka akan dihasilkan sbb:

A1	A2	B1	**B2**
2	**100**	**2**	**4**

Dari hasil operasi join tersebut dapat terlihat bahwa hanya nilai-nilai dari field A1 pada tabel A dan nilai-nilai field B1 pada tabel B yang bersesuaian yang ditampilkan. Dalam skenario

ini adalah A1=B1=2.

Non-Equjoin adalah sebuah join yang menggunakan *join condition* yang berisi operator selain sama dengan (=) misalnya operator BETWEEN...AND

Self Join adalah query yang menggabungkan sebuah tabel dengan dirinya sendiri. Tabel tersebut muncul dua kali pada klausa from dan masing-masing harus diikuti dengan nama aliasnya. Penggunaan tabel alias ini wajib dilakukan untuk menghindari ambiguous karena semua nama kolom pada tabel pertama ada juga pada tabel kedua.
Contoh:

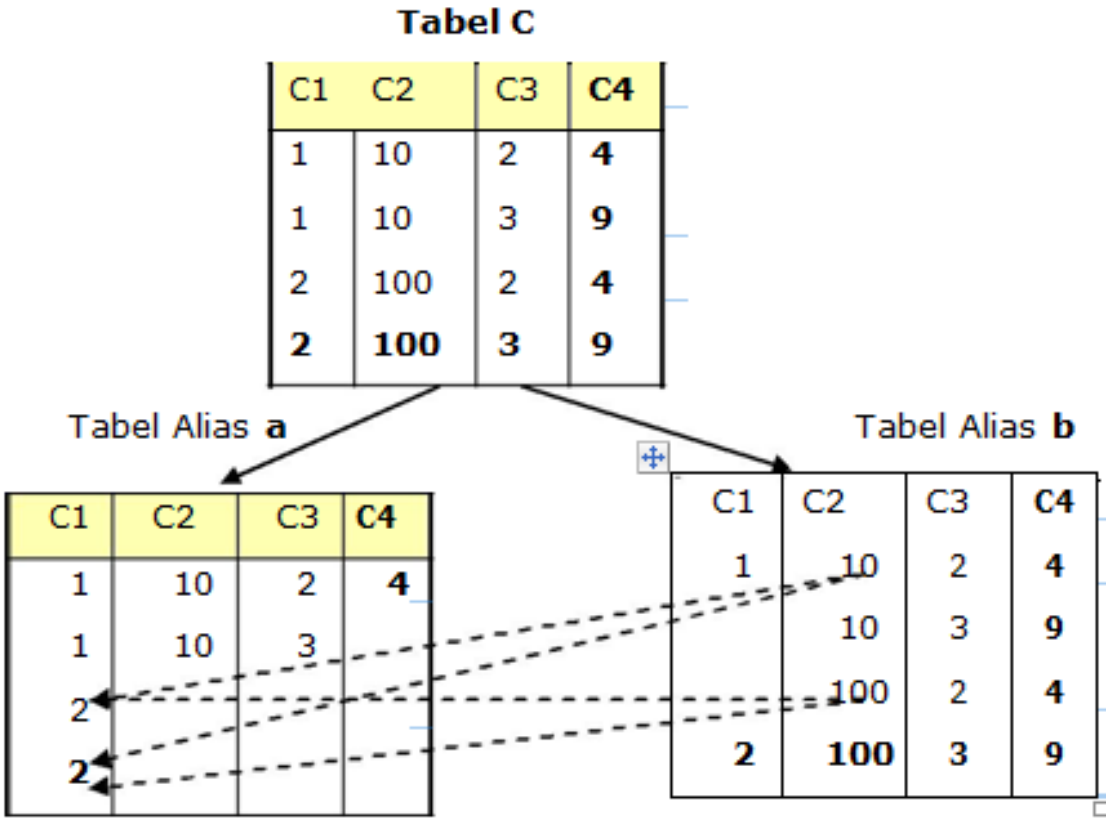
Misalkan terdapat tabel C sebagai berikut

Tabel C

C1	C2	C3	**C4**
1	10	2	**4**
1	10	3	**9**
2	100	2	**4**
2	**100**	**3**	**9**

Jika dilakukan perintah:
SELECT a.C1,a.C2,a.C3,a.C4 FROM C a, C b WHERE a.C1=b.C3

Dari skenario ini dapat dibayangkan kita memiliki dua buah tabel yang identik yaitu **tabel alias a** dan **tabel alias b.** Dimana operasi yang dilakukan dapat digambarkan sebagai berikut:



Maka akan dihasilkan seperti ini

C1	C2	C3	**C4**
2	100	2	**4**
2	100	3	**9**
2	100	2	**4**
2	**100**	**3**	**9**

Outer Join

Outer join biasanya digunakan ketika kita ingin baris data dari tabel yang pertama, atau dari tabel yang kedua atau dari keduanya tetap ditampilkan meskipun tidak terdapat pasangan baris data tersebut dari tabel lain pada kondisi join-nya. Misalkan tabel A di-join dengan tabel B. Semua baris data pada tabel A ingin ditampilkan walaupun tidak memenuhi *join condition.* Untuk itu, operator outer join (+) pada join condition harus disertakan setelah kolom-kolom tabel B. Penulisan operator join dapat ditulis sebagai berikut.

***A.col_name* = *B.col_name*(+)**

Dalam hal ini, tabel B akan memberikan nilai NULL untuk setiap baris pada tabel A jika *join condition* tidak terpenuhi.

Contoh :
Misalnya jika kita melakukan operasi SELECT * FROM A, B WHERE

A.A1 = B.B1(+) maka akan dihasilkan

A1	A2	B1	**B2**
1	10	NULL	**NULL**
2	**100**	**2**	**4**

Pada outer join ada beberapa aturan yang perlu diperhatikan :

Operator "+" hanya boleh ada pada klausa WHERE dan hanya dapat dikenakan pada kolom dari tabel atau view.

Jika terdapat lebih dari satu *join condition*, operator "+" harus dikenakan pada semua *join condition* tersebut. Jika tidak, hasilnya dengan *inner join*.

Jika ada kondisi yang mengandung kolom tabel B dan kolom tersebut dibandingkan dengan sebuah nilai konstan, operator "+" harus dikenakan pada kolom tersebut. Misalkan:

SELECT * FROM A, B WHERE A.A1 = B.B1(+) and **B.B1(+) > 0**;

Kondisi yang mengandung operator "+" tidak boleh dikombinasikan dengan logika **OR** dan **IN**.

Kolom dengan operator "+" tidak boleh dibandingkan dengan hasil subquery. Misalkan:

SELECT * FROM A, B WHERE A.A1 = B.B1(+) and **B.B1(+) > (SELECT C4 FROM C WHERE C4=2)**;

Untuk query yang menggunakan *outer join* dan melibatkan lebih dari dua tabel, misalnya tabel A, B, dan C, maka tabel yang satu hanya dapat men-*generate* tabel NULL untuk satu tabel. Dengan demikian, operator

"+" tidak dapat dikenakan pada kolom tabel B untuk dua buah *join condition*: A join B dan C join B.

Misalkan:
SELECT * FROM A, B WHERE A.A1 = **B.B1(+)** and C.C1=**B.B1(+)** ;

Kesimpulan: Operasi join digunakan untuk menampilkan data dari beberapa tabel. Ada beberapa macam operasi join, antara lain :

Operator Join	Deskripsi	Contoh
Cartesian Product	Menampilkan data dari beberapa tabel tanpa kondisi tertentu.	SELECT NAMA_PEGAWAI, NAMA_DEPARTEMEN FROM PEGAWAI P, DEPARTEMEN D
Join Condition	Menampilkan data dari beberapa tabel dengan kondisi tertentu	SELECT NAMA_PEGAWAI, NAMA_DEPARTEMEN FROM PEGAWAI P, DEPARTEMEN D WHERE P.NO_DEP= D.NO_DEPARTEMEN;
Outer Join	Menampilkan data dari beberapa tabel	SELECT P. NAMA_PEGAWAI, M. NAMA_PEGAWAI AS
	dimana kolom yang diacu tidak memiliki anggota	MANAJER FROM PEGAWAI P, PEGAWAI M WHERE P.NO_MANAJER = M.NO_PEGAWAI(+);

3.5 QUERY BERSARANG

Pada kondisi tertentu, terkadang beberapa query membutuhkan nilai yang dihasilkannya dan digunakan sebagai kondisi pembandingan (dalam klausa

WHERE sebagai contoh). Query tersebut dapat diformulasikan menggunakan nested query (query bersarang). Query bersarang akan melibatkan query dan subquery.

Subquery merupakan statement SELECT yang bersarang didalam klausa WHERE dari statement SELECT yang lain. Meskipun jarang, tetapi subquery juga bisa terdapat pada statement DML yang lain seperti INSERT, UPDATE ataupun DELETE.

Secara umum, sintaks dari NESTED QUERY adalah:

```
SELECT [DISTINCT] select_list
FROM table1, table_2 [,table_3 …]
WHERE {expression
{[NOT] IN | comparison operator} | [NOT] EXIST }
( SELECT [DISTINCT] subquery_select_list
        FROM table_list
        WHERE search_conditions)
```

Yang dilakukan oleh subquery adalah menetapkan kondisi pencarian pada klausa WHERE dalam beberapa cara:
Menghasilkan list untuk klausa IN

Mengembalikan sebuah nilai yang dapat digunakan oleh operator Mengembalikan nilai boolean (true atau false)

Didalam sebuah SELECT statement, subquery boleh berada pada klausa-klausa berikut ini:

- Klausa WHERE
- Klausa HAVING
- Klausa FROM
- Klausa START WITH (pada query hirarki)

Sebagai catatan, subquery pada klausa WHERE sering disebut sebagai *nested subquery*.
Sedangkan subquery pada klausa FROM sering disebut dengan istilah *inline view*.

Banyak permasalahan yang dapat dipecahkan dengan menggunakan subquery.
Daftar berikut ini menunjukkan beberapa kegunaan subquery:

- Memberikan nilai sebagai kondisi di dalam klausa WHERE, HAVING, dan START WITH (subquery pada SELECT, UPDATE, dan INSERT statement).

- Menentukan sekumpulan baris data untuk kebutuhan membuat view (subquery pada CREATE VIEW statement).

- Mendefinisikan sebuah tabel (*inline view*) yang digunakan oleh query utama (subquery pada klausa FROM dari sebuah SELECT statement).

- Menentukan sekumpulan baris data yang dimasukkan ke dalam tabel tujuan, baik ke dalam tabel yang sudah ada (subquery pada INSERT statement) maupun pada saat yang bersamaan dengan pembuatan tabel (subquery pada CREATE TABLE statement).

- Mendapatkan satu nilai atau lebhih yang digunakan untuk mengubah data yang sudah ada (subquery pada UPDATE statement).

Klausa IN

Subquery yang dilakukan menggunakan klausa IN akan dilakukan pengecekan apakah suatu nilai ada atau tidak dalam hasil dari subquery. Contoh :
SELECT A1 FROM A WHERE A1 **IN** (SELECT C3 FROM C)

Hasilnya sbb:

A1	A2
2	100

Klausa ALL

Membandingkan nilai dengan setiap nilai yang dikembalikan oleh query tersarang. Kondisi perbandingan akan bernilai benar jika semua nilai yang dikembalikan oleh subquery memenuhi kondisi tersebut.

Contoh :
SELECT * FROM A WHERE A1 < **ALL**(SELECT C3 FROM C)

Hasilnya sbb:

A1	A2
1	10

Klausa ANY/SOME

Membandingkan nilai dengan salah satu nilai yang dikembalikan oleh query tersarang. Kondisi perbandingan akan bernilai benar jika salah satu nilai yang dikembalikan oleh subquery memenuhi kondisi tersebut dan akan bernilai salah atau tidak terpenuhi jika subquery tidak mengembalikan nilai apapun (tabel kosong).

Contoh :

SELECT * FROM A WHERE A1 < **ANY**(SELECT C3 FROM C) Hasilnya sbb:

A1	A2
1	10
2	100

Klausa [NOT] EXISTS

Klausa EXISTS digunakan untuk melakukan pengecekan apakah hasil dari nested query yang berkorelasi menghasilkan baris data atau tidak. Operator *exists* akan menghasilkan nilai “TRUE” jika subquery yang mengikutinya menghasilkan paling tidak satu baris data.

Contoh :

SELECT * FROM A WHERE **EXISTS** (SELECT C3 FROM C) Hasilnya sbb:

A1	A2
1	10
2	100

3.6 Latihan

1. Buatlah query untuk menampilkan data-data supplier(id supplier, nama) dan jumlah jenis buku yg disupplay oleh supplier ybs!

2. Buatlah query untuk menampilkan data-data buku beserta nama supplier dan jenis bukunya!

3. Jelaskan dan tuliskan output query dibawah ini, jika query tersebut salah maka buatlah query yang benar:

```
select a.id_buku, (select kategori from jenis
where id_jenis=a.id_jenis) as kategori,
(select nama from jenis where
id_supplier=a.id_supplier) as supplier,
a.judul from buku a where stock > 50;
```

4. Jelaskan dan tuliskan output query dibawah ini, jika query tersebut salah maka buatlah query yang benar:

```
select a.id_jenis, a.kategori, sum(b.stock) from jenis a,
buku b where a.id_jenis=id_supplier;
```

4. BACKUP DAN RECOVERY

Issue dari Backup dan Recovery bagi administrator database yaitu:

- mengantisipasi terjadinya suatu failure pada database
- meningkatkan *mean time between failure*
- mengurangi *mean time to recover* dan
- mengurangi terjadinya loss data

Backup berarti suatu kegiatan untuk menyimpan database dalam bentuk *datafile, control file dan archieve log* ke dalam media penyimpanan lain seperti *disk* atau *tape.* Sedangkan Recovery merupakan suatu proses untuk mengupdate database dengan file *Backup* yang telah disimpan terakhir kalinya. Ada dua cara untuk melakukan Backup dan Recovery, yaitu:

1. *Physical Backup*
   Contohnya : RMAN Backup dan Restore
2. *Logical Backup*
   Contohnya : Export dan Import, Flashback

Dalam melakukan backup dan Recovery pada oracle 10g, disediakan tampilan web (Enterprise Manager) sehingga memudahkan admin database untuk melakukan administrasi. Berikut beberapa hal yang perlu dilakukan apabila akan melakukan backup dan Recovery pada Enterprise Manager :

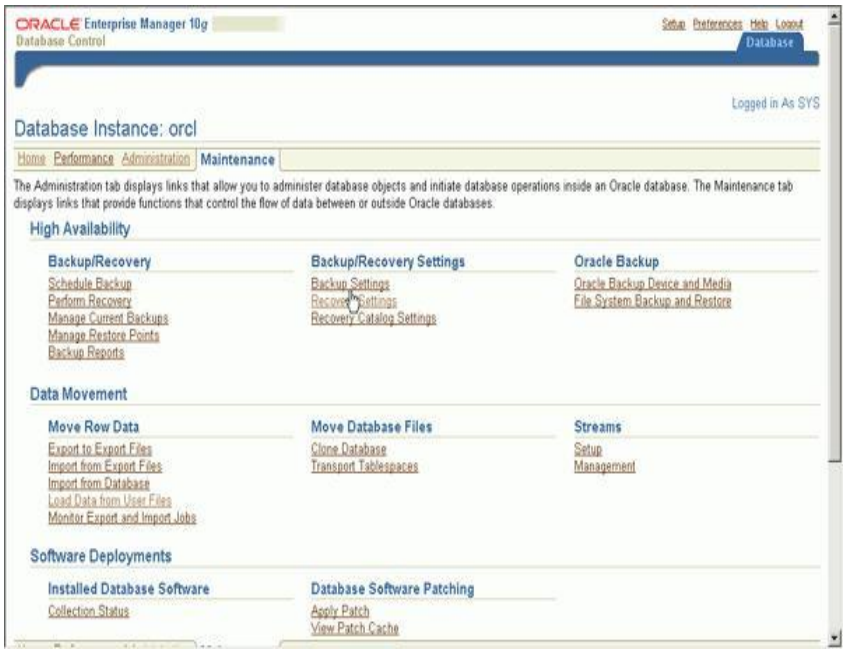
1. Konfigurasi setting backup dan kebijakan (policy)
2. Menentukan DBID dan DB_UNIQUE_NAME
3. Melakukan backup seluruh database
4. Melakukan backup database dengan Strategi Oracle-suggested
5. Melakukan restoring dan recovery database dengan menggunakan strategi oracle-suggested
6. Flashback table
7. Flashback Drop
8. Mengatur backup

4.1 KONFIGURASI SETTING BACKUP DAN KEBIJAKAN

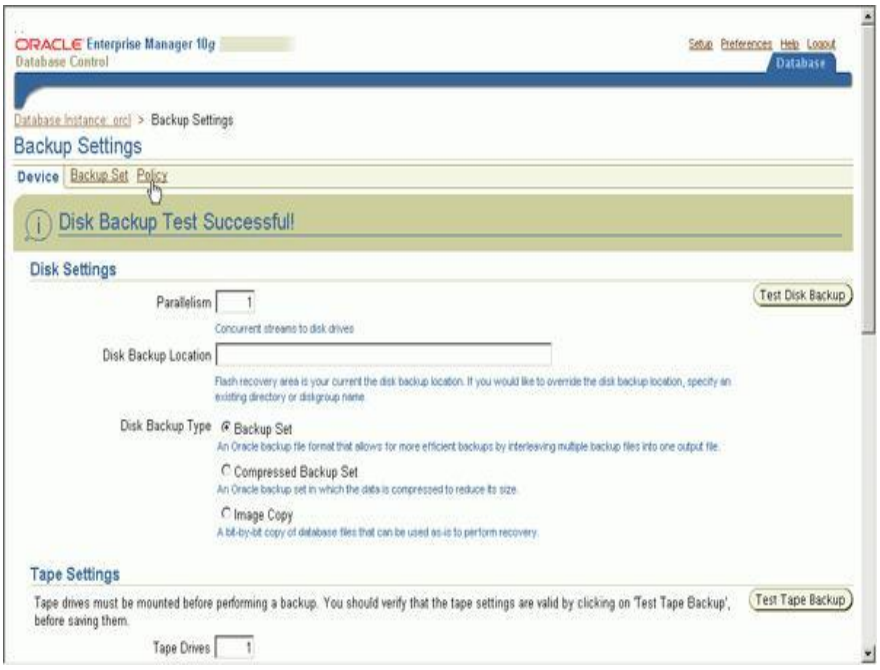
Kita dapat mengkonfigurasi sejumlah setting dan kebijakan (policy) tentang bagaimana cara untuk melakukan backup, data yang akan di backup, bagaimana cara penyimpanan file yang di backup, bagaimana untuk melakukan restore, dan berapa lama backup akan disimpan di

Recovery area. Kita juga bisa mengkonfigurasi feature untuk memperbaiki Perform ansi dari backup.

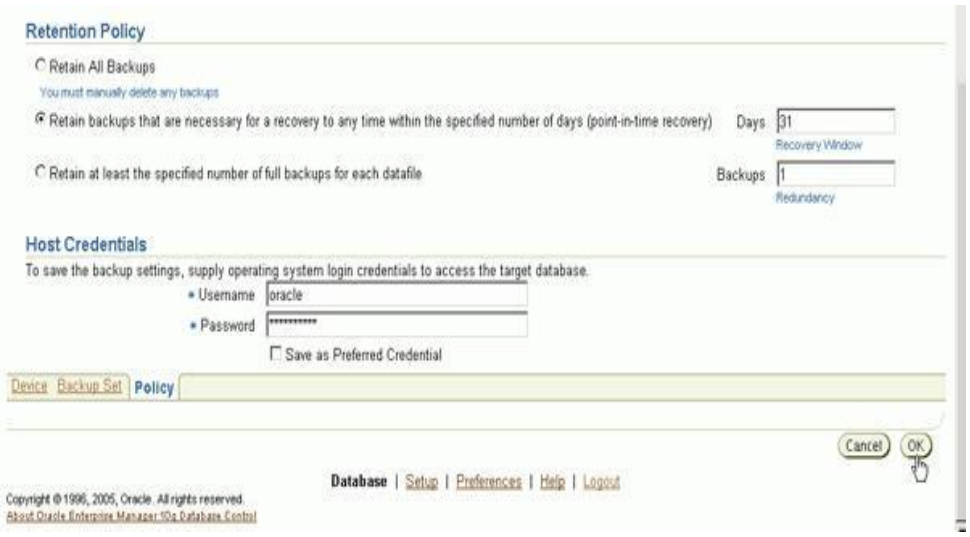
1. Klik Maintenance di halaman home Oracle Database



2. Pilih Backup Setting di bagian Backup/Recovery Settings.
3. Arahkan ke host credential di halaman Backup Setting. Kemudian masukkan username dan password system operasi kita. Kemudian arahkan ke bagian disk setting.
4. Terima nilai 1 di kolom parallelism di bagian disk setting pada halaman device. Kolom lokasi disk Backup diset kosong jadi untuk backup akan digunakan flash Recovery area. Pilih backup set untuk tipe disk backup. Kemudian klik test disk Backup.



5. Akan ada tampilan pesan bahwa test disk setting untuk backup telah sukses.
6. Sekarang kita akan mengkonfigurasi settingan kebijakan untuk backup. Klik policy untuk mengakses halaman policy.
7. Pilih Automatically Backup untuk control file dan server parameter file (spfile) dengan pengubahan struktur database dan backup database.
8. Pilih optimize seluruh database dengan melewatkan file yang sudah di backup.
9. Pilih enable block change tracking for faster incremental backup.



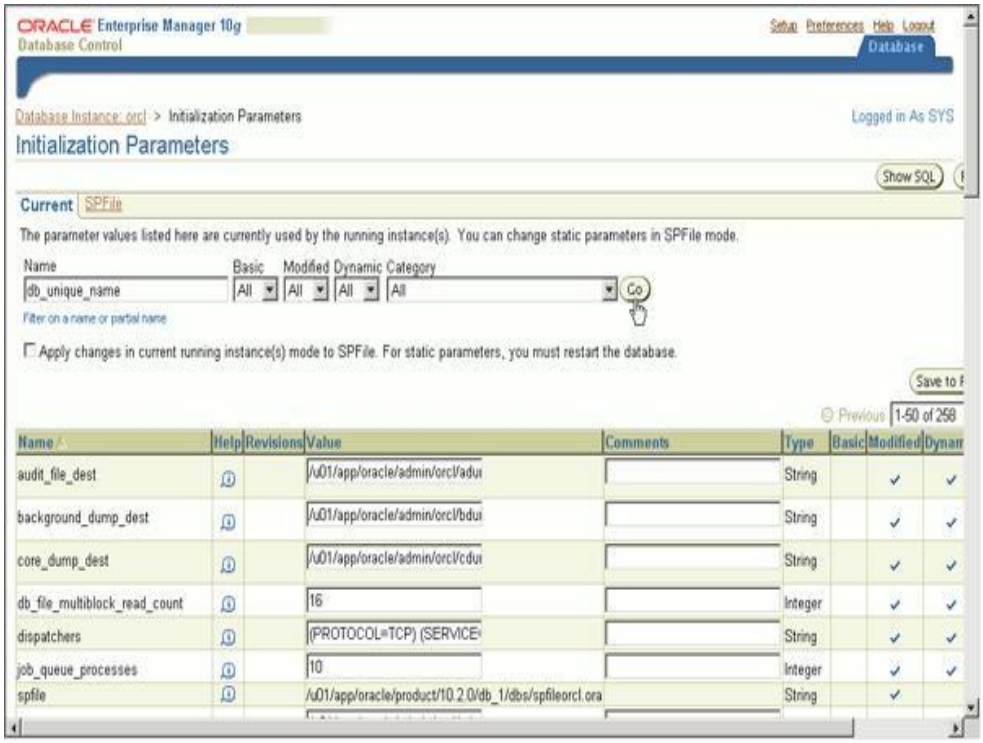
10. Masukkan nama untuk block change tracking file. Kemudian arahkan ke bagian retention policy.
11. Pilih Retain Backup that are necessary for a Recovery to any time within the asecified number of days (point-in-time Recovery) dan terima jumlah harinya yaitu 31 hari.

12. Klik ok. kemudian kita akan kembali ke halaman Maintenance.

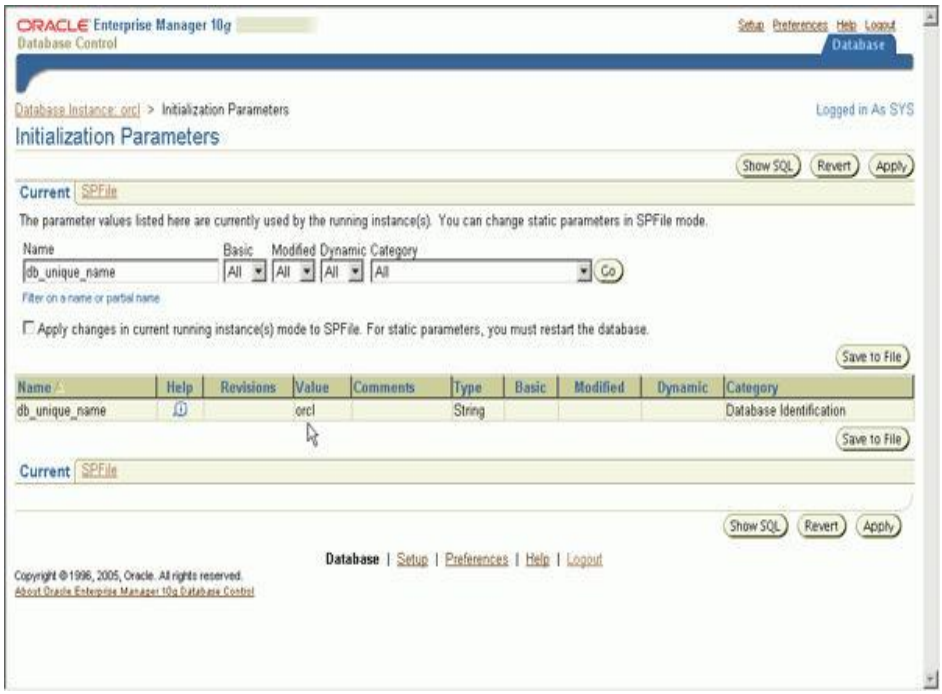
4.2 MENENTUKAN DBID DAN DB_UNIQUE_NAME

Jika kita kehilangan controlfile atau spfile database kita, Enterprise Manager tetap dapat mengembalikan database kita yang telah di backup, selama kita mengetahui **DB_UNIQUE_NAME** dan **DBID** dari database kita. Berikut langkah-langkah untuk mengetahui **DB_UNIQUE_NAME** :

1. Klik All initialization Parameters di bagian Instance di halaman Administration
2. Masukkan „db_unique_name“ di kolom Filter pada halaman property. Current dan kemudian klik button Go



1. Halaman berikutnya akan tampil hasil eksekusi diatas yang menunjukkan db_unique_name di kolom nama dan db_unique_name value yaitu orcl di kolom value



Berikut langkah-langkah untuk menentukan DBID :

1. Klik controlfiles di bagian storage pada halaman administration.
2. Halaman control files muncul. Pilih property advanced.

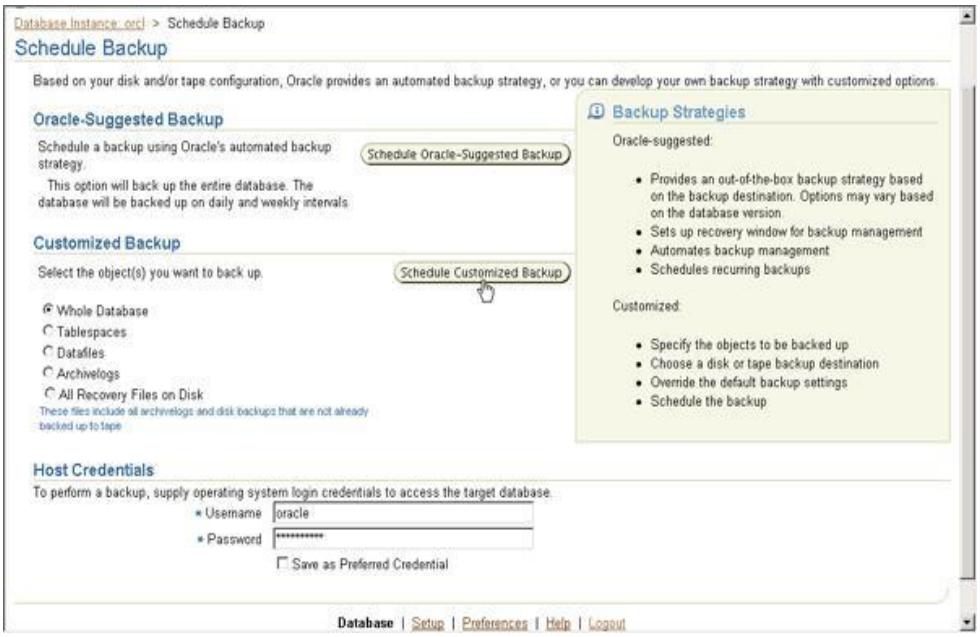
3. Kolom database ID berisi nomor DBID.

Note : DBID harap diingat karena dibutuhkan untuk operasi Recovery.

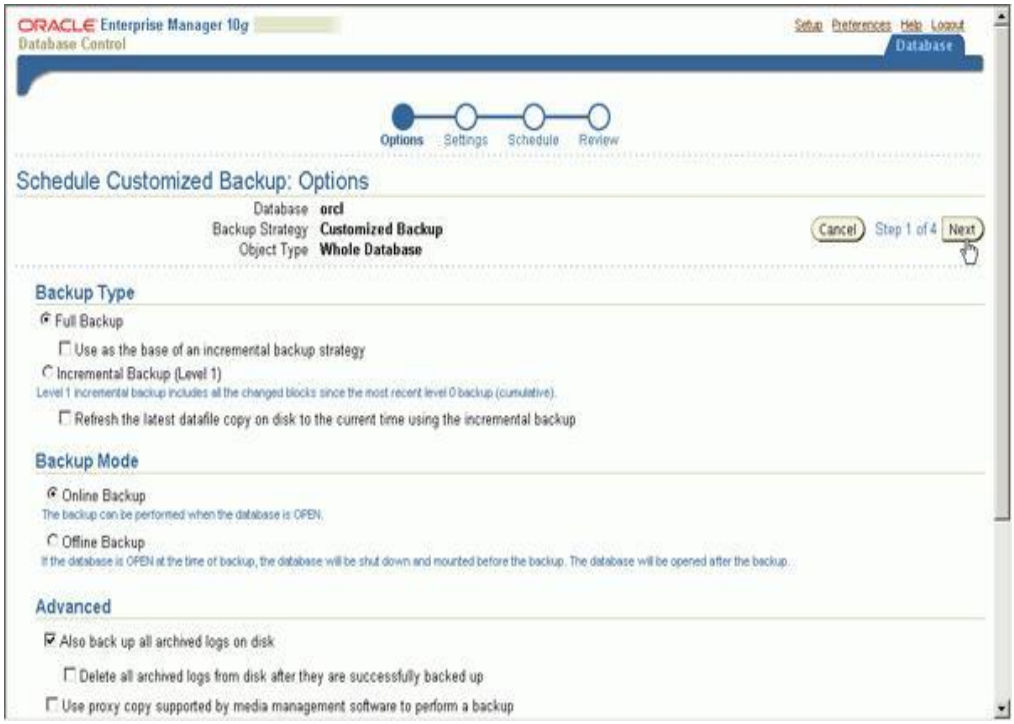
4.3 MELAKUKAN BACKUP SELURUH DATABASE

Kita dapat melakukan backup seluruh isi database kita dengan melakukan backup keseluruhan (backup seluruh datafile). Hasil dari backup seluruh database kita akan disimpan sebagai salinan gambar atau sebagai tumpukan backup, tetapi di contoh yang lain isi dari seluruh database akan direpresentasikan sama dengan control file, archieved redo log dan server parameter file. Database akan dapat di Recovery dengan file-file ini. Berikut langkah-langkah untuk backup seluruh database :

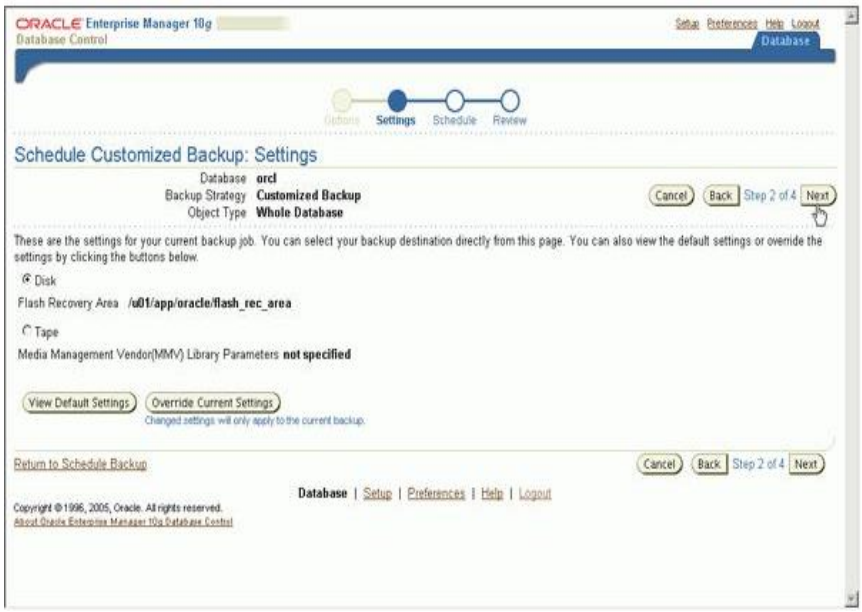
1. Pada halaman Home pilih menu Maintenance**.**
2. Arahkan ke menu Backup/Recovery, dan pilih Schedule Backup.
3. Halaman Schedule Backup muncul. Arahkan cursor ke bagian Customized Backup. Pilih Whole Database dan masukkan nama dan password di host credential. Kemudian klik schedule customized backup.



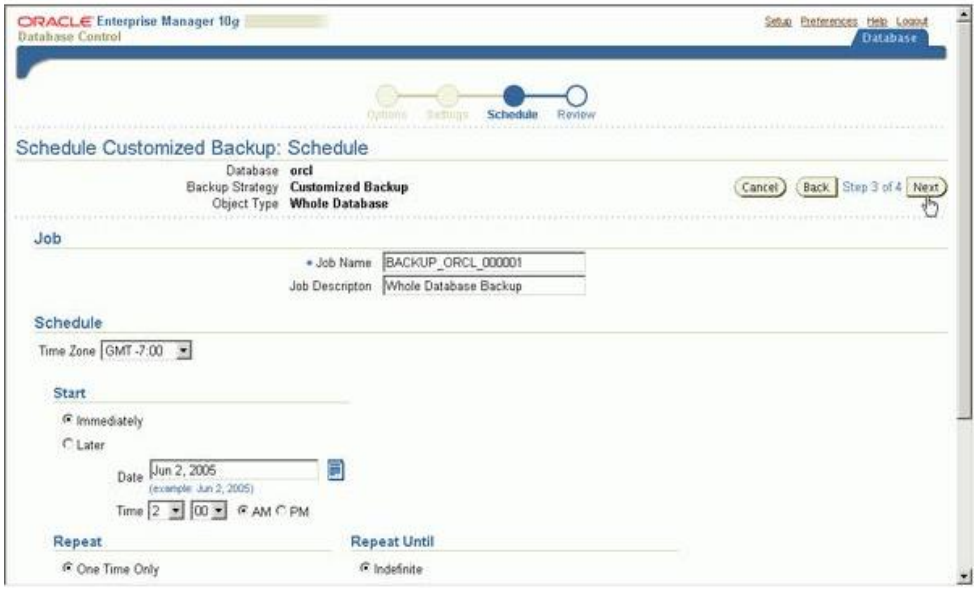
4. Halaman Schedule Customized Backup : Options muncul. Pilih Full Backup di bagian Backup Type. Kemudian pilih Online Backup di bagian Backup Mode. Pilih Back up all archieved logs on disk di bagian Advanced. Klik Next.



5. Halaman Schedule Customized Backup **:** Settings muncul. Pilih Disk atau Tape, terserah kita akan membackup data dimana (pada contoh ini kita menggunakan disk sebagai tempat beackup). Klik Next**.**



6. Halaman Schedule Customized Backup **:** Schedule muncul. Kita dapat mengubah atau membiarkan Job Name dan Job Description. Pilih Immediately untuk mengeksekusi tugas tersebut secepat mungkin atau memasukkan waktu untuk mengeksekusi waktu pada waktu tertentu. Kemudian Klik Next**.**



7. Halaman Schedule Backup **:** Review muncul. Klik Submit Job

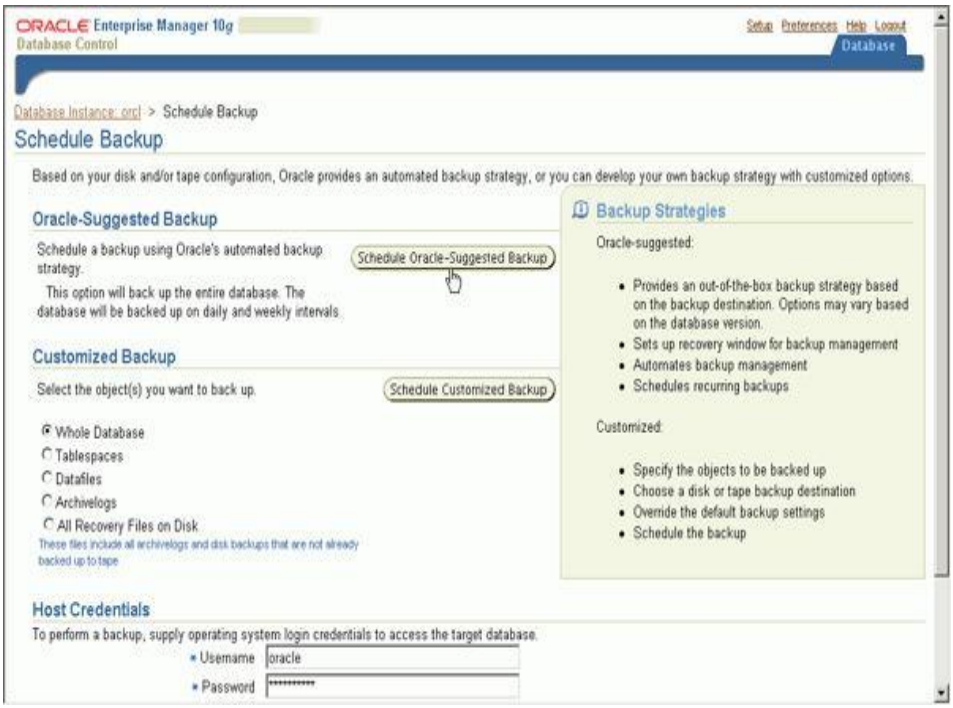
8. Kemudian akan tampil pesan bahwa permintaan untuk Submit telah sukses. Klik OK

Note : kita hanya bisa melakukan backup secara online jika database kita berada pada mode archive

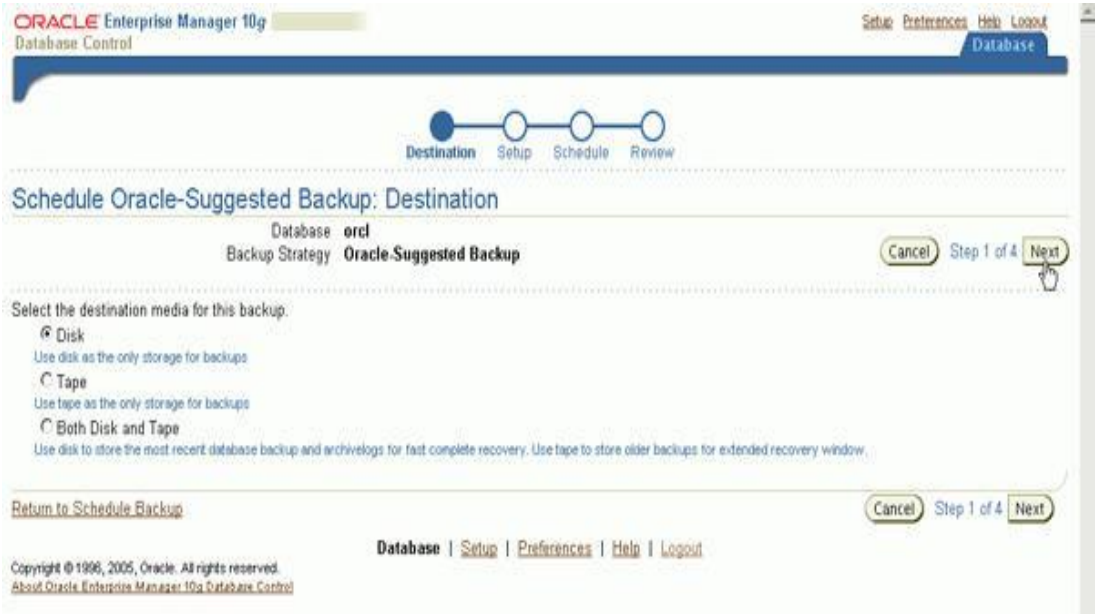
4.4 MELAKUKAN BACKUP DATABASE DENGAN STRATEGI ORACLE-SUGGESTED

Strategi Backup Oracle-suggested ialah backup akan menyalin gambaran database (image copy) dengan menggunakan backup incremental
RMAN. Oracle Enterprise Manager akan menjadwalkan backup dengan RMAN. Berikut langkah-langkah untuk mensetting Schedule Backup :

1. Pilih Schedule Backup di bagian Backup/Recovery
2. Halaman Schedule Backup muncul. Pilih Oracle-suggested backup. Kemudian masukkan username dan password di bagian host credentials. Klik Next**.**

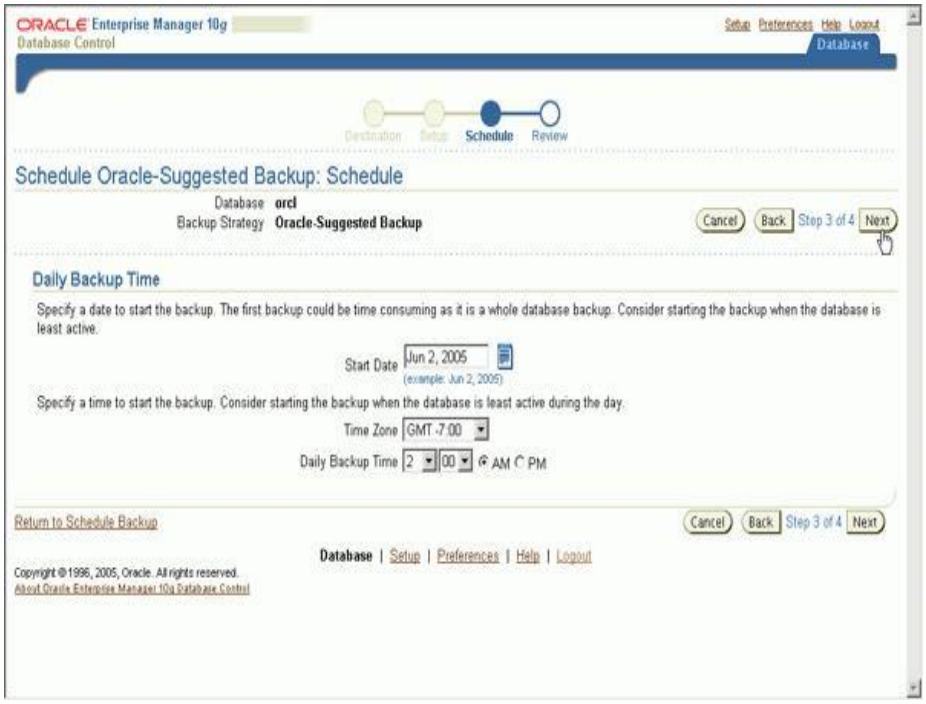


3. Pilih disk sebagai tujuan dari tempat backup yang akan kita coba. Kemudian klik Next.



4. Halaman Schedule Backup : Schedule muncul. Baca kembali informasinya klik Next

5. Halaman Schedule Backup : Schedule muncul. Baca kembali informasinya dan sesuaikan tanggal dan waktu sesuai keperluan kita. Kemudian Klik Next



6. Halaman Schedule Backup : Review muncul. Baca kembali informasinya dan klik Submit Job

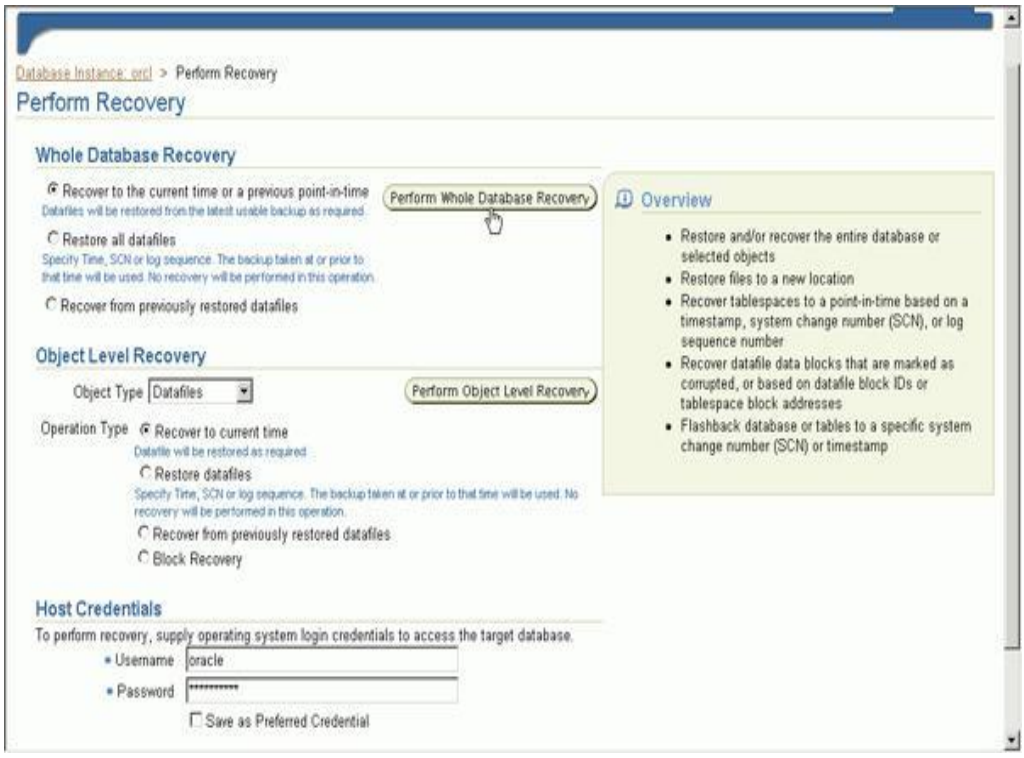
7. Halaman Status akan muncul dengan sebuah pesan yang menunjukkan bahwa job telah didaftarkan dengan sukses. Kita dapat klik View Job untuk mengakses halaman status job atau klik OK untuk melengkapi operasi tersebut.

Note : backup dengan strategi Oracle-suggested dilakukan secara incremental

4.5 MELAKUKAN RECOVERY DATABASE SECARA KESELURUHAN

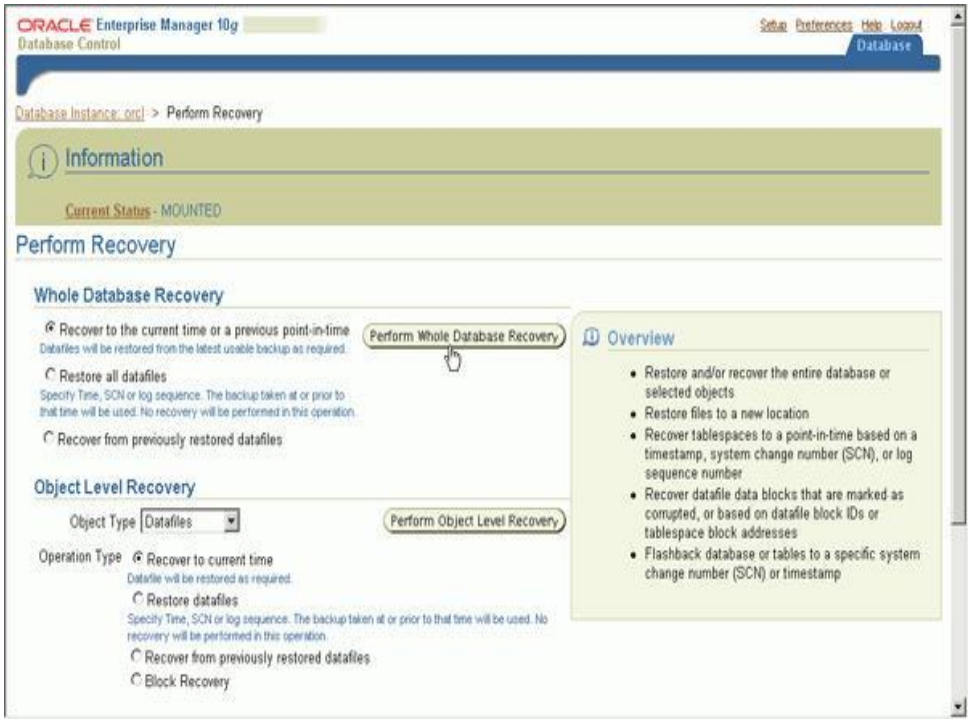
Berikut langkah-langkah untuk melakukan Recovery Database secara keseluruhan :

1. Pilih Maintenance pada menu halaman Home Enterprise Manager.
2. Pilih Perform Recovery di bagian Backup/Recovery.
3. Halaman Perform Recovery muncul. Pilih Recover to the current time or a previous point-in-time sebagai tipe operasi. Masukkan username dan password di bagian Host Credentials. Kemudian klik Perform Whole Database.



4. Halaman Recovery Wizard kemudian muncul dan memberitahu jika instance akan di matikan (shutdown) dan dihidupkan kembali (restarted). Klik Refresh untuk lanjut menggunakan Recovery Wizard

5. Kita kembali ke halaman Maintenance. klik Perform Recovery. masukkan host credentials. klik Perform Whole Database Recovery untuk merecovery database.



6. Halaman Perform Recovery : Point-in-time muncul. Pilih Recover to the current time. Klik Next

7. Halaman Perform Recovery : Rename muncul. Pilih "No. Restore the files to the default location" jika itu pilihan kita atau pilih "Yes. Restore the file to a new, commen location" dan kemudian masukkan lokasinya. Klik Next

8. Halaman Perform Recovery : Review kemudian muncul. Baca kembali informasi dan klik Submit

9. Kemudian akan muncul pesan “Operation succeeded”. Klik Ok.
   Note : proses ini hanya dapat dilakukan jika spfile dan controlfile tidak hilang atau rusak

4.6 FLASHBACK TABEL

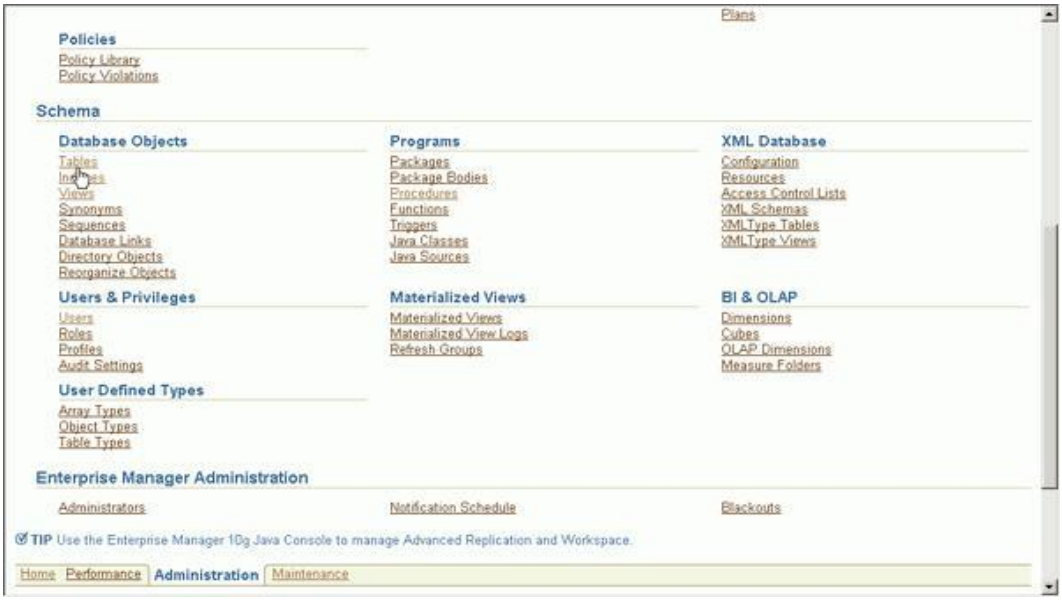
Untuk dapat melakukan operasi flashback pada tabel, maka harus dilakukan langkah-langkah berikut :

- Enable Row Movement
- Simulasikan User Error
- Melakukan Flashback pada tabel
- Flashback Drop

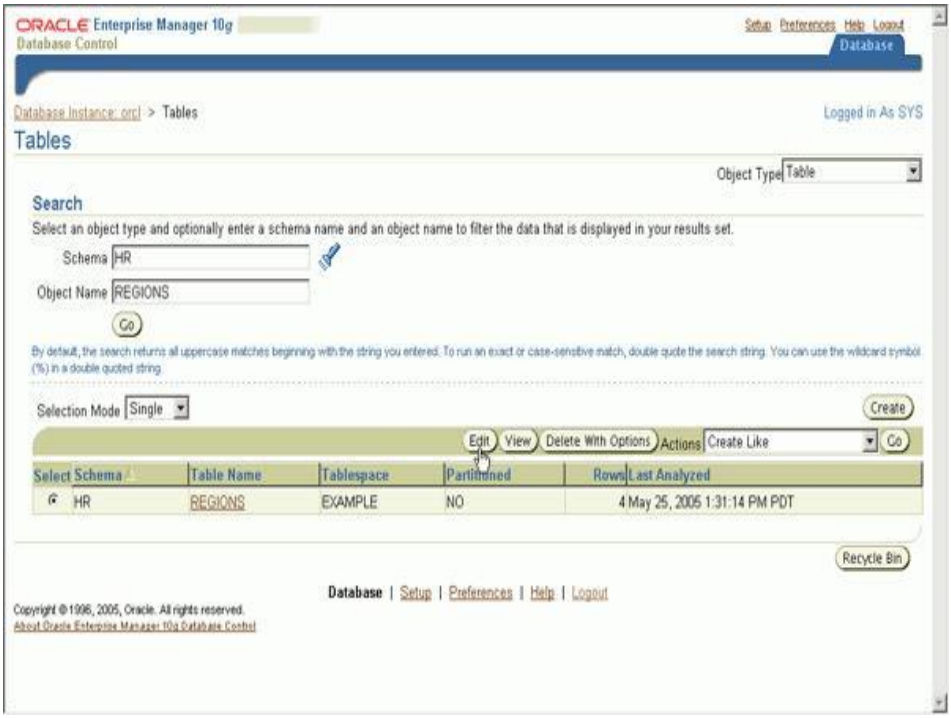
a. Enable Row Movement

Enable Row Movement dilakukan pada tabel-tabel tertentu, misalnya kita memiliki tabel HR.REGIONS.

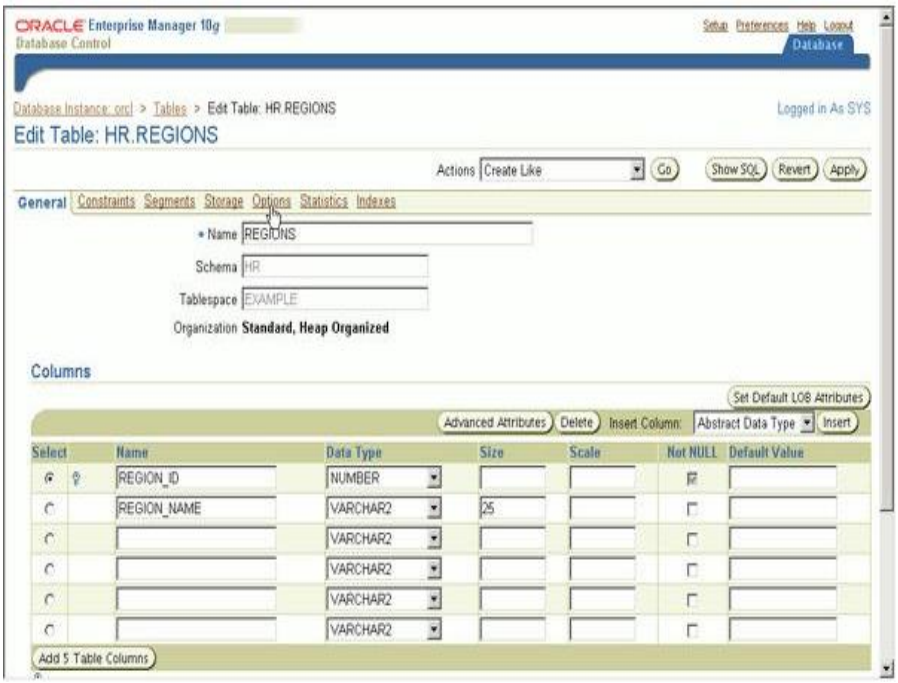
1. Klik Administration di halaman Oracle Database Home.
2. Halaman property Administration kemudian muncul. Pilih Tables di bagian Schema.



3. Halaman Tables kemudian muncul. Masukkan HR di bagian Schema dan Regions di kolom Object Name. klik Go.
4. Tabel REGIONS kemudian terpilih di bagian Results. Klik Edit



5. Halaman Edit Table kemudian muncul. Klik tab Options



6. Pilih Yes dari menu drop-down Enable Row Movement. Klik Apply

7. Kemudian kita akan menerima sebuah pesan yang menyatakan bahwa tabel telah dengan sukses di modifikasi. Kemudian pilih Tables

b. Simulasikan User Error

Sekarang kita akan mensimulasikan error yang disebabkan karena perubahan data di tabel regions.

1. Lihat data di tabel regions dengan membuka sql plus dan mengeksekusi perintah berikut

   Sqlplus HR/HR
   col region_name format a30
   *select * from regions;*

2. Simulasikan user error dengan mengeksekusi perintah sql untuk mengganti nilai dari kolom region_name di semua baris :

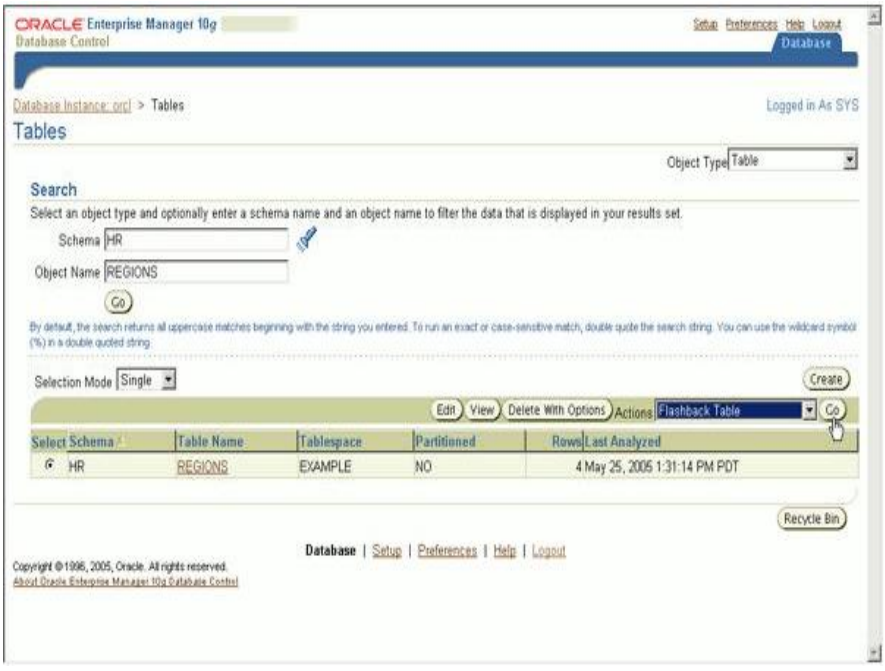
   Update regions set region_name = "Oracle";
   Commit;

3. Lihat perubahannya dengan menuliskan perintah
   *Select * from regions;*

c. Melakukan Flashback pada tabel

Berikut langkah-langkah untuk melakukan flashback pada tabel :

1. Pastikan bahwa REGIONS masih terpilih di daftar tabel. Pilih Flashback Table dari menu drop-down. Klik Go



2. Halaman Perform Recovery : Point-in-time kemudian muncul. Pilih Flashback to a timestamp dan masukkan tanggal dan waktu beberapa menit yang lalu. Klik Next

3. Halaman Perform Recovery : Flashback Tables muncul. Baca kembali informasi di halaman tersebut kemudian klik Next

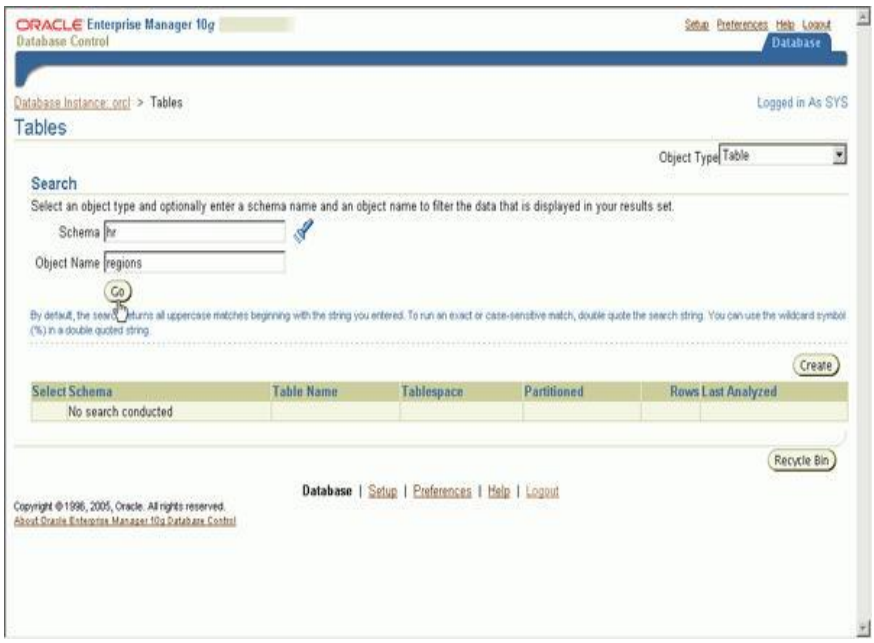
4. Halaman Perform Recovery : Review muncul. Baca kembali informasnya dan kemudian klik Submit.

5. Kita akan menerima konfirmasi bahwa tabel telah di flashback. Klik Ok.
6. Sekarang pindah ke sesi sql*plus dan eksekusi perintah untuk melihat operasi flashback tabel yang telah kita lakukan : *Select * from regions*

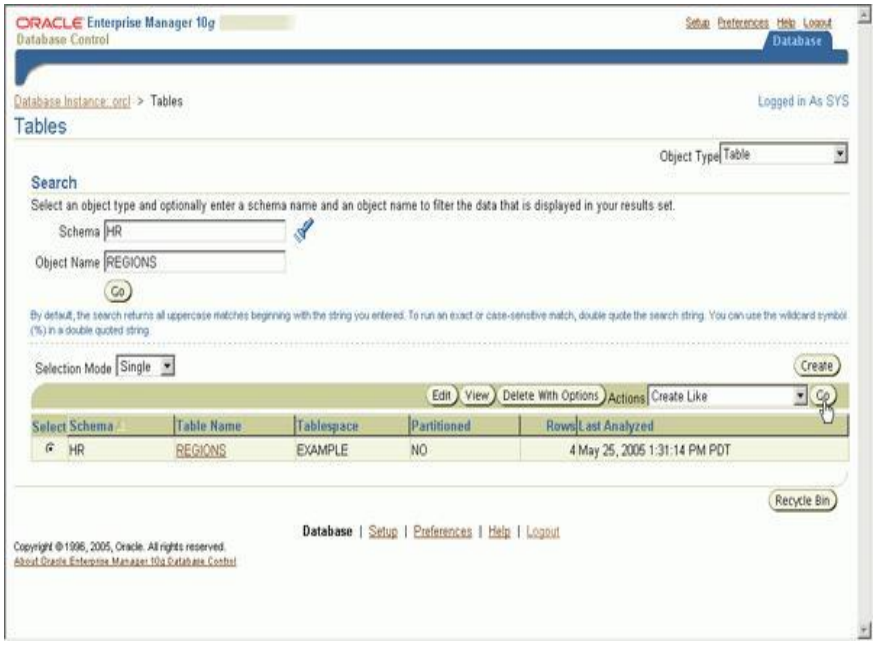
d. Flashback Drop

Pada bagian ini kita akan menggunakan fitur flashback drop untuk mengembalikan sebuah tabel yang telah di drop. Sebagai latihan, akita akan membuat sebuah tabel, menghapus tabel, kemudian mengembalikannya dengna menggunakan flashback drop. Berikut langkah-langkah untuk membuat tabel baru dan kemudian kita hapus tabel tersebut:

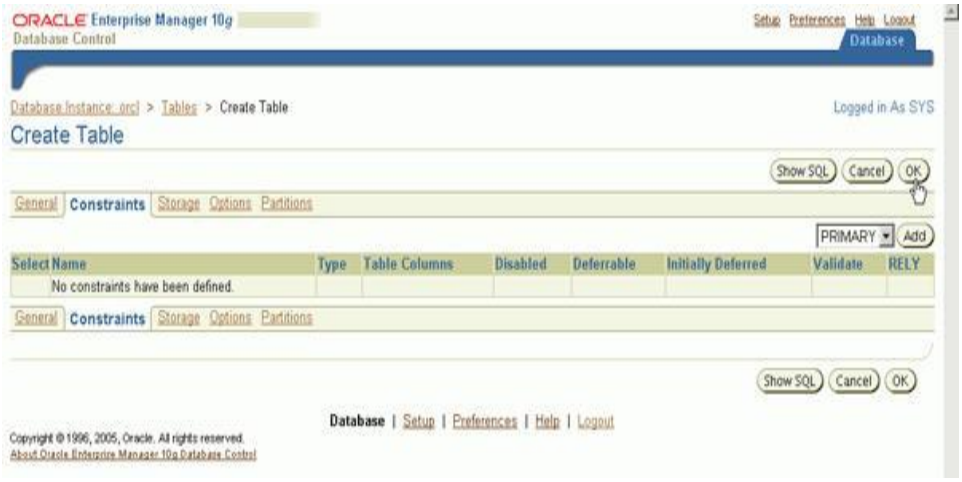
1. Masukkan HR di kolom Schema dan REGIONS di kolom Object Name atau sebagian dari nama di kolom Object Name dan klik Go



2. Pilih Create Like dari menu drop down actions. Klik Go



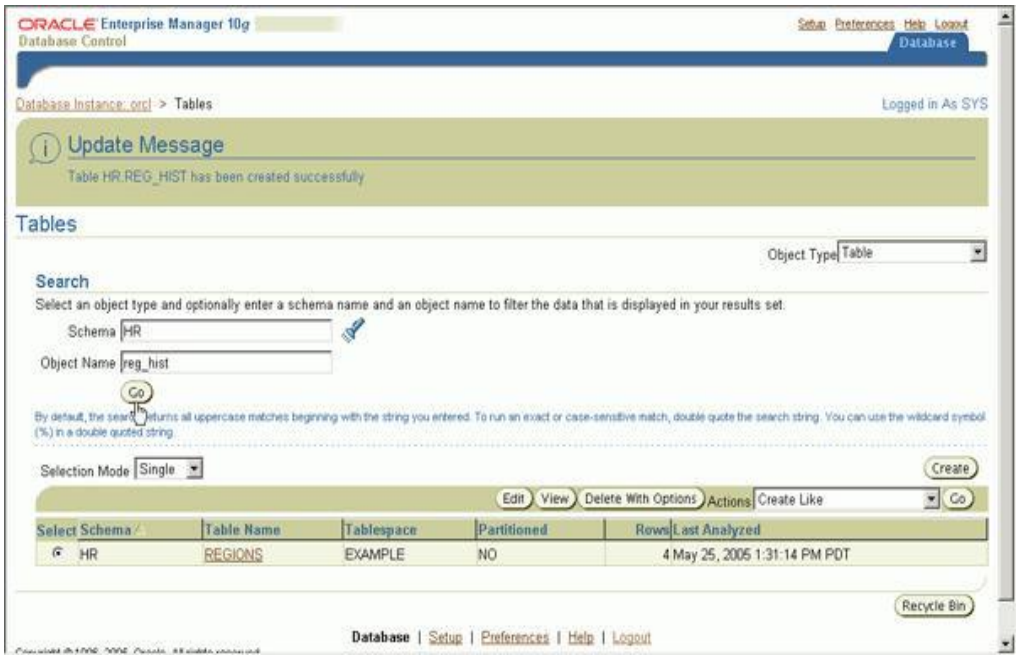
3. Halaman untuk membuat tabel kemudian muncul. Masukkan REG_HIST di kolom nama. Jangan klik Not Null pada kolom REGION_ID. Klik Constraints



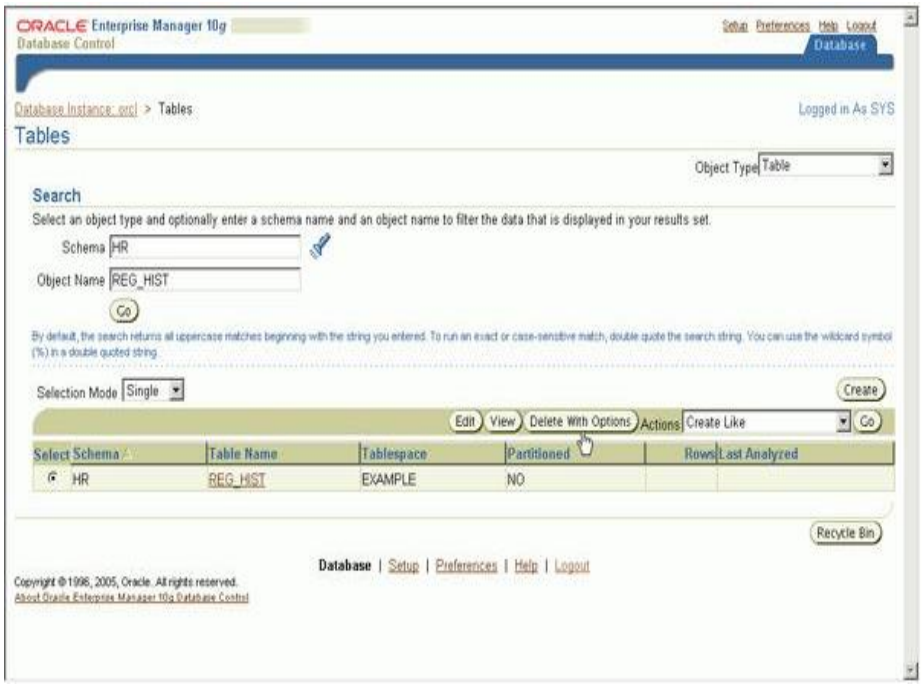
4. Halaman Constraints kemudian muncul. Hapus constraint pada tabel dengan memilihnya satu-satu kemudian klik delete

5. Klik OK untuk membuat tabel REG_HIST

6. Kita akan menerima pesan yang menyatakan bahwa tabel telah dibuat. Kemudian masukkan REG_HIST ke dalam kolom Object Name dan klik Go



7. Halaman Tables kemudian ditampilkan dengan tabel REG_HIST di bagian Results. Klik Delete untuk menghapus tabel REG_HIST



8. Klik Yes untuk mengkonfirmasi penghapusan tabel

9. Kemudian dikeluarkan pesan yang menandakan bahwa proses penghapusan tabel telah dieksekusi. Klik Go untuk mencari tabel.

10. Tidak ada yang akan ditampilkan di bagian Result Displayed in the results section.

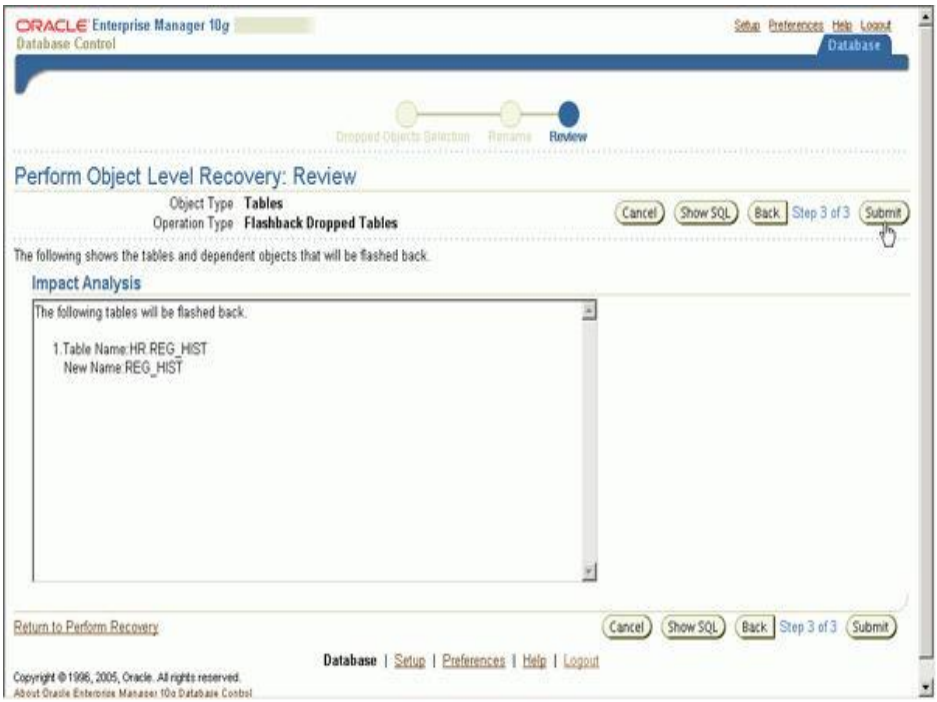
Untuk mengembalikan tabel yang telah terhapus, kita dapat melakukan flashback drop. Berikut langkah-langkahnya :

1. Pilih Recycle Bin
2. Inputkan HR di kolom nama Schema dan klik Go.

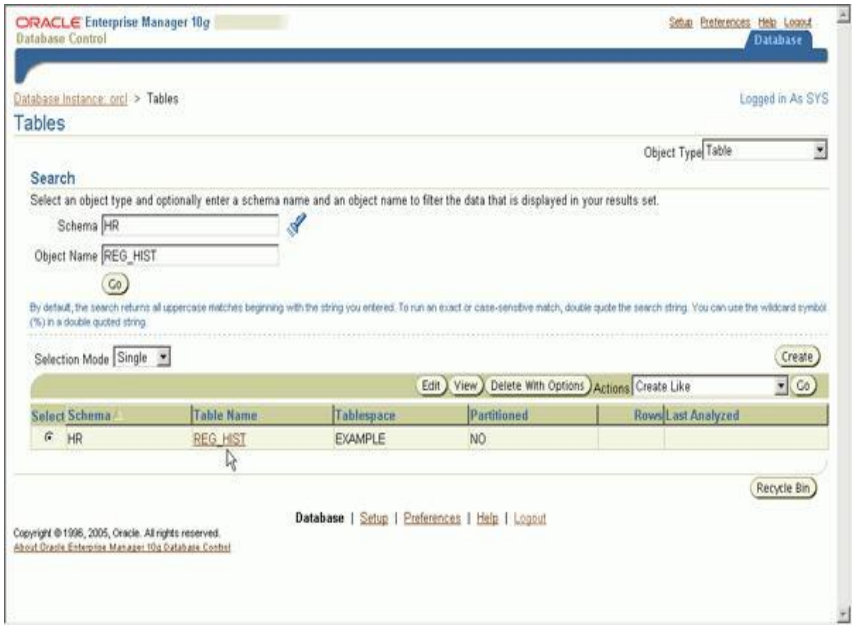
3. Pilih tabel REG_HIST kemudian klik Flashback Drop

4. Halaman Perform Recovery : Rename kemudian muncul. Klik Next

5. Halaman Perform Recovery : Rview kemudian muncul. Baca kembali informasi tersebut kemudian klik Submit.



6. Kemudian akan muncul pesan konfirmasi. Klik Ok
7. Tabel tersebut tidak lagi berada di Recycle Bin. Klik Tables.
8. Sekarang tabel REG_HIST telah ada di daftar tabel.



4.7 MENGATUR BACKUP

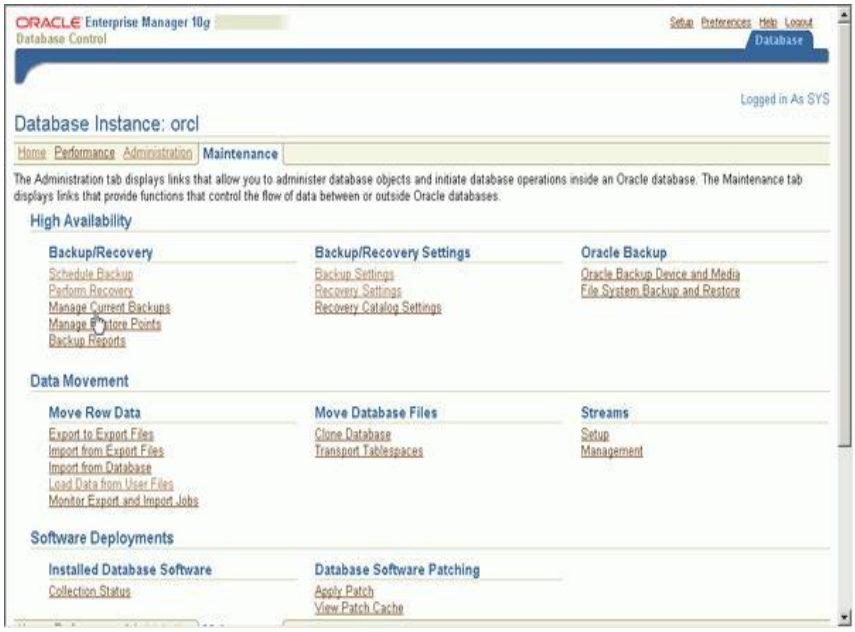
Mengatur backup terdiri dari 2 tugas yaitu mengatur backup file yang ada di disk atau tape, dan mengatur catatan dari backup yang ada di

RMAN Repository. Pada bagian ini, kita akan melakukan maintenance pada backup dan meng-update RMAN repository. Pada bagian ini kita akan melakukan hal-hal berikut :

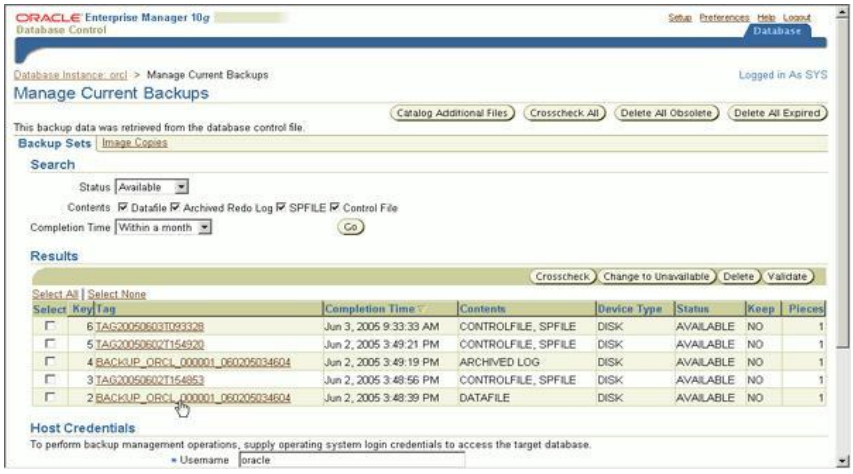
- Menggunakan halaman untuk mem backup database yang ada pada waktu ini
- Backup crosschecking
- Menghapus backup yang sudah expired
- Menghapus backup yang Obsolete
- Menandakan backup menjadi tidak ada (unavailable)
- Membuat catalog untuk backup

a. Menggunakan halaman untuk mem backup database yang ada pada waktu ini.

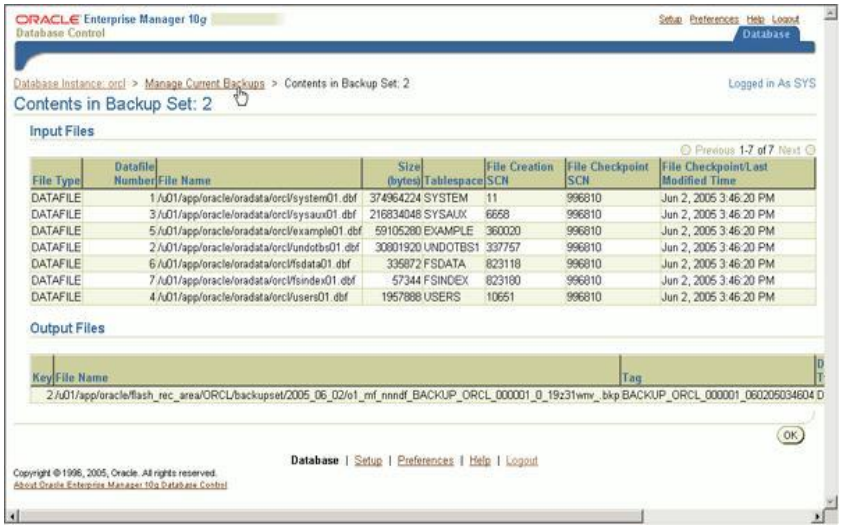
1. Klik Manage Current Backups pada bagian Maintenance > Backup/Recovery



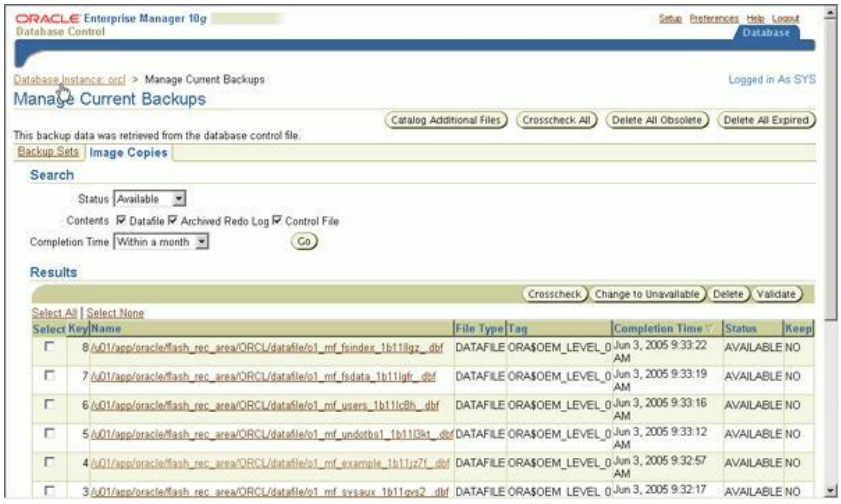
2. Halaman Manage Current Backups muncul. Halaman Backup Sets akan menunjukkan backup yang ada di RMAN repository. Klik link di kolom. Contents untuk melihat detail informasi mengenai isi dari backup



3. Halaman Contents kemudian muncul. Klik Manage Currents Backups untuk kembali.



4. Klik Image Copies untuk melihat halaman Image Copies

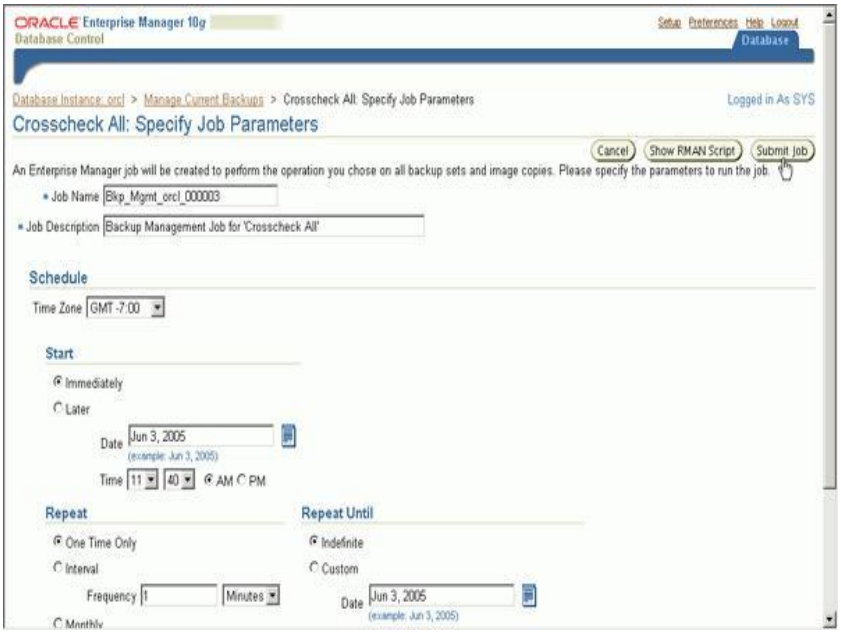


5. Halaman Image Copies kemudian memperlihatkan jika image copies yang telah disimpan di RMAN repository

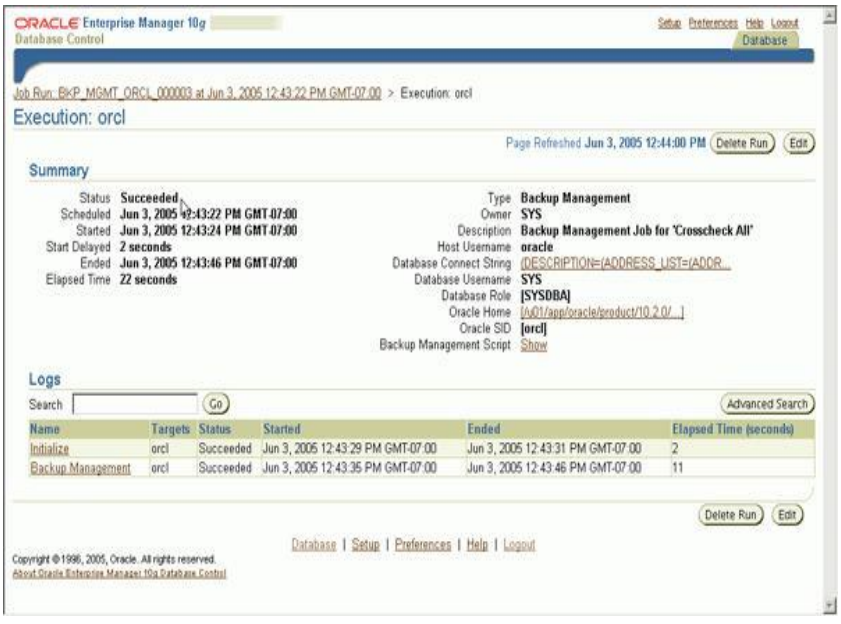
b. Backup crosschecking

Saat kita melakukan crosschecking sebuah backup, RMAN akan memverifikasi bahwa informasi yang disimpan di repository sesuai dengan yang ada di backup secara fisik. Jika tidak sesuai, repository akan di update sesuai dengan status yang ada sekarang. Kita dapat melakukan crosscheck seluruh database kita dengan langkah-langkah berikut :

1. Klik Manage Backups di bagian Backup/Recovery
2. Halaman Manage Current Backups kemudian muncul. Klik Crosscheck All dibagian kanan atas untuk melakukan crosscheck semua file yang ada di RMAN repository.
3. Halaman Crosscheck All : Specify Job Parameters muncul. Kita dapat menerima nama job standard untuk nama job, deskripsi job, waktu mulai, dan ulangi spesifikasinya atau masukkan nilai yang kita inginkan. Klik Submit Job untuk memasukkan job untuk crosscheck.

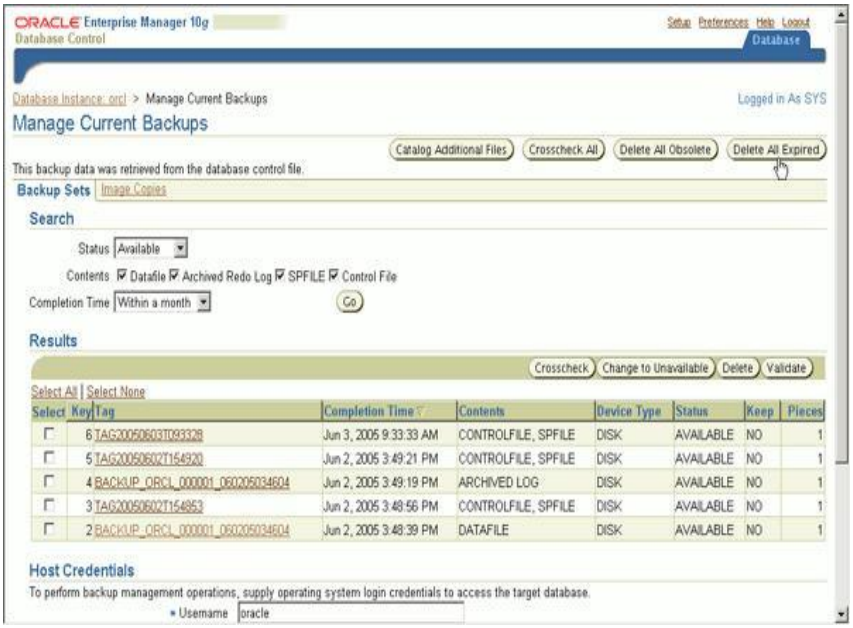


4. Sebuah pesan bahwa permintaan job telah sukses kemudian ditampilkan. Kemudian klik View Job untuk melihat status dari job tersebut.
5. Di bagian Summary, kita dapat melihat statu dari job

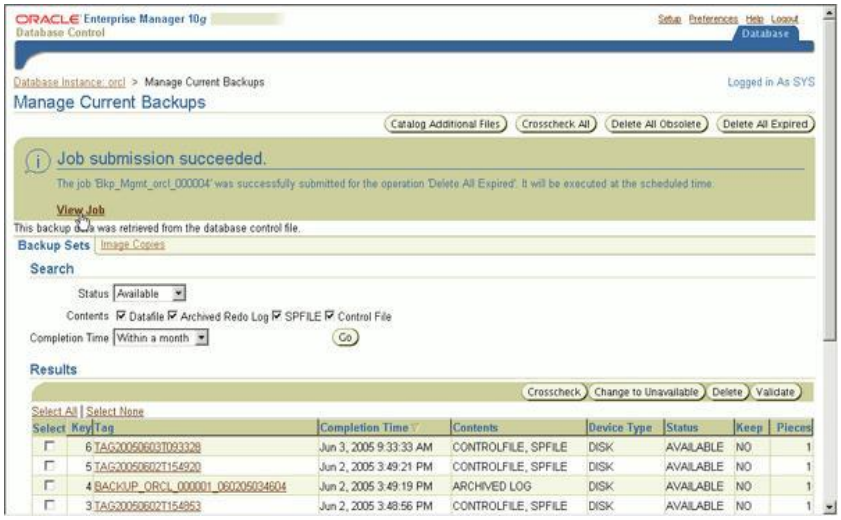


c. Menghapus backup yang sudah expired

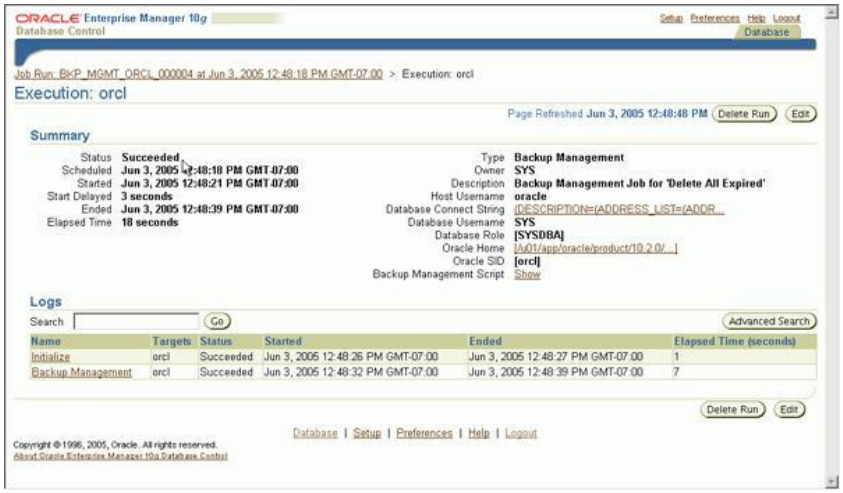
1. Klik Manage Current Backups pada bagian Backup/Recovery di halaman Maintenance.
2. Halaman Manage Current Backups muncul. Klik Delete All Expired di atas halaman untuk menghapus RMAN repository semua backup yang terdeteksi sebagai backup yang using.



3. Halaman Delete All Expired: Specify Job Parameters kemudian muncul.
   Kita bisa membiarkan standar untuk nama Job, deskripsi job, waktu mulai, dan spesifikasi yang berulang atau memasukkan nilai tertentu. Pilih Perform the operation „ Crosscheck All ’ before „Delete All Expired‟. Jika kita tidak melakukan operasi crosscheck. Klik Submit Job untuk memasukkan job tersebut.
4. Kemudian pesan yang mengatakan jika Job Submission sukses ditampilkan. Kita dapat memilih View Job untuk melihat status dari Job



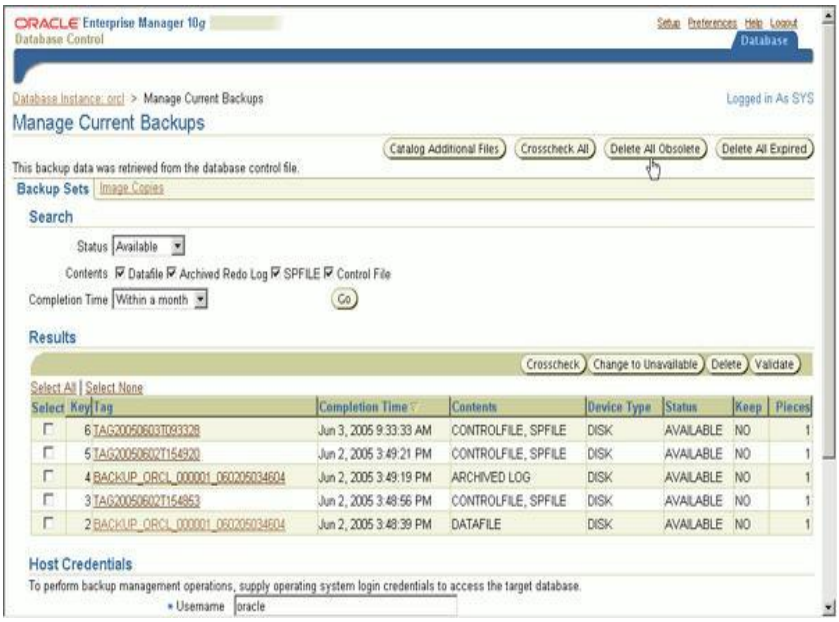
5. Pada bagian Summary, kita dapat melihat status dari job



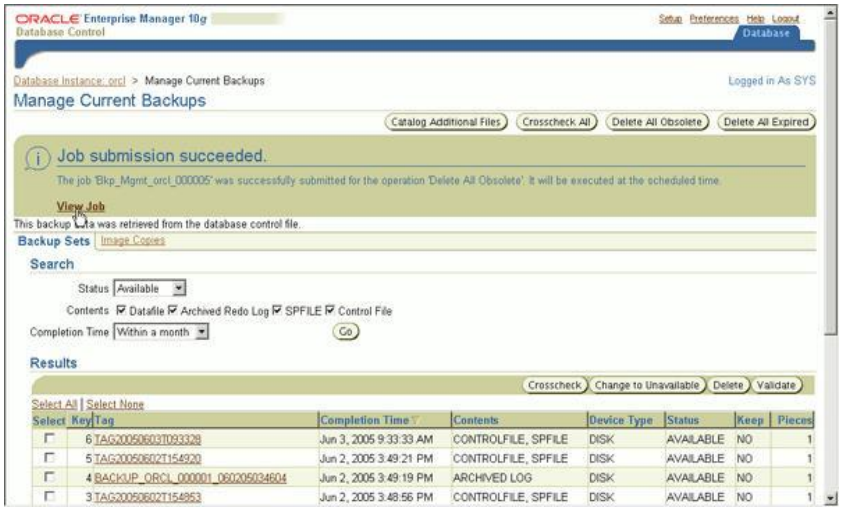
d. Menghapus backup yang Obsolete (Usang)

Backup yang using (obsolete) berarti file Backup sudah tidak layak untuk digunakan dalam melakukan Recovery. Untuk dapat menghapus Backup yang obsolete berikut langkah-langkahnya :

1. Klik Manage Current Backups di bagian Backup/Recovery pada halaman Maintenance.
2. Halaman Manage Current Backups muncul. Klik Delete All Obsolete di bagian kanan atas halaman untuk membuang backup yang tidak terpakai dari RMAN repository yang tidak terpakai.



3. Halaman Delete All Obsolete: Specify Job parameters kemudian muncul. Kita dapat menerima standar untuk nama Job, deskripsi Job, waktu mulai dan spesifikasi yang diulang-ulang atau masukkan nilainya sendiri. Klik Submit Job untuk mendaftar job tersebut
4. Kemudian akan muncul pesan konfirmasi bahwa pendaftaran job telah sukses didaftarkan dan kemudian ditampilkan pada halaman Manage Current Backups. Kita dapat memilih View job untuk melihat status dari job.



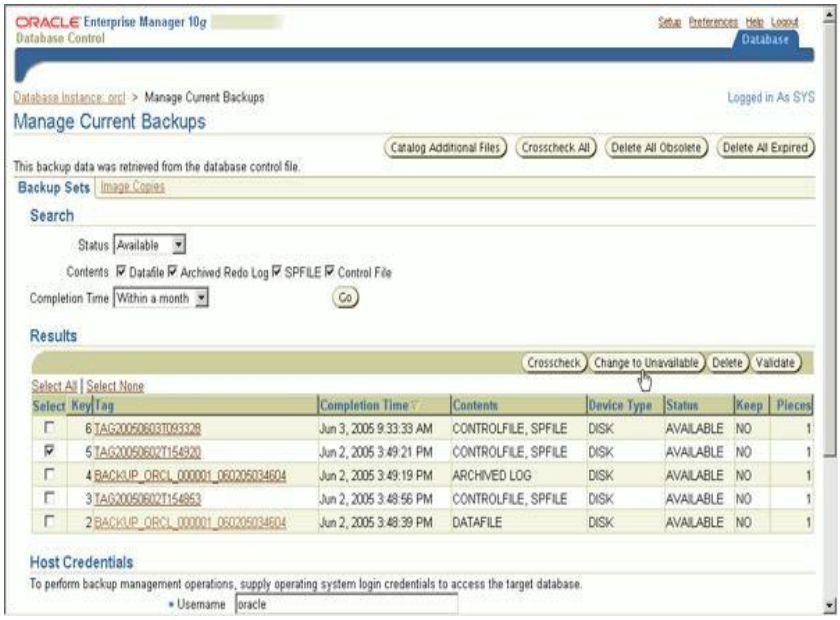
5. Di bagian Summary kita dapat melihat status job.

6. Kembali ke halaman **Image Copies** atau **Backup Sets** untuk memastikan bahwa **Backup** yang usang telah dihapus

e. Menandakan backup menjadi tidak ada (unavailable)

Untuk dapat menandakan backup menjadi tidak ada (unavailable) maka dapat dilakukan langkah-langkah berikut :

1. Klik Manage Current Backups di bagian Backup/Recovery pada halaman Maintenance.
2. Halaman Manage Current Backups muncul. Pilih Backup yang akan kita tandai sebagai UNAVAILABLE dan klik Change to Unavailable.



3. Sebuah halaman akan ditampilkan. Klik Yes untuk melanjutkan operasi.
4. Halaman permintaan sedang dalam keadaan progress ditampilkan.

f. Membuat catalog untuk backup

Kita dapat mengklasifikasikan backup dengan perintah system operasi sehingga RMAN dapat menggunakan mereka dalam sebuah operasi Recovery. Sebagai contoh, disini kita akan melakukan backup datafile yang termasuk ke dalam contoh table space dengan menggunakan perintah system operasi. Kemudian kita bisa mengklarifikasikan file backup di RMAN repository dengan menggunakan Enterprise Manager. Berikut langkah-langkahnya :

1. Buka SQL*Plus dan login sebagi user dengan hak akses sebagai SYSDBA. Disini kita diberikan pilihan untuk melakukan backup online pada sebuah tablespace. Missal dilakukan backup pada tablespace example.

```
[oracle@EDRSR10P1 oracle]$ sqlplus /nolog

SQL*Plus: Release 10.2.0.0.0 - Beta on Fri Jun 3 13:53:54 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

SQL> connect / as sysdba
Connected.
SQL> alter tablespace example begin backup;

Tablespace altered.
```

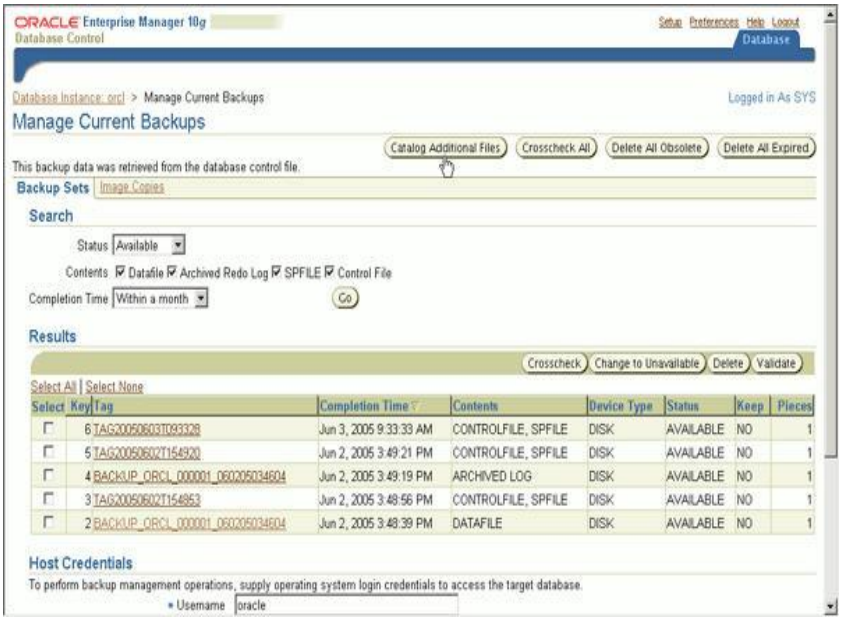
2. Kembali ke promt system operasi dan buatlah sebuah salinan system dan membuat salinan dari datafile yang termasuk ke dalam directory baru yaitu backup. Kita dapat menggunakan directory lain terserah keinginan.

```
SQL> host
[oracle@EDRSR10P1 oracle]$ cd $ORACLE_BASE
[oracle@EDRSR10P1 oracle]$ cp oradata/orcl/example01.dbf $HOME/backup
[oracle@EDRSR10P1 oracle]$
```

3. Ambil contoh table space yang akan melakukan online backup dengan mengunakan perintah di bawah ini:

```
[oracle@EDRSR10P1 oracle]$ exit
exit

SQL> alter tablespace example end backup;

Tablespace altered.
```

4. Sekarang kita lakukan catalog backup di RMAN repository dengan menggunakan Database Control Enterprise Manager. Klik Manage Current Backups di bagian Backup/Recovery pada halaman Maintenance.
5. Pilih Catalog Additional Files pada bagian kanan atas halaman.



6. Pilih "Catalog files in the specified disk location into the Recovery Manager repository" dan masukkan lokasi dan nama dari backup file. Klik Ok.

7. Halaman progress untuk permintaan ditampilkan

8. Sebuah pesan konfirmasi menyatakan jika file telah dikatalog-kan.

4.8 LATIHAN

1. Buatlah sebuah database dengan nama sesuai nim anda, kemudian bikin tabel dan diisi sesuai studi kasus pada bab sebelumnya . lakukan strategi fullbackup terhadap database anda.

2. Dari database yang telah anda bikin, lakukan operasi flashback tabel.

3. Recover seluruh database anda mengunakan strategi full recovery.
4. Cari cara bagaimana melakukan backup dan recovery menggunakan perintah RMAN (console based)?

5. PENGAMANAN BASIS DATA

5.1 USER MANAGEMENT

a. User sebagai Tipe

Secara umum operasi-operasi untuk pemakai adalah tiga operasi klasik yaitu CREATE, ALTER, dan DROP. Tiga operasi klasik ini berlaku bagi objek-objek hampir semua tipe dalam database (bahkan ke objek bertipe database itu sendiri).

b. User versus Schema

Literatur oracle sering sekali mamakai kata „schema", dan kata ini dipakai secara kurang hati-hati. Oracle memiliki operasi CREATE SCHEMA, namun tidak ada operasi DROP SCHEMA dan ALTER SCHEMA.

Schema didefinisikan sebagai kumpulan objek-objek yang dimiliki pemakai/user. Kumpulan objek yang dimiliki pemakai SYSTEM membentuk schema SYSTEM. Kumpulan objek yang dimiliki SCOTT membentuk schema

SCOTT. Anda dapat melihat bahwa kadang ada literature atau piranti yang menyebutkan CREATE SCHEMA namun tidak membuat objek sama sekali. Dalam hal tersebut, lebih tepat jika piranti meyatakan CREATE USER.

Pemakain schema lebih tepat diterapkan pada topic data warehouse, saat membicarakan star schema. Selain itu dengan adanya schema semakin memperbanyak tree dalam beberapa piranti (termasuk oracle enterprise manager).

c. User versus account

Ada suatu istilah lagi yang terkait dengan user yaitu account. Istilah ini kelihatannya tidak begitu tepat dalam konteks pemakai database. Saat kita menyatakan LOCK atau UNLOCK suatu account pada dasarnya kita melakukan LOCK USER atau UNLOCK USER.

d. CREATE USER

Hanya dengan hak DBA saja yang bisa melakukan CREATE USER.

Pertama anda connect sebagai pemakai dengan hak DBA untuk melakukan langkah-langkah berikut ini. Kita mulai dengan sintaks sederhana operasi CREATE USER

```
CREATE USER user_name
IDENTIFIED EXTERNALLY | {by password}
[ DEFAULT TABLESPACE tablespace_name ]
[ TEMPORARY TABLESPACE tablespace_name ]
[ ACCOUNT {LOCK|UNLOCK}];
```

Contoh :

```
CREATE USER scott2
IDENTIFIED BY tiger
DEFAULT  TABLESPACE mahasiswa
TEMPORARY TABLESPACE temp_ mahasiswa;
```

Pilihan diatas adalah pembuatan pemakai yang diautentifikasi oleh password. Misalkan kita ingin membuat user scott3 yang diidentifikasi secara external oleh OS. Sintaks yang lengkap dapat dilihat dibuku manual ORACLE.

```
CREATE USER scott3
IDENTIFIED EXTERNALLY
DEFAULT  TABLESPACE praktikum
TEMPORARY TABLESPACE temp_praktikum;
```

Sangat sulit mengecek effektifitas dan konsekuensi autentikasi lewat cara ini.

e. ALTER USER

Mengubah pemakai dilakukan dengan operasi ALTER USER. Perubahan yang perlu biasanya mencakup perubahan password, default tablespace, temporary tablespace, profile, quota, dan status penguncian. Contoh berikut menyangkut pengubahan pemakai, memaksa pemakai untuk mengubah passwordnya.

```
ALTER USER scott2
IDENTIFIED BY peterpan
PASSWORD EXPIRE;
```

Pemanggilan operasi berikut merubah quota pemakai scott2 sehingga tidak bisa membuat objek di tablespace praktikum. Perhatikan bahwa tidak ada penyebutan tipe TABLESPACE .

```
ALTER USER scott2
QUOTA 0 MON MAHASISWA;
```

f. DROP USER

Membuang pemakai dilakukan dengan operasi DROP USER. Sintaksnya :

```
DROP USER user_name [CASCADE];
```

Aturannya sebagai berikut :

Jika pemakai tidak memiliki objek maka penyebutan CASCADE tidak perlu dilakukan. Sekalipun CASCADE dispesifikasikan, pembuangan pemakai bisa gagal yang disebabkan oleh :

i. Mencoba membuang pemakai/user SYS dan SYSTEM. Hal ini disebakan karena kedua user ini tidak bisa dibuang.
ii. Mencoba membuang user yang masih connect ke service.

Perhatikan bahwa tidak ada operasi untuk hanya membuang obyek-obyek seorang pemakai tanpa membuang pemakai tersebut. DBA harus menulis program khusus untuk hal itu.

g. MONITORING USER

Berikut metadata views untuk memantau pemakai ;

i. DBA_users : berisi informasi seluruh user yang dimiliki oleh DBA
ii. DBA_TS_Quotas : berisi informasi tentang tablespace quota bagi setiap pemakai.
iii. V$session : untuk melihat user yang sedang connect ke system

Coba lakukan perintah selectin berikut :

```
SELECT * FROM DBA_users;
```

Perintah diatas digunakan untuk menampilkan semua informasi users yang dimiliki DBA

```
SELECT * FROM DBA_TS_Quotas;
```

Perintah diatas digunakan untuk menampilkan semua informasi quota tablespace masing-masing user

```
SELECT * FROM V$session;
```

Perintah diatas digunakan untuk menampilkan semua user yang connect ke system

5.2 PRIVILAGES

Salah satu model keamanan dalam penggunaan DBMS kita adalah adanya privilege(hak). Model keamanan ini biasa diapakai di OS(operating system) maupun server lainnya.

a. Operasi-operasi

Operasi-operasi untuk privilege terbatas pada GRANT dan REVOKE. Operasi GRANT memberikan hak akses kepada satu atau lebih pemakai, sebaliknya operasi REVOKE membuang hak akses dari satu atau lebih pemakai.

Apa saja hak-hak yang diperoleh seorang pemakai/user apabila DBA memberi grant role connect dan resource, kita bisa memberikan jawabannya dengan melihat metadata view berikut.

1. connectlah sebagai DBA
2. jalankan perintah

```
DESCRIBE DBA_sys_privs;
SQL> set line 10;
SQL> desc dba_sys_privs; Name                 Null?        Type
 ----------------       --------       ------------
 GRANTEE                          NOT NULL     VARCHAR2(30)
  PRIVILEGE                       NOT NULL     VARCHAR2(40)
  ADMIN_OPTION                                 VARCHAR2(3)
```

```
SQL> SELECT * FROM DBA_SYS_PRIVS
WHERE grantee in ('CONNECT','RESOURCE')
ORDER BY grantee;

GRANTEE                        PRIVILEGE                 ADM
------------------------------ ----------------------------------------- ---
CONNECT                        ALTERSESSION              NO
CONNECT                        CREATE CLUSTER            NO
CONNECT                        CREATE DATABASE LINK      NO
CONNECT                        CREATE SEQUENCE           NO
CONNECT                        CREATE SESSION            NO
CONNECT                        CREATE SYNONYM            NO
CONNECT                        CREATE TABLE              NO
CONNECT                        CREATE VIEW               NO
RESOURCE                       CREATE CLUSTER            NO
RESOURCE                       CREATE INDEXTYPE          NO
RESOURCE                       CREATE OPERATOR           NO
RESOURCE                       CREATE PROCEDURE          NO
RESOURCE                       CREATE SEQUENCE           NO
RESOURCE                       CREATE TABLE              NO
RESOURCE                       CREATE TRIGGER            NO
RESOURCE                       CREATE TYPE               NO
```

Scott berhak merubah session, membuat cluster, database link, type, sequence, session, synonym,

view, cluster, indextype, operator, procedure, sequence, table, dan trigger.
Privilage terbagi menjadi dua jenis yaitu system-level dan object-level. Kita akan pelajari keduanya. Sebelum itu mari kita buat seorang user lagi.

```
CREATE USER scotti
IDENTIFIED BY tiger
DEFAULT TABLESPACE mahasiswa
TEMPORARY TABLESPACE temp_ mahasiswa;

GRANT conncet, resource to scotti;
```

System-level privileges

System level privilege tidak terkait dengan object tertentu secara spesifik.
Semua (15) privilege yang kita lihat diatas adalah system level, tak terkait dengan suatu object tertntu secara spesifik. Ada lebih dari 80 system level privilege dalam oracle RDBMS. DBA berhak untuk men-drop user dan backup tables.

Object-level privileges

Privilege ini terkait dengan object tertentu secara spesifik. Misal user bernama scott9 memiliki dua object (bertipe sama maupun bukan). Jika scott9 hanya memberi object level privilege kepada scotti, maka scotti tak otomatis memiliki object level privilege terhadap object yang satu lagi. Mari kita lihat contoh dibawah

```
SQL> connect scott9
Enter password: *****
Connected.

SQL> create table satu(
    a number,
    b varchar(9));

Table created.

SQL> insert into satu values(1,'x');
1 row created.

SQL> create table dua(
    c number,
    d varchar(9));

Table created.

SQL> insert into dua values(1,'x');
1 row created.

SQL> GRANT SELECT ON satu TO scotti;

Grant succeeded.
SQL> connect scotti@basdat/tiger;
```

```
Connected.

SQL> select * from scott9.satu; A B
--------- ---------
        1 x
        2 y
SQL> select * from scott9.dua; select * from scott9.dua
                      *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Scotti dapat mengakses table satu milik scott9 tetapi tidak bisa mengakses table dua milik scott9, karena scotti tidak diberi akses oleh scott9 untuk mengakses table dua.

b. REVOKE PRIVILEGES

Sekarang kita akan mencoba mnghapus hak akses scotti terhadap table satu. Ikuti langkah berikut :

1. masuk sebagai user scott9
2. jalankan perintah berikut

```
SQL> revoke select on satu from scotti;
Revoke succeeded.
```

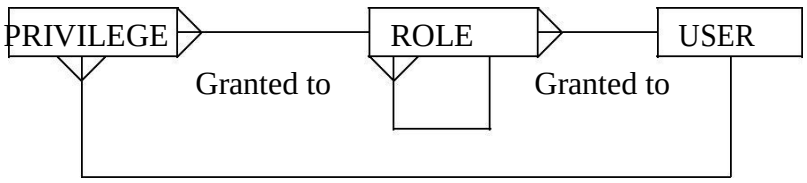
3. masuk sebagai user scotti
4. jalankan selection terhadap table satu milik scott9

```
SQL> select * from scott9.satu; select * from scott9.satu
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Terlihat bahwa revoke telah menghilangkan hak akses terhadap suatu object.
Revoke hanya dapat dilakukan oleh pemberi hak akses.

c. Keamanan Privilege dan Roles

Sekarang kita akan melihat model keamanan yang melibatkan privilege dan role. Untuk memudahkan pemahaman mari kita lihat gambar berikut



Keterangan :
satu atau lebih privilege bisa di grant ke role satu atau lebih role bisa

digrant ke role satu atau lebih role bisa di grant ke user
satu atau lebih privilege bisa di grant ke user.

5.3 ROLES

Role adalah sekumpulan named set of privilege. Operasi-operasi yang terdapat pada role adalah CREATE, ALTER, dan DROP, satu operasi SET untuk pengaktifan ditambah dengan dua operasi GRANT dan REVOKE (seperti untuk privilege). Role dibuat untuk mempermudah pengelolaan privilege.

a. System Defined Role

DBMS Oracle menyediakan system defined role yang siap dipakai

Role	Kegunaan
CONNECT	Lihat di bab sebelumnya
RESOURCE	Lihat di bab sebelumnya
DBA	Semua privilege dengan ADMIN OPTION
EXP_FULL_DATABASE	Export full database
IMP_FULL_DATABASE	Import full database

b. Creating Roles

Membuat role dialakukan dengan operasi CREATE ROLE. Ada syarat agar seorang pemakai dapat menggunakan hak ini(membuat role), yaitu harus memiliki hak akses untuk membuat role(CREATE ROLE). Secara default hak ini tidak diberikan ke pemakai jadi DBA harus memberikan hak ini secara explicit.

1.masukklan sebagai user anda
2.jalankan perintah berikut

```
SQL> CREATE ROLE mahasiswa;
Role created.
```

c. Operasi GRANT

Operasi GRANT dipakai untuk meng-assign privilege atau role kepada role. Pertama kita akan memberikan sebuah privilege kepada role mhs_role berupa kemampuan untuk selection kepada table mahasiswa.

```
SQL> grant select on mahasiswa to mhs_role;
Grant succeeded
```

Kita akan memberikan role yang telah kita buat kepada pemakai kita yaitu scotti.
Jalankan perintah SQL berikut

```
SQL> GRANT mhs_role to scotti; Grant succeeded.
```

d. Memakai Roles

Untuk menggunakan role mhs_role tadi oleh scotti maka scotti harus melakaukan SET terhadap role tersebut

```
SQL> set role mhs_role; Role set.

SQL> select * from scott9.mhs;
NIM          NAMA
-----------  --------------------
30108001  paijo
30108002  paimen
```

e. Removing Roles

Operasi yang digunakan adalah REVOKE

1. masukklah sebagai scott9
2. jalankan SQL berikut

SQL> REVOKE mahasiswa from scotti;

Revoke succeeded.

3. masuklah sebagai scotti jalankan perintah berikut

```
SQL> connect scotti@basdat/tiger

   Connected

SQL> select * from scott9.mhs;
select * from scott9.mhs
         *
ERROR at line 1:
ORA-00942: table or view does not exist
```

f. Mengubah Roles

Operasi ALTER ROLE diapakai untuk mengubah tingkat kerahasiaan.

Sintaks :

```
ALTER ROLE name {NOT IDENTIFIED | IDENTIFIED {BY
password | EXTERNALLY };
```

ALTER ROLE mhs_role IDENTIFIED EXTERNALLY;

g. Membuang Roles

Sama seperti membuang table, index, type dan yang lainnya, yaitu menggunakan operasi DROP.

```
SQL> connect scott9@basdat/tiger

Connected.

SQL> drop role mahasiswa;

Role dropped.
```

Perhatikan efeknya, hak hak yang diperoleh dalam role tersebut akan hilang

```
SQL> connect scotti@basdat/tiger

Connected.

SQL> select * from scott9.mhs; select * from scott9.mhs
*

ERROR at line 1:
ORA-00942: table or view does not exist
```

Untuk melihat role yang dimiliki anda bis amenjalankan perintah SQL berikut

```
SQL >  select * from role_sys_privs
```

ROLE	PRIVILEGE	AD
CONNECT	ALTER SESSION	NO
CONNECT	CREATE CLUSTER	NO
CONNECT	CREATE DATABASE LINK	NO
CONNECT	CREATE SEQUENCE	NO
CONNECT	CREATE SESSION	NO
CONNECT	CREATE SYNONYM	NO
CONNECT	CREATE TABLE	NO
CONNECT	CREATE VIEW	NO
RESOURCE	CREATE CLUSTER	NO
RESOURCE	CREATE INDEXTYPE	NO
RESOURCE	CREATE OPERATOR	NO
RESOURCE	CREATE PROCEDURE	NO
RESOURCE	CREATE SEQUENCE	NO
RESOURCE	CREATE TABLE	NO
RESOURCE	CREATE TRIGGER	NO
RESOURCE	CREATE TYPE	NO

16 rows selected.

Untuk melihat role yang sedang digunakan

```
SQL > select * from session_roles;
ROLE
------------------------------
CONNECT
RESOURCE
```

5.4 LATIHAN

Dari studi kasus di modul yang pertama buatlah skema user privileges dan role sebagai berikut:

1. Terdapat user biasa (pengunjung) yang dapat melihat dan mencari katalog buku melalui aplikasi yang disediakan (hanya dapat melakukan operasi select). Terdapat user kasir yang otoritasnya hanya dapat melihat data buku dan mengolah/memanipulasi data-data transaksi.
2. Terdapat user administrasi logistik yang memiliki otoritas penuh terhadap semua tabel kecuali tabel-tabel transaksi.
3. Terdapat user manager yang hanya dapat melakukan view ke semua tabel.
4. Terdapat user administrator yang memiliki akses penuh ke semua tabel dan schema dan dapat memberikan akses yang dimilikinya ke user yang lain.

6. MANAJEMEN TRANSAKSI

6.1 INDEX

Index adalah objek schema yang berisi catatan dari nilai-nilai yang muncul pada satu kolom atau kombinasi kolom di index dari sebuah tabel. Index dibuat untuk mempercepat pengaksesan data pada suatu tabel. Index ini dibuat berdasarkan pada field – field dari sebuah tabel. Index bisa dibuat secara otomatis untuk constraint primary key atau unique key dan secara manual melalui CREATE INDEX statement.

a. Membuat Index (CREATE INDEX)

Query :

```
CREATE INDEX nama_index ON
nama_tabel(nama_field1, nama_field2,….);
```

Contohnya :

```
SQL>CREATE INDEX mahasiswa_idx ON
mahasiswa(nim, nama, alamat );
```

b. Memodifikasi Index (CREATE INDEX)

Query :

```
ALTER INDEX nama_index
[INITRANS integer]
[MAXTRANS integer]
[STORAGE storage_clause]
```

Contoh :

```
SQL > ALTER INDEX mahasiswa_idx INITRANS 10;
```

c. Mengubah Nama Index

Queri :

```
ALTER INDEX nama_index_lama RENAME TO nama_index_baru;
```

contoh :

```
ALTER INDEX mahasiswa_idx RENAME TO mhs_idx;
```

d. Menghapus Index (CREATE INDEX)

Query :

```
DROP INDEX nama_index;
```

Contoh :

```
SQL > DROP INDEX mahasiswa_idx;
```

6.2 SEGMEN INDEX

Segmen indeks dibuat ketika indeks diciptakan. Untuk setiap indeks non-partisi akan terdapat satu segmen indeks sedang pada indeks terpartisi, setiap partisi memiliki satu segmen indeks. Pada saat indeks dibuat melalui perintah *create index*, proses server melakukan operasi sort nilai data yang diindeks sebelum berubah menjadi segmen indeks.

Segmen indeks tidak harus berada dalam satu tablespace yang sama dengan segmen datanya. Script di bawah membuat tablespace khusus untuk menampung indeks yang terpisah dengan segmen data.

6.3 MENENTUKAN INDEX

Developer membuat indeks agar unjuk kerja aplikasi lebih baik. Perintah *create index* menghasilkan indeks dengan entry berupa nilai data yang diperoleh dari suatu kolom tunggal, gabungan beberapa kolom, ekspresi, dan fungsi.

Ketika bekerja dengan indeks disarankan untuk mengacu pada kolom-kolom yang diindeks agar meningkatkan performansi join tabel, buatlah indeks dengan urutan kolom-kolom tabel yang tepat atau yang sering digunakan pada klausa *where* agar indeks digunakan untuk pencarian row. Pada pembuatan primary dan unique key secara otomatis akan dihasilkan indeks, begitu juga ketika primary dan unique key itu dihapus maka indeksnya secara otomatis dihapus. Untuk menghindari proses pembuatan indeks secara otomatis itu buatlah terlebih dahulu indeks non-unik berdasarkan primary key dan unique key. Selain itu buatlah juga indeks pada foreign key.

Kolom yang dipilih sebagai bagian dari indeks sebaiknya mengandung nilai data yang unik atau kolom yang sering digunakan dalam klausa *where*. Jika perbedaan nilai data dari suatu atau beberapa kolom sangat bervariasi, gunakan indeks B-Tree. Sedangkan untuk nilai data yang kurang bervariasi gunakan bitmap.

Ekspresi Tunggal

Indeks ini menggunakan satu kolom sebagai kunci indeksnya. Misalnya untuk kemudahan akses data penduduk menurut nama penduduk dilakukan dengan perintah *create index penduduk on kependudukan(upper(nama)).*

Ekspresi Gabungan

Indeks ini dikenal juga sebagai *concatenated* atau *composite index* yaitu indeks yang menggunakan beberapa kolom suatu tabel untuk membentuk indeks entry. Jumlah kombinasinya dapat dibuat hingga 32 kolom namun dalam prakteknya penggunaan lebih dari lima kolom jarang digunakan.

Anggaplah terdapat suatu tabel *kependudukan* yang terdiri dari kolom kabupaten, kecamatan, kelurahan, serta beberapa kolom untuk entitas penduduk. Apabila tabel itu diindeks, kunci indeksnya bisa berupa *create index pddk_ix on penduduk (kab, kec, kel, nama)*. Dengan demikian susunan indeks entrinya diurutkan menurut kabupaten, kecamatan, kelurahan, dan nama penduduk. Untuk memanfaatkan indeks, query harus dilakukan dengan memperhatikan susunan kondisi klausa *where*, misalnya *select * from penduduk where kab='PNK' and kec='SEL' and kel='BANGKA'.*

6.4 JENIS-JENIS INDEX

Untuk performansi query Oracle mendukung penerapan indeks *B-Tree* yang merupakan indeks default, indeks *bitmap* untuk kumpulan key yang cardinality-nya rendah, indeks pada *cluster B-Tree* dan *hash*, indeks global dan lokal untuk partisi tabel, indeks reverse key pada aplikasi real application cluster (RAC), indeks *function-based* pada key yang berupa ekspresi atau fungsi, serta indeks domain untuk aplikasi atau cartridge.

a. Indeks B-Tree

Indeks ini menyimpan key dan *rowid* pada struktur B-Tree untuk menangani transaksi dengan intensitas tinggi dan cocok digunakan pada kolom-kolom tabel dengan cardinality tinggi atau mengandung nilai data yang sangat beragam. Ketika transaksi berlangsung, nilai data pada tabel dan pohon indeks diperbarui. Apabila terjadi query, rowid yang digunakan untuk menemukan letak row data dalam tabel dicari pada indeks. Jadi pada indeks ini rowid untuk setiap key dari masing-masing row tabel akan disimpan dalam indeks.

Perintah *create index boy.kar_idx on boy.karyawan (nik)* akan menghasilkan indeks B-Tree dan cocok untuk menangani transaksi OLTP karena update terhadap kolom yang dindeks dapat berlangsung secara cepat melalui penerapan penguncian pada level row.

b. Indeks Bitmap

Berbeda dengan indeks B-Tree yang secara default menyimpan *rowid*, indeks bitmap menyimpan suatu bitmap untuk setiap nilai kunci pada node *leaf*. Bitmap itu merupakan pengenal yang disusun oleh sejumlah bit dan dipetakan ke rowid. Jika bitnya diset, berarti baris dengan sejumlah rowid yang bersesuaian mengandung nilai key.
Indeks bitmap sangat efektif untuk query yang mengandung banyak kondisi pada klausa *where* dengan *and* dan *or* karena operasi itu secara langsung membandingkan bitmap sebelum mengkonversi bitmap ke rowid. Indeks ini menggunakan space yang lebih kecil dan cocok untuk hardware dengan prosesor dan memori yang terbatas. Indeks bitmap cocok untuk menangani data berukuran besar dengan tingkat transaksi kecil atau pada lingkungan data warehouse serta kolom-kolom dengan cardinilty rendah. Namun mungkin saja DBA membuat indeks bitmap pada kolom dengan cardility yang tinggi untuk lingkungan data warehouse. Untuk data warehouse dengan star schema gunakan indeks bitmap join yang merupakan fungsionalitas baru di Oracle10g.

Jika kolom pada dimension table digunakan untuk membatasi data yang dipilih dari fact tabel (dengan foreign key) dan n dimension tabel (dengan primary key), indeks bitmap join bisa menghindari operasi join antartabel tersebut.
Cardinality rendah merupakan kolom dengan nilai data berulang atau kolom yang perbedaan nilai datanya sangat kecil dibandingkan jumlah row-nya.
Cardinality dapat dilihat pada tabel karyawan berikut ini.

NIK	Nama	Kelamin	Status	Dept
100	Goge	Laki-laki	Menikah	02
101	Titin	Perempuan	Janda	01
102	Beni	Laki-laki	Belum Menikah	03
103	Joel	Laki-laki	Duda	05
104	Susan	Perempuan	Menikah	03

Kolom kelamin, status, dan memiliki cardinality rendah karena itu tepat untuk menggunakan indeks bitmap misalnya *create index boy.kar_depix bitmap on boy.karyawan(dept)*; sedangkan nik dan nama memiliki cardinality tinggi sehingga indeks B-Tree dapat diterapkan.

3. Indeks Reverse

Jika dibandingkan dengan indeks B-Tree, indeks reverse key membalik (reverse) byte setiap kolom yang diindeks (kecuali rowid) dan mempertahankan urutan kolomnya agar perubahan dapat disebar pada beberapa block indeks. Misalnya jika nilai suatu kolom yang diindeks adalah

1234 maka indeks reverse menggunakan angka 4321 agar pemutakhiran pohon indeks tersebar pada beberapa leaf blok. Oleh karena itu indeks ini cocok digunakan jika kolom-kolom yang diindeks memiliki nilai data yang berurutan atau mirip. Indeks ini digunakan pada *real application cluster* (RAC) di mana perubahan indeks dilakukan pada kumpulan blok *leaf* yang kecil.

Dengan me-reverse key yang diindeks maka insert akan tersebar pada berbagai leaf suatu pohon indeks.

—MEMBUAT INDEKS REVERSE.

```
SQL> CREATE INDEX sales_wiltglstok ON
2 sales (wilayah, tgl, stok) COMPRESS REVERSE; SQL> ALTER INDEX
sales_wiltglstok REBUILD;

--Mengubah indeks reverse menjadi noreverse
SQL> ALTER INDEX sales_wiltglstok REBUILD NOREVERSE;
```

Perintah pertama membuat reverse indeks, sedangkan perintah kedua melakukan rebuild indeks. Apabila indeks ini akan diubah ke mode default, gunakan klausa NOREVERSE untuk menormalkan pola penyimpanan key pada pohon indeks.

Pencarian data dengan range-scanning tidak dapat diterapkan pada reverse indeks karena kunci indeks tidak lagi disimpan secara berdekatan sehingga pengambilan data hanya dapat dilakukan melalui key yang ditentukan atau full-table scan.

4. Indeks Fungsi

Indeks ini menggunakan fungsi (*funtion-based index*) untuk mendefinisikan kunci indeksnya.

—MEMBUAT INDEKS FUNGSI.

```
SQL> CREATE INDEX nama_ix ON penduduk(UPPER(nama));

SQL> CREATE INDEX idx1 ON stat_sales(funcsal);

--Menghasilkan statistic index
SQL> ANALYZE INDEX idx1 VALIDATE STRUCTURE;
```

Perintah pertama menghasilkan indeks entry dengan mengkapitalkan nama penduduk melalui fungsi built in upper. Pada contoh kedua digunakan fungsi PL/SQL. *funcsal* yang harus ditentukan *deterministic* dan parameter inisialisasi

QUERY_REWRITE_ENABLED, QUERY_REWRITE_INTEGRITY bernilai TRUE dan TRUSTED. Tabel stat_sales itu dapat dianalisa setelah indeks dibuat dan query harus tidak memerlukan nilai null.

6.5 MENGHINDARI DUPLIKASI DATA

Indeks mengorganisasikan row sehingga kolom-kolom yang digunakan sebagai kunci indeks menyimpan nilai kolom yang ditentukan dalam ekspresi indeksnya. Pada indeks dengan ekspresi tunggal maupun composite di atas, penataan entitas penduduk memungkinkan adanya indeks entri yang sama. Agar indeks memelihara keunikan penduduk sehingga tidak ada duplikasinya, gunakan klausa UNIQUE pada ekspresi indeksnya. Misalnya *create unique index pddk_uq on penduduk(ktp)*. Indeks ini memastikan tidak adanya duplikasi row sehingga bisa menjadi kandidat untuk primary key.

6.6 KEPUTUSAN REBUILD INDEKS

Pemeliharaan indeks untuk membuat ulang (rebuild) indeks dilakukan melalui perintah *alter index…rebuild*. Jika terjadi korupasi indeks, mungkin saja rebuild indeks tidak berhasil karena masih ada korupsi indeks setelah proses rebuild.
Untuk kasus ini, *drop index* kemudian *create index* yang dihapus tersebut.

Jika operasi DML sering dikerjakan, indeks suatu tabel mungkin tidak tersebar secara merata pada pohon indeks. Oleh karena itu perlu pengecekan untuk menentukan bilamana indeks perlu direbuild.

—MEMERIKSA BRANCH LEVEL INDEKS B-TREE.

```
--Ambil statistik indeks
SQL> ANALYZE INDEX tes_idx_idx COMPUTE STATISTICS; Index Analyzed

--Cek BLevel
SQL>   SELECT   index_name,   blevel,   DECODE(blevel,0,'OK
BLEVEL',  1,'OK  BLEVEL',  2,'OK  BLEVEL',  3,'OK  BLEVEL',
4,'OK BLEVEL','BLEVEL HIGH') keterangan
```

```
FROM dba_indexes
WHERE owner='BOY' ORDER BY bleave;
INDEX_NAME                          BLEVEL KETERANGAN
BUDVERPORT_ORG_FK_I                      0  OK BLEVEL
SKS_C006134                              0  OK BLEVEL
BUDVERPORT_BUDVERIORT2_UK                1  OK BLEVEL
BUDVERPORT_PL_TITLE_FK_I                 1  OK BLEVEL
BUDVERPORT_BV_FK_I                       2  OK BLEVEL
BUDVERPORT_DIRCTE_FK_I                   3  OK BLEVEL
S_WAREHOUSE_ID_FK                        4  OK BLEVEL
TES_IDX_IDX                              5  BLEVEL HIGH
A1_PP                                       BLEVEL HIGH
A1_UK                                       BLEVEL HIGH
S_ITEM_ORDID_ PRODID_UK                     BLEVEL HIGH
```

BLEVEL pada data dictionary DBA_INDEXES adalah B-Tree level atau *branch level* yang menunjukkan kedalaman atau level indeks dari node root. Level nol menunjukkan node root dan node leaf yang sama. Jika nilai blevel lebih dari empat maka direkomendasikan untuk me-rebuild indeks. Nilai *blevel* diperoleh setelah indeks dianalisa sehingga nilai *blevel* yang kosong atau keterangan *BLEVEL HIGH* menunjukkan indeks yang belum dianalisa. Untuk itu indeks *tes_idx_idx* dengan blevel 5 perlu di-rebuild dengan perintah *alter index tes_idx_idx rebuild.*

Rebuild Indeks Online

Oracle10g index nama_idx remendukung rebuild indeks dan pembuatan statistiknya secara online dengan perintah *alter index nama_idx rebuild compute statistics online.* Pada versi terdahulu proses itu melibatkan statement *alter index nama_idx rebuild online* dan *alter index nama_idx rebuild compute statistics.*

Mulai Oracle10g, proses tadi dapat dilakukan pada indeks reverse key, function-based maupun indeks reguler dan IOT.

Peningkatan itu memungkinkan user untuk tetap mengakses indeks sementara rebuild dan statistik indeks dibuat. Opsi online memperbolehkan operasi

DML pada tabel atau partisi berlangsung sementara pembuatan indeks dan statistik dikerjakan. Setelah rebuild selesai, indeks yang lama di-drop. Jika opsi *online* tidak disertakan maka tabel akan dikunci hingga proses rebuild indeks berakhir. Jika digunakan opsi *online nologging* maka informasi redo tidak dihasilkan.

6.7 KEPUTUSAN MENGUBAH INDEKS

Perbedaan nilai data kolom (cardinality) yang diindeks juga dapat menjadi acuan untuk keputusan me-rebuild indeks atau mengubah jenis indeks.

—CARDINALITY INDEKS.

```
SQL> ANALYZE INDEX boy.tes_idx_idx VALIDATE STRUCTURE; Index Analyzed

SQL> SELECT del_lf_rows *100/
2                  DECODE(lf_rows,           0,1,lf_rows)
PCT_DELETED,
3      (lf_rows - distinct keys) *100/
4            DECODE(lf_rows,0,1, lf_rows) DISTINCTIVENESS
5      FROM index_stats
6           WHERE NAME='&index_name';
Enter value for index_name: TES_IDX_IDX
```

```
Old     6:    WHERE NAME='&index_name'
New     6:    WHERE NAME='TES_IDX_IDX'

PCT_DELETED DISTINCTIVENESS
----------       ---------------
16.7724777    910.9142073
```

Kolom *pct_deleted* menunjukkan persentase leaf (index entry) yang telah dihapus dan masih belum diisi. Semakin banyak persentasenya , pohon indeks menjadi tidak balance. Sebagai acuan jika *pct_deleted* bernilai di atas 20 persen, indeks itu perlu di-rebuild. Namun angka di atas 10 persen juga dapat dijadikan dasar untuk me-rebuild indeks lebih sering.

Kolom *distictiveness* menunjukkan seberapa sering suatu nilai kolom yang diindeks berulang. Misalnya jika suatu tabel memiliki 10000 row dan ada 9000 variasi nilai untuk kolom yang diindeks maka berdasarkan formula script di atas diperoleh hasil 10. Angka ini menunjukkan distribusi yang baik untuk indeks. Jika untuk 10000 row hanya terdapat variasi dua nilai data maka diperoleh hasil 99,98. Ini berarti hanya sedikit variasi nilai data terhadap seluruh yang ada pada kolom yang diindeks. Kolom ini bukan merupakan calon untuk proses rebuild indeks tetapi sebaiknya dibuatkan indeks bitmap.

6.8 SUMBER INFORMASI

Keberadaan indeks dapat diketahui dengan mengakses data dictionary DBA_IND_CLOUMNS berikut ini:

--MENGAMBIL INFORMASIINDEKS.

```
SQL> SELECT index_name, index_type, status
2      FROM dba_indexes
3             WHERE owner='BOY';

INDEX_NAME                 INDEX_TYPE          STATUS
-----------------------------------------------
PEG_DEP_REVERSE            NORMAL/REV          VALID
KAR_DEP_BITMAP             BITMAP              VALID
KOTA_PENDUDUK_NDX          CLUSTER             VALID
NAMA_IX                    FUNTION-BASED NORMAL VALID
PK PELATIHAN               IOT-TOP             VALID
PROD_IDX                   NORMAL              N/A
SYS_C002976                NORMAL              VALID
SYS_I00000033455C00002$$   LOB                 VALID
TES_IDX_IDX                NORMAL              VALID
…

SQL>SELECT index_name, table_name, column_name
2      FROM dba_ind_columns
3      WHERE index_owner='BOY'
4      ORDER BY table_name;

INDEX_NAME           TABLE_NAME           COLUMN_NAME
-------------------------------------------------
SYS C002985          DAFKURSUS            SYS NC0000600007$
SYS_C002986          DAFKURSUS            SYS_NC_OID$
SYS_IOT_TOP_33369    EMPSUS_TAB           NESTED_TABLE_ID
SYS_IOT_TOP_33369    EMPSUS_TAB           NO
PEG_DEP_REVERSE      PEGAWAI              DEPT_NO
KAR DEP BITMAP       KARYAWAN             DEPT NO
SYS_C003094          PRODUKSI_RANGE       NIK
PROD_IDX             PRODUKSI_RANGE       TGL
TED_IDX_IDX          TES_IDX              A1
…

SQL>SELECT o.object name
2      FROM sys.dba_objects o
3      WHERE owner = 'BOY' AND o.object_id
4      IN (SELECT i.obj# FROM sys.ind$ I WHERE
5             BITAND(i.property,4)=4);

OBJECT_NAME
-----------------------------------------------------
PEG_DEP_REVERSE
```

6.9 PARTISI INDEKS

Seperti halnya table, indeks dapat juga dipartisi. Table terpartisi dapat menggunakan indeks terpatisi maupun indeks non-partisi. Demikian pula sebaliknya suatu table non-partisi dapat menggunakan indeks terpartisi maupun indeks non-partisi.

Indeks Global

Indeks global dapat dipartisi secara range dan cocok digunakan untuk mengakses row secara OLTP. Pada partisi indeks global terdapat batas partisi misalnya *maxvalue*. Penambahan partisi pada indeks global tidak bisa dilakukan karena partisi tertinggi telah dibatasi dengan *maxvalue*. Untuk tujuan itu, partisi harus dibagi melalui statement *alter index…split partition*.

—MEMBUAT INDEKS GLOBAL.

```
SQL> CREATE INDEX prod_idx ON produksi_range(tgl)
 2     GLOBAL PARTITION BY RANGE(tgl)
 3       (PARTITION prod1_idx VALUES LESS THAN(
 4       TO_DATE('01-11-2003','DD-MM-YYYY')),
 5                PARTITION  prod2_idx  VALUES  LESS  THAN
(MAXVALUE));
Index created
```

Setiap partisi indeks itu diberi nama dan disimpan pada tablespace indeks default. Agar pemutakhiran indeks global selalu dilakukan, sertakan klausa *update global indexes* pada setiap statement yang melibatkan operasi DDl pada partisi.

Indeks Lokal

Partisi indeks local umumnya digunakan pada lingkungan data warehouse atau decision support system (DSS) dimana setiap partisi table berhubungan dengan satu partisi indeks local. Untuk lingkungan OLTP dapat digunakan indeks local yang unik dengan ketentuan bahwa partition key suatu table harus merupakan kunci bagi indeks tersebut.

Suatu partisi indeks local bersifat independen sehingga status unusable suatu indeks local tidak mempengaruhi status indeks local lainnya. Penambahan partisi pada indeks local tidak bisa dibuat secara eksplisit tetapi dihasilkan ketika suatu partisi baru ditambahkan. Demikian pula sebalikny, drop partisi dari indeks local hanya dapat dilakukan ketika partisi tablenya di-drop. Indeks local dibentuk menurut struktur table dasarnya dan bersifat equipartitioned sehingga indeks local dipartisi menurut kolom yang sama dengan table dasarnya dan menggunakan jumlah partisi atau subpartisi yang sama pula.

6.10 MENGIDENTIFIKASI INDEKS UNUSED

Indeks mempercepat pembacaan data dengan mengambil rowid dari phon indeks untuk selanjutnya mengambil data di table. Jika pada table terjadi perubahan data dari kolom yang diindeks maka pohon indeks harus dimutakhirkan. Suatu table dapat memiliki beberapa indeks dan mungkin saja dari sekian indeks itu terdapat indeks yang sebenarnya tidak digunakan.

Oracle10g dapat mendeteksi bilamana suatu indeks sedang digunakan atau tidak diperlukan

berdasarkan waktu yang digunakan. Indeks yang tidak digunakan harus di-drop karena menambah overhead. Untuk mengamati indeks gunakan perintah *alter indeks nama_indeks monitoring usage*. Setelah anda yakin bahwa dalam selang waktu tertentu operasi yang melibatkan indeks pasti sudah dilakukan, hentikan pemantauan indeks dengan *alter indeks nama_indeks nomonitoring usage.*

—MEMANTAU INDEKS UNUSED.

```
--Buat table
SQL> CREATE TABLE kanwil (kode NUMBER(5), nama VARCHAR2(10));
SQL> INSERT INTO kanwil VALUES (1,'KANWIL I'); SQL> INSERT INTO kanwil VALUES
(2,'KANWIL II'); SQL> INSERT INTO kanwil VALUES (3,'KANWIL III'); SQL> INSERT INTO
kanwil VALUES (4,'KANWIL IV');
SQL> COMMIT;

--Buat indeks primary key SQL> ALTER TABLE kanwil ADD(
  2 CONSTRAINT kanwil_pk PRIMARY KEY (kode)); Table altered.
--Monitoring indeks belum bekerja
SQL> SELECT index_name, monitoring, used,
  2         start_monitoring, end_monitoring
  3     FROM v$object_usage;
No row selected
```

```
--Aktifkan monitoring indeks
SQL> ALTER INDEX kanwil_pk MONITORING USAGE; Index altered.
--Monitoring indeks diaktifkan
SQL> SELECT index_name, monitoring, used,
  2         start_monitoring, end_monitoring
  3     FROM v$object_usage;
INDEX NAME     MONITORING    USED    START MONITORING     END MONITORING
-----------    ------------  ------  -----------------    --------------
KANWIL PK      YES           NO        01/31/2004 01:30:16
--Buat      table      PLAN_TABLE      jika      belum      ada.      SQL>
@D:\ora9i\rdbms\admin\ut1xplan.sql --Tracing rencana eksekusi
 SQL> SET AUTOTRACE ON EXPLAIN

 SQL> SELECT * FROM kanwil WHERE kode =1;
       KODE  NAMA
 ----------  ------------
             1   KANWIL I
 Execution  Plan
 ----------------------------------------------------------------------------
    0     SELECT STATEMENT Optimizer=CHOOSE
    1     0  TABLE ACCESS (BY INDEX ROWID) OF 'KANWIL'
    2     1       INDEX (UNIQUE SCAN) OF 'KANWIL_PK' (UNIQUE)
SQL> SET AUTOT OFF
--Jalankan query
SQL> SELECT * FROM kanwil WHERE kode =1;
SQL> SELECT index_name, monitoring, used,
   2       start monitoring, end monitoring
     3     FROM v$object t_usage;
INDEX_NAME       MONITORING    USED    START_MONITORING     END_MONITORING
 ----------      ------------  ------  ----------------     --------------
KANWIL_PK        YES           YES       01/31/2004 01:45:47
  --Akhiri monitoring indeks
  SQL> ALTER INDEX kanwil_pk NOMONITORING USAGE; Indes altered.
  SQL> SELECT index_name, monitoring, used,
    2         start_monitoring, end_monitoring
    3     FROM v$object_usage;
  INDEX_NAME   MONITORING  USED    START_MONITORING      END_MONITORING
  -----------  ----------  ----   -------------------   ------------------
  KANWIL PK       NO        YES   01/31/2004 01:45:47   01/31/2004 01:47:07
```

View dictionary V$OBJECT_USAGE berisi informasi indeks yang dimonitor untuk mengetahui indeks yang telah digunakan. Jika kolom used bernilai *yes* maka indeks pernah digunakan selama selang waktu tertentu berdasarkan kolom *start_monitoring* dan *end_monitoring*. Kolom monitoring bernilai *yes* jikamonitoring indeks sedang berlangsung dan berakhir setelah perintah monitor menghentikan monitor indeks dijalankan.

Untuk memantau indeks dalam ruang lingkup database maka perintah alter index…monitoring usage harus dilakukan bagi setiap nama indeks. Untuk itu perlu dibuat script yang menghasilkan statement tersebut dengan membaca data dictionary DBA_INDEXES baik script untuk memulai maupun menghentikan monitoring indeks.

—MEMERIKSA INDEKS UNUSED SCOPE DATABASE.

```
--Buat file script untuk start monitor indeks SQL> SPOOL
D:\STARTMONITOR.SQL
SQL> SELECT 'ALTER INDEX  ||OWNER||
  2          '.'||INDEX_NAME||'MONITORING USAGE;'
  3     FROM DBA INDEXES
  4     WHERE OWNER NOT IN ('SYS','SYSTEM');
SQL> SPOOL D:\STOPMONITOR.SQL
SQL> SELECT 'ALTER INDEX  ||OWNER||
```

```
  2         '.'||INDEX_NAME||'NOMONITORING USAGE;'
  3     FROM DBA INDEXES
  4     WHERE OWNER NOT IN ('SYS','SYSTEM');
SQL> SPOOL OFF
--Edit file startmonitor.sql seperlunya --dan jalankan monitor
SQL> @D:\startmonitor
--Edit file stopmonitor.sql seperlunya
--dan hentikan monitoring indeks setelah periode tertentu SQL> @D:\stopmonitor
--Periksa index yg tidak pernah digunakan selama periode SQL> SELECT d.owner,
v.index_name
  2     FROM dba_indexes d, v$object_usage v
  3       WHERE v.used='NO AND d.index_name=v.index_name;
```

Setelah pemantauan berakhir query dictionaru DBA_INDEXES dan V$OBEJCT_USAGE untuk mengetahui nama indeks yang tidak pernah digunakan melalui kolom unused dan view V$OBEJCT_USAGE.

6.11 VIEW

View adalah sebuah virtual tabel yang dibangun dari satu atau beberapa tabel yang sudah ada, baik berdasarkan kondisi tertentu ataupun tidak. Secara fisik view tidak menyimpan record seperti pada tabel, tetapi ia menyimpan data berupa pointer yang menunjukkan ke record yang bersangkutan di dalam tabel.

Sumber data view dapat berasal dari table atau view lain. Mirip dengan table,
Anda dapat melakukan update, delete, dan insert pada view sehingga perubahan itu akan direfleksikan pada base tabelnya. Berbeda dengan table, view tidak menyimpan data, view hanya menyimpan definisi query pada data dictionary dan tidak memerlukan ruang penyimpanan data. Penerapan view dapat diaplikasikan pada situasi berikut :

- Membatasi akses sesuai otoritas user
- Memudahkan pemahaman tehadap kolom penampung data yang mungkin berbeda dengan definisi kolom pada table dasar
- Menyederhanakan pandangan user terhadap data
- Menangani data kompleks
- Memudahkan penggunaan quey yang berulang karena disimpan sebagai stored query

Membuat View (CREATE VIEW)

Query :

```
CREATE [or replace] [force] [noforce] VIEW nama_view
[(nama_field1, …)]
AS  subquery [with check option]
```

or replace : mendefinisikan kembali view yang sudah ada

noforce : view hanya akan dibuat jika tabel induk telah dibuat.
force : view dapat dibuat walau tabel induk belum dibuat.
with check option : view akan menvalidasi data yang diinsert atau diupdate ke view

Contoh :
SQL > CREATE or REPLACE VIEW mahasiswa_view (nim, nama, alamat)
AS
SELECT nim, nama, alamat
FROM mahasiswa
WHERE alamat != „" with check option;

Memodifikasi View (ALTER VIEW)

Alter view digunakan untuk mengkompilasi ulang sebuah view.

Query :

ALTER VIEW nama_view COMPILE;

Contoh :

SQL > ALTER VIEW mahasiswa_view COMPILE:

Menghapus View (DROP VIEW)

Query :

DROP VIEW nama_view;

Contoh :

SQL > DROP VIEW mahasiswa_view;

View Read-Only

View yang ditujukan hanya untuk dibaca saja, dibuat dengan menyertakan klausa with read only. Sebagai gambarannya, berikut ini didefinisikan objek table dan view yang diciptakan dalam satu kali transaksi melalui create schema.

—MEMBUAT VIEW READ ONLY.

```
SQL> CREATE SCHEMA AUTHORIZATION boni
2      CREATE TABLE boy.dept (
3        no NUMBER (4) NOT NULL PRIMARY KEY,
4        nama VARCHAR2 (20) NOT NULL
5      CREATE TABLE boy.karyawan (
6        nik NUMBER(4) NOT NULL,
7        dept_no NUMBER(4) NOT NULL,
```

```
8         nama VARCHAR2(20),
9         PRIMARY KEY(nik),
10        FOREIGN KEY(dep_no) REFERENCES boy.dept(no)
11        ON DELETE CASCADE
12      CREATE VIEW dep_karyawan_rw
13        AS SELECT a.nik, a.nama AS "Nama Karyawan",
 14               a.dept_no, b.nama "Department"
 15           FROM boy karyawan a, boy.dept b
16           WHEREa.dept_no = b.no
17      CREATE VIEW dep_karyawan_rro
18        AS SELECT a.nik, a.nama AS "Nama Karyawan",
19               a.dept_no, b.nama "Department"
20           FROM boy karyawan a, boy.dept b
21           WHEREa.dept_no = b.no
22        WITH READ ONLY;
Schema created.

SQL> INSERT INTO boy.dept VALUES (100,'SDM');
SQL> INSERT INTO boy.dept VALUES (100,'Produksi'); SQL> INSERT INTO boy.dept
VALUES (100,'Pemasaran'); SQL> INSERT INTO boy.karyawan VALUES (100,'SDM');
SQL> INSERT INTO boy.karyawan VALUES (100,'SDM'); SQL> INSERT INTO boy.karyawan
VALUES (100,'SDM'); SQL> INSERT INTO boy.karyawan VALUES (100,'SDM'); SQL>
COMMIT
Commit complete.
```

View dep_karyawan_ro merupakan view read-only yang menggunakan table karyawan dan dept. View ini hanya dapat dibaca saja dan menghindari manipulasi data dengan delete, insert, atau update melalui view.

—MEMANIPULASI PADA VIEW READ ONLY.

```
SQL> SELECT nik, "Nama Karyawan", "Departmen" 2 FROM dep_karyawan_ro;

       NIK Nama KARYAWAN   Deaprtmen
----------------------------------------------------
         1 Isman                    SDM
         2 Nova              SDM
         3 Donda             Produksi
         4 Rino              Pemasaran

SQL> INSERT INTO dep_karyawan_ro
  2  (nik, dept_no,"Nama Karyawan")
  3    VALUES (5,2.0,'Didik');
   ORA-017333: virtual column not allowed here

SQL> SELECT column_name, updatable
  2    FROM user_updatable columns
  3      WHERE table_name = 'DEP_KARYAWAN_RO';

COLUMN_NAME
-----------------------------------
NIK                 NO
Nama_karyawan NO
DEPT_NO             NO
Department          NO
```

View dep_karyawan_ro didefinisikan dengan empat kolom dimana referensi terhadap kolom karyawan.nama dialisakan menjadi “Deaprtmen”. Ini berarti referensi terhadap kolom itu bersifat case sensitive karena dinyatakan dalam tanda petik “”, pada statement kedua tampak bahwa insert tidak dapat dilakukan ORA17333 da iyu dibuktikan melalui dictionary

USER_UPDATABLE_COLUMNS yang melaporkan bahwa kolom-kolom view itu tidak dapat diupdate.

View Updatable

Meskipun view tidak menyimpan data seperti table, perubahan terhadap base table dapat dilakukan melalui view seperti ditunjukkan pada script berikut :

```
—MANIPULASI PADA VIEW UPDATABLE.


SQL> SELECT column_name, updatable
  2    FROM user_updatable_columns
  3      WHERE table_name = 'DEP_KARYAWAN_RW';

COLUMN NAME         UPD
-----------------------------------
NIK                       YES
Nama Karyawan       YES
Dept_NO                   YES
Departmen                 NO

SQL> INSERT INTO dep_karyawan_rW
  2  (nik, dept_no,"Nama Karyawan")
  3    VALUES (5,200,'Didik');
1 ROW CREATED
```

```
SQL> SELECT * FROM dep_karyawan_rw;

NIK NAma Karyawan   Dep_no Departmen
----------------------------------------
1   Isaman    100   SDM
2   nova            100    SDM
3   donda     200   Produksi
4   rino            300    Pemasaran
5   didik           200    Produksi
```

View dep_karyawan_rw memiliki tiga kolom yang dapat diupdate dan melalui statement insert di atas , penabahan data pada view tersebut akan direfleksikan pada table karyawan.

Materialized View

Materialized view (MV) merupakan objek schema yang berisi hasil query. Tabel-tabel yang digunakan pada query dapat berupa hasil, view atau MV lain yang disebut sebagai table master (replikasi) atau table detail (data warehouse) dan informasinya tersedia pada dictionary ALL_MVIEWS, DBA_MVIEWS, dan UESR_MVIEWS.

Materialized view atau snapshotini digunakan pada database terdistribusi untuk membuat aplikasi dengan sinkronisasi data pada berbagai site maupun untuk data warehouse yang mempersiapkan dan menyimpan data agregat. MV meningkatkan kecepatan akses query melalui perkalkulasi join dan operasi agregat sebelum menjalankan dan menyimpan hasinya pada database. Ketika query terhaap MV dilakukan, query optimizer akan mengetahui bilamana MV yang ada dapat digunakan dan segera mengakses MV, bukan ke table detail (query rewrite)

Membuat Materialized View

Privilege pembuatan MV haus di-grant secara langsung jadi tidak melalui role.
Untuk membuat MV pada Schema user diperlukan priviledge system create materialized view dan create table atau create any table serta previlege system select any table. Sedangkan pembuatan MV pada schema user lain memerlukan previledge system create any materialized view.
Untuk dapat membuat MV yang berisi summary jumlah penduduk setiap kota dapat dibuat denagn cara berikut ini :

—MEMBUAT MATERIALIZED VIEW.

```
 SQL> CREATE MATERIALIZED VIEW ivana.SNAP_KOTA
  2    BUILD IMMEDIATE REFRESH FORCE
  3    OF DEMAND AS
  4    SELECT k.namakota, COUNT(p.nama) jml_penduduk
  5      FROM boy.kota k, boy.penduduk p
  6      WHERE k.nokota=p.nokota
  7      GROUP BY k.namakota;
Materialize view created

SQL> SELECT * FROM ivana.Snap_kota; NAMAKOTA JML_PENDUDUK
-------------------------------------------
Pontianak                  700000
mempawak                   1200000
sintang                           500000
...
```

Perintah diatas menghasilkan data dalam MV (build immediete) ynag mengguanakn metode refresh force untuk memilih pemutakhiran data secara incremental atau komplit MV_sap_kot itu tidak bias digunakan untuk query rewrite. Agar kemampuan itu tersedia maka opsi with query rewrite enabled harus dipilih dengan syarat owner memiliki privileged system query rewrite.

Materialized view juga dapat dibat melalui OEM Console
Untuk memeriksa integritas struktur materialized view seperti halnya table, index, arau cluster gunakan perintah analyze. Miasalnya *analyze table sanp_kota validate structure cascade*. Jika objek tidak valid, lakukan refresh MV secar komplit namun jika masih tidak valid lakukan drop dan buat ulang materiliazednya.
Perintah *alter materialized view snap_kota compile* melakukan vaildasi MV secara eksplist dan digunakan pada situasi dimana telah terjadi perintah drop atu alter tterhadap objek-objek yang

digunakan oleh MV. Perintah alter ini juga digunakan untuk mengubah karakteristik MV seprti metode dan mengaktifkan query rewrite.

Materialized View Read/Write

MV dapat ditujukan hanya untuk dibaca saja sehingga menghindari perubahan MV da data pada table master. Sebagai conroh daftar kota yang terdapat pada table master kota dapat dibuatkan MV-nya dengan nama sanp_kota sehingga user dapat mengakses data melalui MV itu.

MATERIALIZED VIEW UNTUK READ/WRITE.

```
SQL> CREATE MATERIALIZED VIEW Ivana.SNAP_KOTA
  2    BULID IMMEDIATE REFRESH FORCE
  3    ON DEMAND AS
  4      SELECT *FROM boy.kota; Materialized view created.

SQL> INSERT INTO Ivana.SNAP_KOTA
  2    VALUES (157000,'Poso','Large');
ORA-01732: data manipulation operation not legal on this view

SQL> DROP MATERIALIZED VIEW Ivana.SNAP_KOTA; MATERIALIZED view dropped.

SQL> CREATE MATERIALIZED VIEW Ivana.SNAP_KOTA
  2    BULID IMMEDIATE REFRESH FORCE
  3    ON DEMAND FOR UPDATE
  4    AS
  5      SELECT * FROM boy.Kota; Materilaized view created.

SQL> INSERT INTO Ivana.SNAP_KOTA
  2    VALUES (157000,'pOSO','lage');
1 ROW CREATED.
```

Untuk membuat MV yang dapat diupdate, tambahkanklausa for update pada definisi MV.

Refresh Data

Data dalam MV diperbaharui jika ada perubahan pada table master.
Pemutakhirannya dapat dilakukan secara incremental (fast refresh) atau memutakhirkan semua data (complete) atau memilih cara refresh yang tersedia (force). Jika MV menggunakan metode fast refresh, materialized view log akan merekam perubahan terhadap table master. MV dapat di refresh secara periodic (automatically on), sesuai keperluan (on demand) atau jika
MV itu berada pada database yang sama dengan table masternya, refresh terjadi setelah commit (on commit).
Untuk dapat me –refresh MV secara on commit diperlukan privilege system on commit refresh atau dengan privilege objek on commit refresh pada setiap table master yang tidak dimiliki user. Jika table detail atau master mempunyai primary key, gunakan opsi primary key, sebaiknya gunakan metode refresh yang menggunakan rowed.

Materialized View Log

Materialized view log adalah objek schema yang mencatat perubahan yang terjadi pada table sehingga memungkinkan table master diperbaharui secara incremental (fast refresh). Berikut ini dibuat MV log untuk data warehouse dengan fast refresh yang menggunakan rowed.

```
—MEMBUAT MATERIALIZED VIEW LOG.


SQL> CREATE MATERIALIZED VIEW LOG ON boy.kota
  2    WITH SEQUENCE, ROWID (nokota,namakota,kecamatan)
  3    INCLUDING NEW VALUES;
Materialized view log created.

SQL> CREATE MATERILIZED VIEW LOG ON boy.penduduk
  2    WITH SEQUENCE, ROWID (noktp, nama, nokota)
  3    INCLUDING NEW VALUES;
Materialized view log created.
```

Untuk mendukung fast refresh pada MV tampak bahwa definisi MV log menyertakan klausa ROWID yang disertai including new values untuk merekam niali dat yang lama dan baru pada log.

Data Warehouse

Materialized view digunakan untuk organisasi data pada warehouse, misalnya dalam pembuatan ringkasan penduduk per kota melalui script berikut ini :

```
—MATERIALIZED VIEW UNTUK DATA WAREHOUSE.


SQL> CREATE MATERIALIZED VIEW boy.jiwa_kota
  2    BUILD IMMEDIATE
  3    REFRESH FAST
  4    ENABLE QUERY REWRITE
  5    AS
  6    SELECT  k.namakota,  k.kecamatan,  COUNT(p.nama)  AS
Jiwa
  7           FROM boy.kota k, boy.penduduk p
  8             WHERE k.nokota = p.nokota]
  9 GROUP BY k.namakota, k.kecamatan; Materialized view created.
```

Statement diatas membuat MV jwa_kota dengan menghitung jumlah penduduk dalam satu kota dan kecamatan dengan operasi join. MV segera dipopulasikan karena menggunakan metode bulid immediate dan tersedia untuk digunakan melalui query rewrite. Metode fast refresh bias dilakukan karena MV log telah dibuat untuk table kota dan penduduk.

6.12 SEQUENCE

Sequence digunakan untuk membangkitkan serangkaian nilai serial yang unik.

Membuat Sequence (CREATE SEQUENCE)

Query :

```
CREATE SEQUENCE nama_sequence [INCREMENT BY integer]
      [START WITH  integer]
      [MAXVALUE  integer  |  NOMAXVALUE]  [MINVALUE  integer  |
      NOMINVALUE] [CYCLE | NOCYCLE]
      [CHACE | NOCHACE]
```

INCREMENT BY berfungsiuntuk mendefinisikan jumlah incrementasi setiap kali terjadi penyisipan record.

START WITH berfungsi untuk mendefisikan bilangan awal yang dibangkitkan.

NOMAXVALUE tidak ada batas maximum bilangan sequence yang digenerate.

MAXVALUE mendefinisikan maximum bilangan sequence yang digenerate.

CYCLE mendefinisikan bahwa jika telah bilangan sequence telah maximum, maka nilai akan diulang dari awal lagi

NOCYCLE tidak ada pengulangan bilangan bila telah sampai nilai maximum

CACHE bilangan sequence akan ditampung di buffer

NOCACHE bilangan sequence tidak akan ditampung di buffer.

Contoh :

```
SQL > CREATE SEQUENCE seq_bulan
        INCREMENT BY 1
        START WITH 1
MAXVALUE 12;
```

Memodifikasi Sequence (ALTER SEQUENCE)

Query :

```
ALTER SEQUENCE nama_sequence
      [INCREMENT BY integer]
      [START WITH integer] [MAXVALUE integer | NOMAXVALUE]
      [MINVALUE integer | NOMINVALUE]
      [CYCLE  | NOCYCLE]
      [CHACE | NOCHACE]
```

Contoh :

```
SQL > ALTER SEQUENCE seq_bulan
        INCREMENT BY 2;
```

Menghapus Sequence (DROP SEQUENCE)

Query :

```
DROP SEQUENCE nama_sequence;
```

Contoh :

```
DROP SEQUENCE seq_bulan;
```

6.13 LATIHAN

1. Sebutkan dan jelaskan keuntungan jika kita menggunakan index!
2. Apakah semakin banyak index yang kita pakai maka semakin baik pula pengaruhnya terhadap performansi DBMS?
3. Berdasarkan studi kasus yang ada di modul 1 lakukan analisis terhadap tabel-tabel yang ada untuk menentukan index yang akan dibuat kemudian buatlah index pada tabel-tabel tersebut dan berikan alasan kenapa index tersebut harus dibuat!
4. Apakah syarat-syarat yang harus dipenuhi sehingga sebuah View dapat dinyatakan sebagai view updatable?
5. Buatlah contoh view updatable berdasarkan studi kasus yang ada di modul 1!
6. uatlah view yang isinya merupakan hasil outer join dari tabel buku dan jenis dimana tabel buku menjadi acuannya!
7. Jelaskan fungsi/kegunaan dari sequence jika dikaitkan dengan tabel!
8. Berdasarkan studi kasus yang ada di modul 1 lakukan analisis terhadap tabel-tabel yang ada untuk menentukan sequence yang akan dibuat kemudian buatlah sequence pada tabel-tabel tersebut dan berikan alasan kenapa sequence tersebut harus dibuat!

7. KONTROL KONKURENSI

7.1 LOCKS

Istilah lock pada database berarti suatu kunci atau penguncian. Lock digunakan pada saat terjadi pengaksesan database oleh user secara bersamaan. Hal ini ditujukan untuk menjaga data agar tidak corrupt / hilang atau data yang tidak valid pada saat banyak user mengakses database

secara bersamaan / konkuren. Sebelum database mengijinkan suatu sesi user untuk memodifikasi data, pertama-tama sesi me-lock data yang akan dimodifikasi. Lock memberikan exclusive control terhadap suatu data pada sebuah sesi sehingga tidak akan ada transaksi lain yang dapat memodifikasi data yang ter-lock sampai lock terhadap data tersebut telah di lepas. Transaksi dapat melakukan lock

- pada *sebuah row* data dari suatu tabel,
- *multiple row* data dari suatu tabel, atau
- *satu tabel* itu sendiri yang di lock.

Oracle 10g memiliki mekanisme lock secara manual dan otomatis.

Ada dua mekanisme untuk melakukan locking data pada database yaitu :

1. Pessimistic Locking
   Proses locking yang terjadi pada saat lock di request maka record atau table akan di lock secara immediate / langsung.

2. Optimistic Locking
   Proses locking yang terjadi pada saat suatu record / data pada database dilakukan suatu perubahan / update.

7.2 LOCKING MECHANISM

Locking mekanisme di disain untuk menyediakan derajat konkurensi yang tinggi yang terjadi pada database. Suatu transaksi yang memodifikasi data memperoleh *row-level locks* daripada *table-level locks.* Proses query pada suatu data tidak membutuhkan mekanisme lock terhadap data yang di query, sebuah query tetap berhasil dieksekusi pada suatu data atu tabel walaupun ada user yang member lock pada stuatu data atau tabel. Pada saat multiple transaksi membutuhkan suatu lock pada resource yang sama, transaksi yang pertama meminta lock pada suatu resource tersebut. Transaksi yang lainnya berada pada fase menunggu di antrian sampai transaksi yang pertama telah selesai mengeksekusi resource. Mekanisme antrian ini bersifat otomatis dan tidak membutuhkan interaksi dari suatu administrator atau user yang lain.

Semua lock di lepas pada saat transaksi berakhir. Transaksi dikatakan selesai / complete apabila perintah *commit atau rollback* dilaksanakan. Pada kasus transaksi yang failed, baground proses secara otomatis melakukan rollback kegagalan yang terjadi dengan merubah transaksi yang gagal tersebut menjadi ke posisi *save point* atau sebelum proses transaksi gagal.

7.3 DATA CONCURRENCY

Mekanisme lock secara default berada pada row-level locking mode. Transaksi yang berbeda dapat mengupdate row data yang berbeda yang berada pada sebuah tabel yang sama tanpa bertentangan antara yang satu

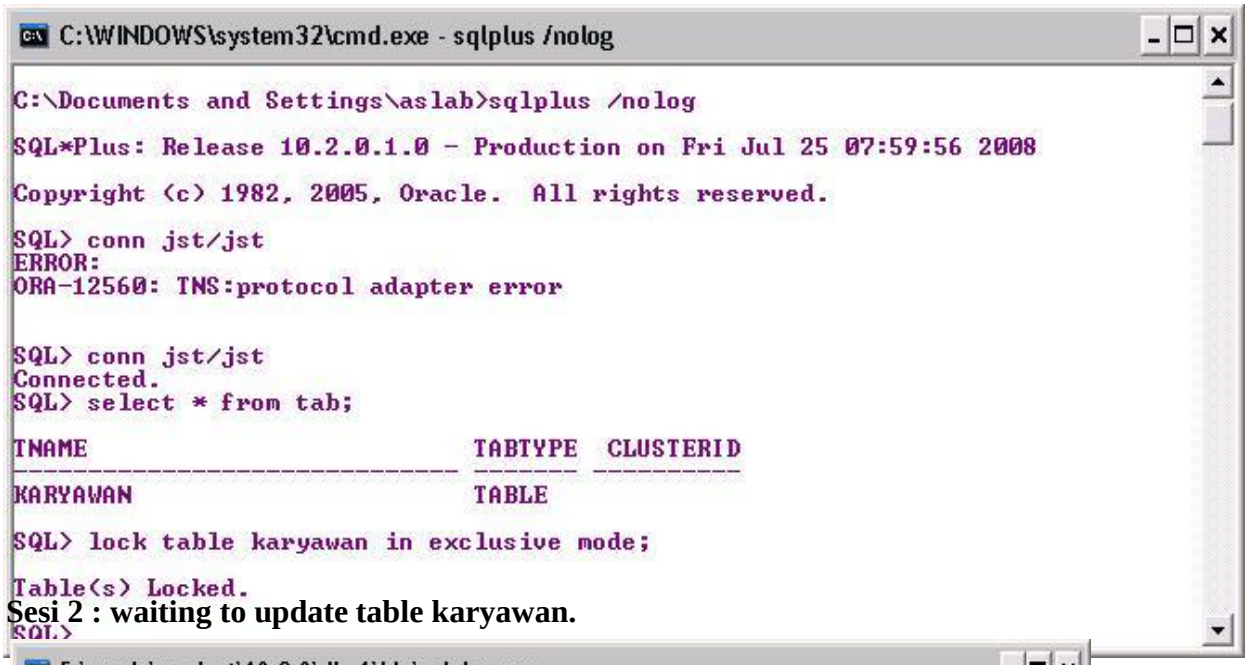
SQL > LOCK TABLE karyawan IN EXCLUSIVE

dengan yang lain. Sementara default model lock berapada pada row-level,

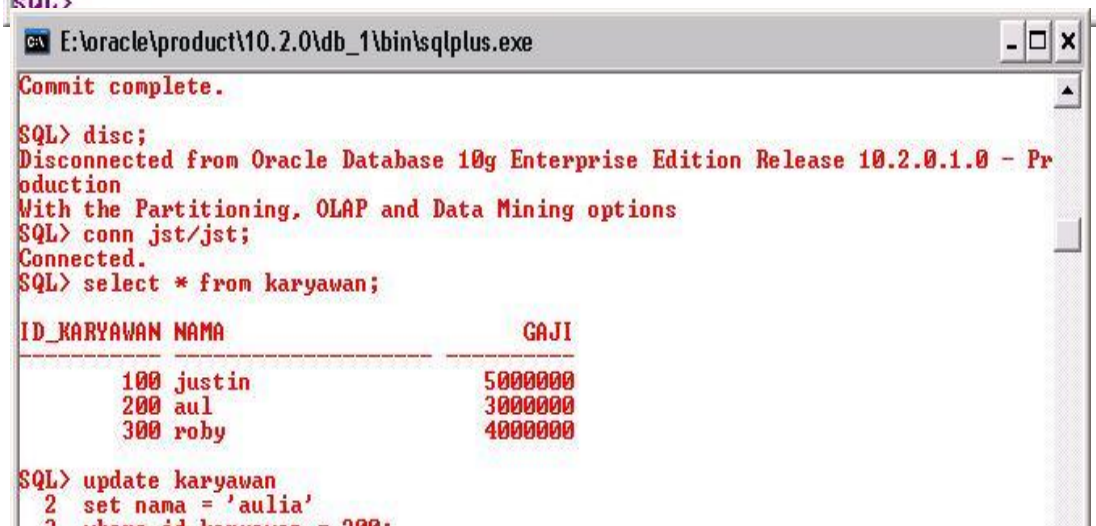
Oracle Database 10*g* mensupport manual locking konfigurasi pada level yang lebih tinggi. Berikut contoh locking suatu tabel

Dengan statement diatas, transaksi yang lain yang mencoba untuk meng-update data pada tabel yang sudah di lock (karyawan) harus menunggu pada antrian sampai transaksi yang mempunyai lock pada tabel tersebut telah selesai (commit).

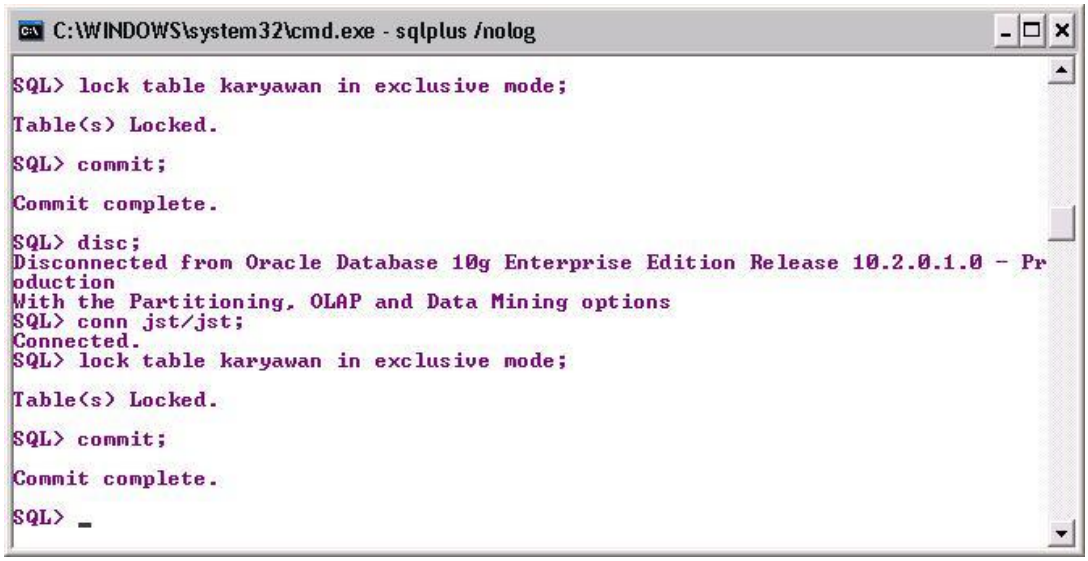
Sesi 1 : locking table karyawan in exclusive mode



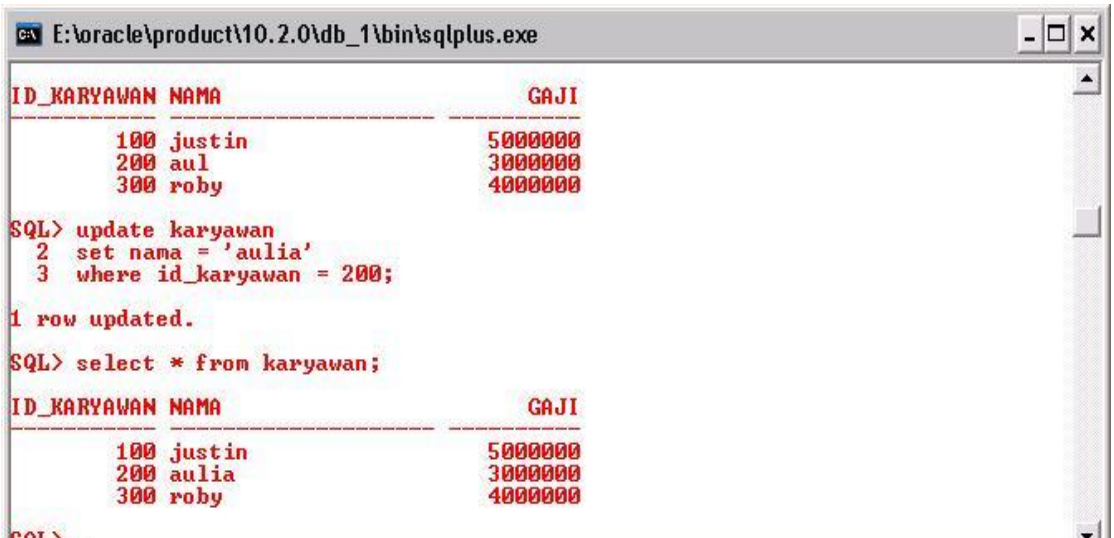
Sesi 2 : waiting to update table karyawan.



Sesi 1 : commit (transaksi telah berakhir dan locking terhadap tabel karyawan di release)



Sesi 2 : traksaksi update success



EXCLUSIVE lock adalah lock yang paling tinggi derajatnya. Berikut ini beberapa lock mode yang ada pada Oracle :

1. ROW SHARE
   Mengijinkan konkuren akses / akses secara bersama-sama pada tabel yang di lock, tetapi user yang lain tidak dapat member lock exclusive pada tabel yang diberi share lock sampai share lock di lepas. user yang tidak melakukan lock pada tabel, masih dapat mengakses data.

2. ROW EXCLUSIVE
   Sama dengan ROW SHARE, tetapi tidak diperbolehkan user lain untuk memberi SHARE mode sebelum lock sudah di lepas.Terjadi secara otomatis pada saat terjadi updating, inserting, atau deleting suatu data.

3. SHARE
   Mengijinkan proses query yang konkuren / bersama-sama tetapi tidak diperbolehkan untuk mengubah mode lock tabel. SHARE lock dibutuhkan (dan secara otomatis diminta) untuk membuat index pada tabel.

4. SHARE ROW EXCLUSIVE
   Digunakan untuk melakukan query pada keseluruhan data pada tabel dan memberi akses kepada sesi user lain untuk melakukan query pada tabel tetapi tidak diperbolehkan sesi user yang lain untuk memberi lock pada tabel serta tidak diperbolehkan untuk mengupdate data pada tabel.

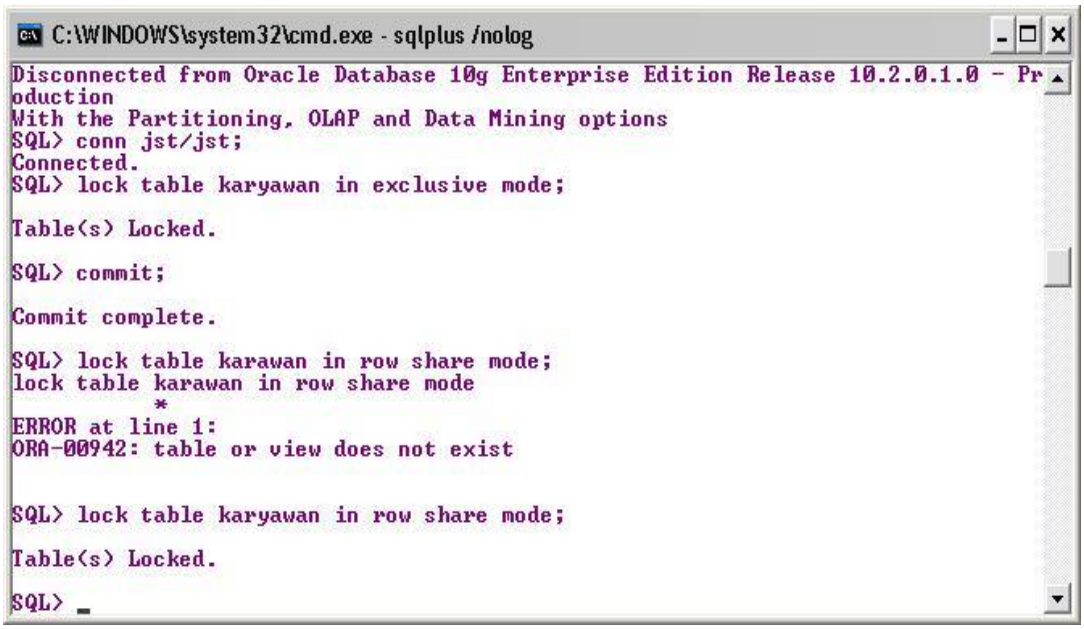
5. EXCLUSIVE
   Mengijinkan sesi user yang lain untuk melakukan query pada tabel yang di lock tetapi tidak diperbolehkan melakukan suatu aktivitas yang lain selain query pada tabel tersebut. Exclusive lock dibutuhkan pada saat proses drop suatu tabel.

Seperti beberapa permintaan untuk lock, eksekusi statement lock secara manual terdapat antrian sampai sesi statement yang lain telah memiliki lock, atau telah melepas lock. perintah LOCK dapat menerima special argument untuk mengontrol proses menunggu yaitu menggunakan NOWAIT.

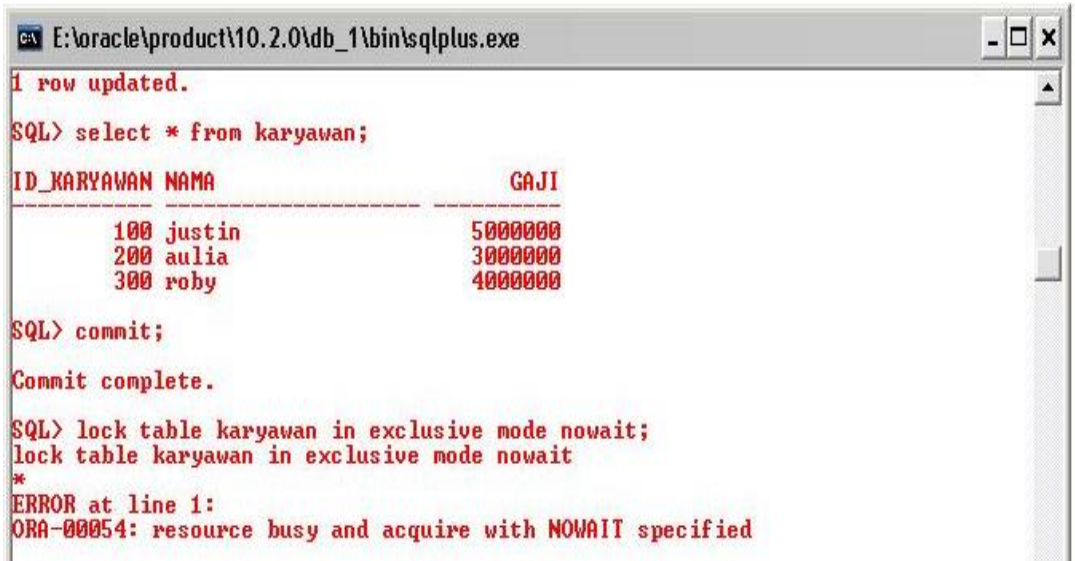
QL > LOCK TABLE nama_tabel IN mode_lock NOWAIT;

NOWAIT mengembalikan control kepada sesi yang meminta lock secara langsung jika tabel yang ingin di lock telah di lock oleh sesi user yang lain.

Sesi 1 melakukan lock tabel karyawan pada mode row share



Sesi 2 ingin melakukan lock pada tabel karyawan tetapi di tolak



7.4 DML LOCKS

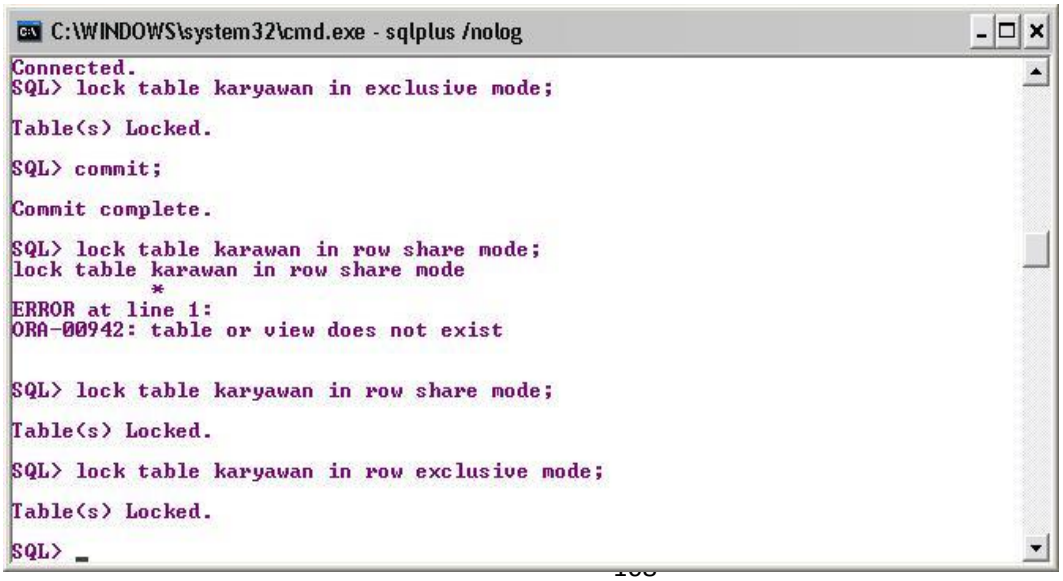
Setiap transaksi DML terdapat 2 lock :

1. Row exclusive lock pada sebuah data atau banyak data yang sedang di update.
   Hal ini akan menjadi row exclusive lock berdasarkan pada jumlah data yang diupdate.

2. Shared table level lock pada tabel yang sedang di update.
   Hal ini mencegah sesi user lain untuk melakukan lock keseluruhan tabel (kemungkinannya untuk drop tabel atau truncate data tabel) sementara perubahan sedang terjadi.

7.5 ENQUEUE MECHANISM

Permintaan suatu lock secara otomatis akan di masukkan ke dalam antrian. Setelah suatu transaksi yang memegang lock telah selesai, maka sesi berikutnya yang telah mengantri terlebih dahulu akan mendapatkan lock. suatu sesi yang sedang memegang lock pada suatu data atau tabel dapat me-request untuk mengubah mode lock yang sedang terjadi tanpa harus melepas sesi lock dan kembali ke antrian. Sebagai contohnya, misalkan terdapat sebuah sesi yang memegang mode shared lock pada suatu tabel. Sesi tersebut dapat merubah dari shared lock ke exclusive lock, selama tidak ada sesi lain yang sedang memiliki exclusive lock atau share lock pada tabel tersebut. Maka sesi yang memegang shared lock akan di beri grant / hak akses untuk mendapatkan exclusive lock tanpa harus menunggu di antrian terlebih dahulu.



7.6 LOCK CONFLICTS

Konflik pada suatu transaksi database sering terjadi, tetapi pada umumnya dapat di selesaikan berdasarkan periode waktu dan mekanisme antrian. Ada suatu kasus yang sangat jarang terjadi lock konflik dibutuhkan konfigurasi dari administrator untuk menyelesaikan konflik. Berikut contoh konflik yang jarang terjadi :

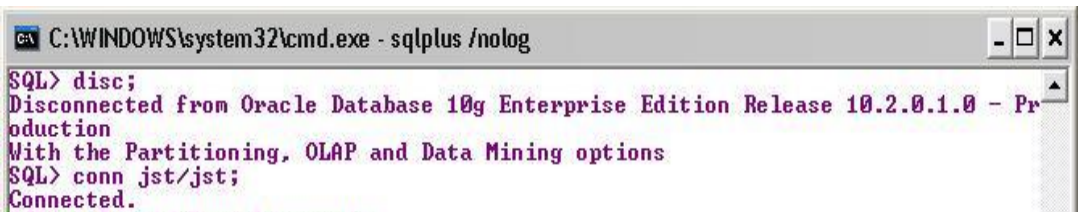
Transaction 1	Time	Transaction 2
UPDATE hr.employees SET salary=salary+100 WHERE employee_id=100 ; 1 row updated.	9:00:00	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=101; 1 row updated.
UPDATE hr.employees SET COMMISION_PCT=2 WHERE employee_id=101 ; Sesi menunggu karena terjadi lock konflik.	9:00:05	SELECT sum(salary) FROM hr.employees; SUM(SALARY) ---------- 895463
Sesi masih menunggu di antrian!!	16:30:00	Terjadi transaksi yang banyak dalam waktu 7.5 jam, tetapi belum ada commit atau rollback
1 row updated.	16:30:01	Commit;

pada kasus diatas, transaksi 2 mendapatkan lock single row pada jam 9:00:00 dan tidak melakukan commit. Di sisi lain transaksi 1 menunggu untuk proses update pada tabel, tetapi transaksi 2 pada saat itu membutuhkan lock pada semua data. Sehingga transaksi 1 di block oleh transaksi 2 sampai transaksi 2 commit pada jam 16:30:01. Untuk hal ini user yang mengeksekusi transaksi 1 menghubungi administrator untuk mengatasi permasalahan ini.

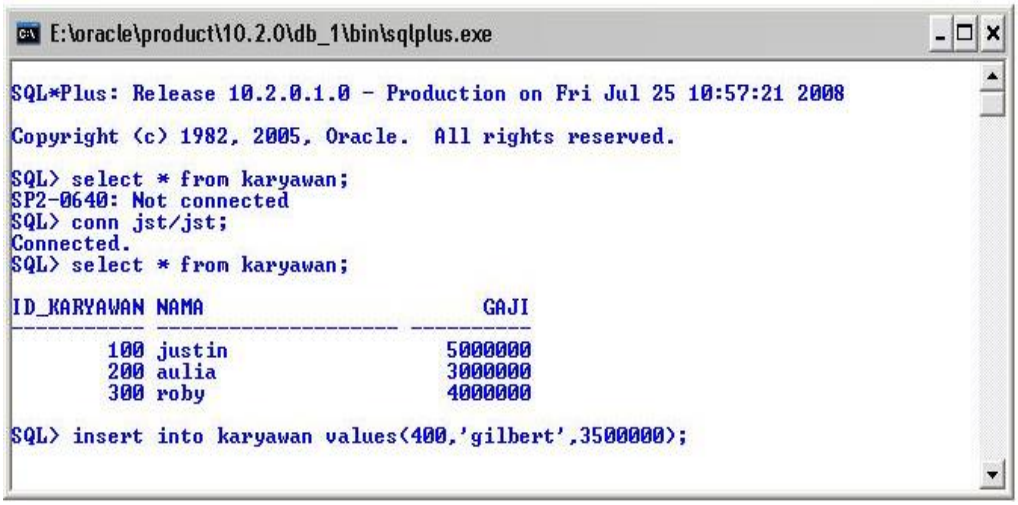
7.7 DETECTING LOCK CONFLICTS

Pada oracle 10g untuk mendeteksi lock konflik kita dapat menggunakan Enterprise Manager. Berikut langkah-langkah mendeteksi lock. Misal sebelumnya sudah ada transaksi berikut :

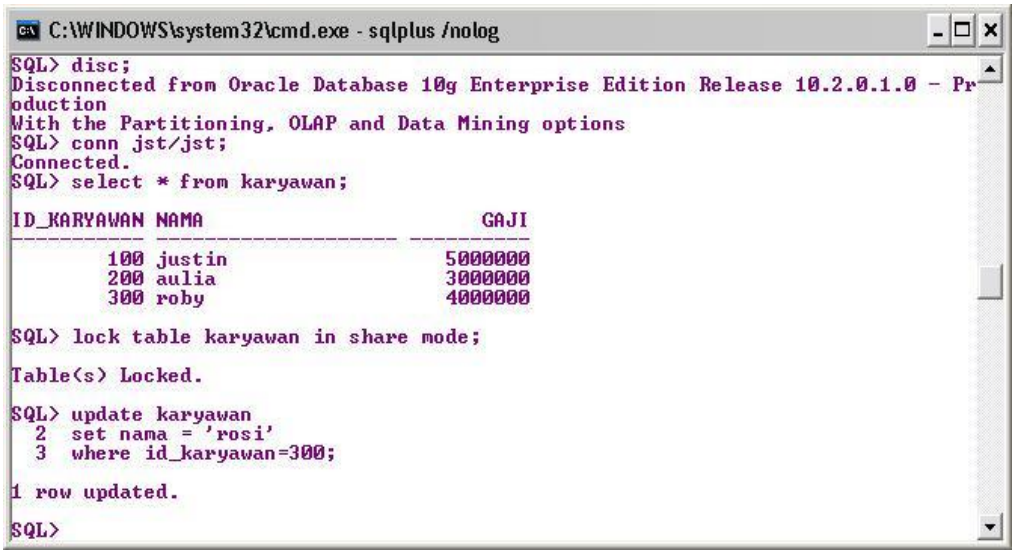
Sesi 1 lock tabel karyawan



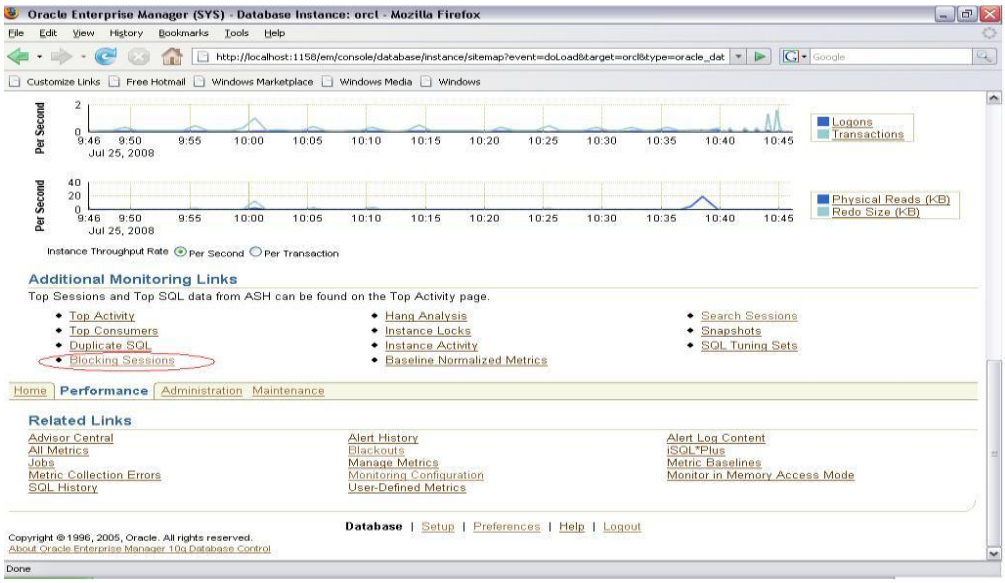
Sesi 2 update tabel karyawan



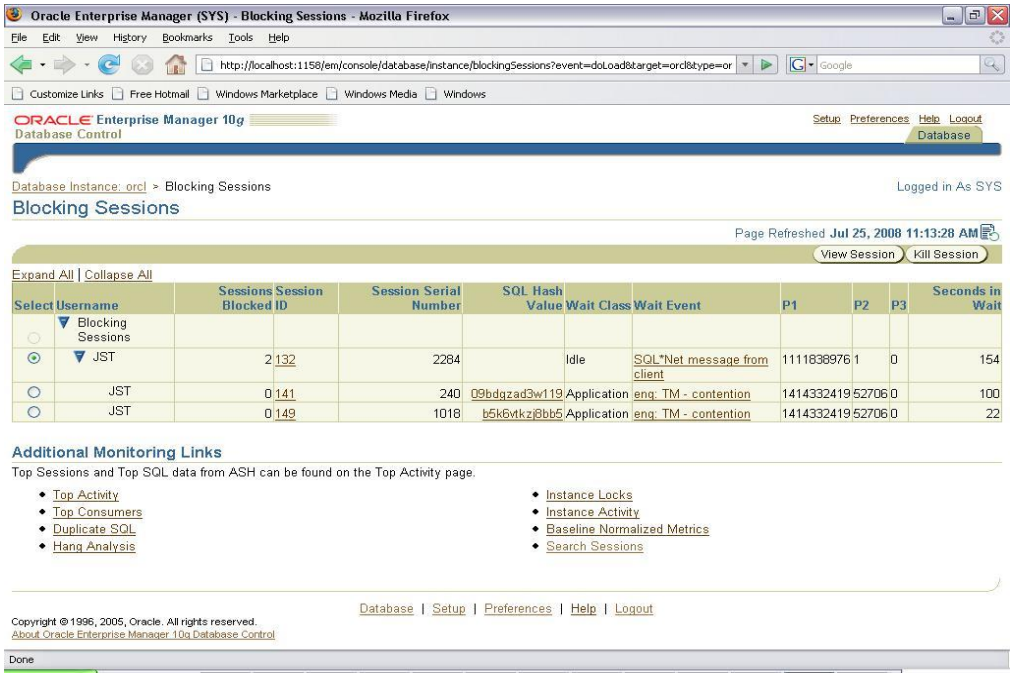
Sesi 3 insert tabel karyawan



1. Pada Enterprise Manager buka halaman Performance
2. Klik Blocking pada bagian kiri bawah halaman.



3. Halaman Blocking Sessions muncul. Terdapat tiga sesi user yang sedang mengalami konflik.



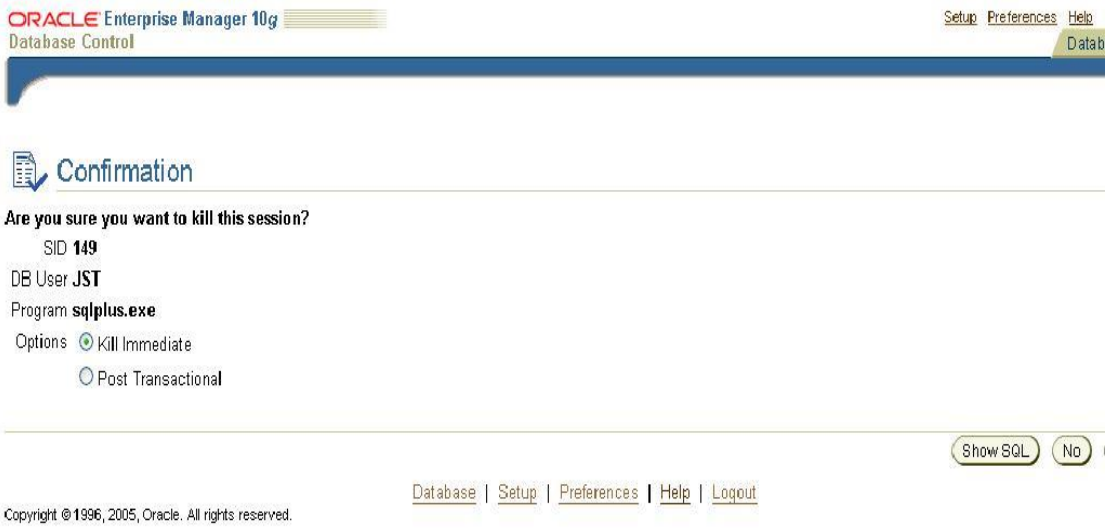
7.8 RESOLVING LOCK CONFLICTS

Untuk menyelesaikan lock konflik, suatu sesi yang memegang lock harus di lepaskan. Ada beberapa cara untuk menyelesaikan masalah lock konflik, cara yang paling baik ialah dengan cara member tahu si user yang sedang memanipulasi data untuk melakukan commit atau rollback terhadap transaksi yang sedang dijalankan. Jika terjadi suatu keadaan yang mendesak, penyelesaian lock konflik memungkinkan administrator untuk men-terminasi sesi user yang sedang memegang lock dengan cara meng-klik tombol **Kill**

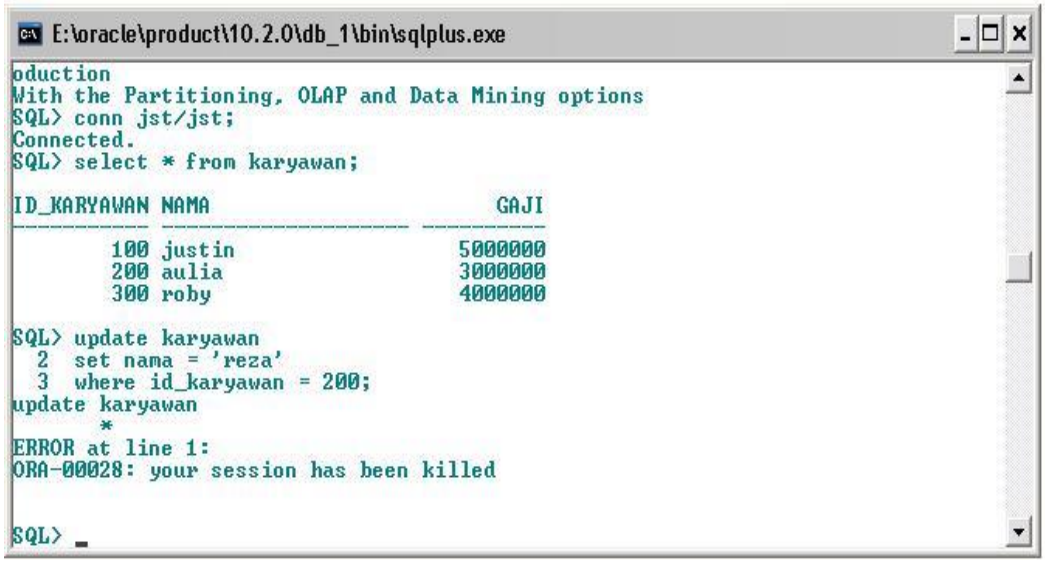
Session. Saat suatu sesi di terminate / di stop, semua transaksi yang sedang terjadi pada sesi akan di rollback. User yang sesinya di terminate harus login terlebih dahulu untuk melakukan transaksi yang telah di terminate.

Berikut langkah-langkah untuk men-terminate sesi user :

1. pada halaman Blocking Sessions, pilih sesi user yang akan di terminate. Kemudian klik Kill Session.
2. Pesan konfirmasi muncul, pilih kill Immediete. Kemudian klik Yes



3. Pada sisi user yang di-terminate akan muncul pesan your session has been killed dan transaksi secara otomatis mengalami rollback.



7.9 DEADLOCKS

Deadlock merupakan contoh lock konflik yang sangat special. Deadlock terjadi pada saat terdapat dua atau lebih sesi yang sedang saling menunggu antar satu sama lain untuk mendapatkan hak atas lock. dikarenakan sesi antar satu dengan yang lain saling menunggu dan tidak dapat menyelesaikan transaksi masing-masing maka akan terdeteksi sebagai deadlock.

Transaksi 1	Time	Transaksi 2
UPDATE karyawan SET nama = „paijo" WHERE employee_id = 400;	9:00	UPDATE karyawan SET nama = „paimin" WHERE employee_id = 300;
UPDATE karyawan SET nama = „paul" WHERE employee_id = 300;	9:15	UPDATE karyawan SET nama = „gilbert" WHERE employee_id = 400;

Oracle secara otomatis mendeteksi dan menyelesaikan deadlock dengan cara me-rollback semua transaksi yang sedang mengalami deadlock

7.10 LATIHAN

1. Coba buat suatu kasus yang menunjukkan suatu kondisi deadlock
2. Bagaimana tahap-tahap penyelesaian lock konflik
3. Buatlah suatu kasus yang menunjukkan lock terjadi hanya pada suatu row
4. Buatlah suatu kasus yang menunjukkan lock terjadi hanya pada suatu tabel

8. PL/SQL ORACLE

8.1 PENDAHULUAN

Karena pada SQL tidak mendukung pemrograman secara prosedural, maka dikembangkanlah PL/SQL. PL merupakan kependekan dari Procedural Language. PL/SQL mengkombinasikan kekuatan dan kefleksibilitasan SQL dengan pemrograman prosedural. PL/SQL memiliki keistimewaan sebagai berikut:

- Programmer dapat mendeklarasikan variable untuk digunakan selama pemrosesan statement.
- Programmer dapat menggunakan penanganan kesalahan untuk mencegah kegagalan program.
- Programmer dapat menulis program yang interaktif yang menerima input dari user.

- Programmer dapat membagi fungsi-fungsi ke dalam blok-blok lojik dari kode. Teknik pemrograman secara modular ini mendukung fleksibilitas selama pengembangan aplikasi.
- Statement SQL dapat diproses secara simultan untuk performansi keseluruhan yang lebih baik

8.2 VARIABEL DAN TIPE DATA

Deklarasi Variabel dan Tipe Data

Bagian deklarasi variabel di antara kata kunci DECLARE dan BEGIN.

Penamaan variabel tidal bersifat case sensitive. Tipe data variabel dapat berupa salah satu tipe data database Oracle atau tipe data built in PL/SQL.
Sintaks:

```
Identifier typedata [(presisi, skala)]

[NOT NULL] [:=iekspresi];
```

Dimana ekspresi bisa merupakan literal, variabel yang lain atau sebuah ekspresi yang terdiri atas operator dan fungsi. Jika nilai inisial/awal tidak diberikan, maka suatu variabel akan diberikan nilai NULL untuk nilai inisialnya.
Contoh untuk data karakter:

```
alamat VARCHAR2(20); kodepos
CHAR(5) := '40257';
```

Contoh untuk tipe data number:

```
gaji NUMBER(7,2);
jumlah NUMBER NOT NULL := 0;
```

Contoh untuk tipe data tanggal:

```
alamat VARCHAR2(20); kodepos
CHAR(5) := '40257';
```

Tanda := dipakai untuk menugaskan (*assign*) nilai kepada suatu variabel. Nilai inisial/awal atau disebut juga nilai default adalah nilai yang diberikan pada saat awal variabel tersebut dideklarasikan.

Tipe Data pada PL/SQL

Selain tipe data yang ada di Oracle SQL, PL/SQL menyediakan beberapa tambahan tipe data yang dapat dideklarasikan sebagai berikut:

Tipe Data	Keterangan
BOOLEAN	Data lojikal dengan nilai TRUE atau FALSE.

DATE	Data tanggal waktu. Nilai yang valid adalah antara 1 Januari 4712 SM sampai dengan 31 Desember 9999.
NUMBER [(*p* [,*s*])]	Tipe data numerik dengan *p* angka penting dan sejumlah *s* angka penting di belakang koma. Nilai *p* adalah integer dengan nilai maksimal 38 dan nilai *s* berada pada rentang -84 sampai dengan 127. Nilai *s* negatif berarti pembulatan sampai dengan 10^s terdekat.
FLOAT	Turunan dari NUMBER. Presisi sampai dengan 38 digit.
DOUBLE PRECISION	Sama dengan FLOAT.
REAL	Turunan dari number. Presisi sampai dengan 18 digit.
DEC [(*p* [,*s*])]	Sama dengan NUMBER [(*p* [,*s*])].
DECIMAL [(*p* [,*s*])]	Sama dengan NUMBER [(*p* [,*s*])].
NUMERIC [(*p* [,*s*])]	Sama dengan NUMBER [(*p* [,*s*])].
INTEGER [(*n*)]	Sama dengan NUMBER [(*n*,0)].
INT [(*n*)]	Sama dengan NUMBER [(*n*,0)].
SMALLINT [(*n*)]	Sama dengan NUMBER [(*n*,0)].
BINARY_INTEGER	Tipe variabel ini digunakan menyimpan nilai mulai dari -2.147.483.647 s/d 2.147.483.647
NATURAL	Bagian dari binary integer, mampu menyimpan mulai dari 0 s/d 2.147.483.647.
NATURALN	Bagian dari binary integer, mampu menyimpan mulai dari 0 s/d 2.147.483.647. Tipe data ini tidak boleh bernilai NULL.
POSITIVE	Bagian dari binary integer, mampu menyimpan mulai dari 1 s/d 2.147.483.647
POSITIVEN	Bilangan integer dengan rentang nilai 1 sampai dengan 2147483647. Tipe data ini tidak boleh bernilai NULL.
SIGNTYPE	Tipe data bilangan yang bernilai -1, 0 atau 1.
PLS_INTEGER	Bilangan integer dengan rentang nilai -2147483647 sampai 2147483647.
VARCHAR2(*n*)	Data karakter dengan panjang tidak tetap. Nilai *n* minimum sama dengan 1 dan maksimum sama dengan 32767 byte.
VARCHAR(*n*)	Sama dengan VARCHAR2(*n*).
CHAR [(*n*)]	Data karakter dengan panjang tetap sebesar *n* byte. Nilai *n* maksimum adalah 32767. Nilai *n* minimum dan juga nilai default adalah 1.
STRING(*n*)	Sama dengan VARCHAR2(*n*).
CHARACTER [(*n*)]	Sama dengan CHAR(*n*).
LONG [(*n*)]	Data karakter dengan panjang tidak tetap. Nilai *n* maksimum sama dengan 32760 byte.
NCHAR [(*n*)]	Data karakter dengan panjang tetap. Panjang maksimum sama dengan 32767 byte. maksimum bergantung pada *national character set* yang dipakai. Nilai default adalah 1.
NVARCHAR2(*n*)	Data karakter dengan panjang tidak tetap. Panjang maksimum sama dengan 32767 byte. Nilai *n* maksimum bergantung pada *national character set* yang dipakai.

RAW(*n*)	Data binary dengan panjang tidak tetap. Nilai *n* maksimum sama dengan 32767 byte.
LONG RAW [(*n*)]	Data binary dengan panjang tidak tetap. Nilai *n* maksimum sama dengan 32760 byte.
ROWID	Identitas baris pada suatu tabel-index yang dinyatakan dengan string heksa desimal. Identitas tersebut menunjukkan posisi baris data. Tipe data ini merupakan balikan dari kolom palsu ROWID.
UROWID [(*n*)]	Identitas baris pada suatu tabel-index yang dinyatakan dengan string heksa desimal. Nilai *n* adalah ukuran kolom UROWID. Nilai *n* maksimum adalah 4000 byte.
BFILE	Tipe data large object untuk data file.
BLOB	Tipe data large object untuk karakter binary.
CLOB	Tipe data large object untuk karakter satu byte.
NCLOB	Tipe data large object untuk karakter multi byte.
%TYPE	Untuk mendeklarasikan variabel dengan tipe data yang sesuai dengan suatu kolom pada suatu tabel.
%ROWTYPE	Untuk mendeklarasikan variabel dengan tipe data yang sesuai dengan semua kolom pada suatu tabel. Biasanya untuk menampung suatu cursor.

Pendeklarasian Konstanta

Sintaks:

```
Identifier CONSTANT typedate[(presisi,skala)] := ekspresi;
```

Contoh:

```
pi CONSTANT NUMBER(9,2):=3.14;
```

Atribut Variabel

Jika menggunakan variabel yang menampung nilai dari suatu kolom dari suatu tabel, maka sebaiknya menggunakan atribut variabel. Hal ini untuk menghindari terjadinya kerepotan seperti: user harus melihat struktur tabel yang terkait terlebih dahulu untuk memberikan tipe data yang cocok. Selain itu jika terjadi tipe data kolom maka deklarasi variabel tersebut harus diubah juga.

Atribut variabel berfungsi untuk memberikan tipe data yang sama dengan tipe data suatu kolom dari suatu tabel. Atribut variabel juga dapat digunakan untuk tipe data record. Dengan demikian, atribut variabel ada dua. Untuk atribut kolom digunakan %TYPE, sedangkan untuk atribut record gunakan %ROWTYPE. Cara penggunaannya ditunjukkan berikut ini:

```
[schema.]table.column%TYPE;

<cursor_name|[schema.]table>%ROWTYPE;
```

Sebagai contoh, varaiabel v_nama mempunyai tipe data yang sama dengan kolom nama pada

tabel pegawai. Deklarasi variabel tersebut dapat dituliskan seperti ini:

```
v_nama mahasiswa.nama%type;
```

Selain dapat digunakan untuk variabel record, atribut %ROWTYPE bisa dipakai pada variabel cursor. Dan untuk mengakses baris-baris pada cursor atau record tersebut digunakan format nama_var.COLUMN.

Contoh:

```
pgw_rec pgw_cur%rowtype;
```

Menugaskan Nilai ke Dalam Variabel Sintaks:

```
identifier := ekspresi;
```

Dimana identifier adalah nama variabel target, atau field untuk menerima nilai dari ekspresi. Sedang ekspresi mungkin berupa literal, nama variabel lain yang sudah ada, atau suatu ekspresi yang cukup kompleks yang diperlukan untuk menentukan suatu nilai yang akan ditugaskan.

Contoh:

```
v_jumlah := 0;
```

Operator pada PL/SQL

Operator-operator SQL statement juga berlaku pada PL/SQL. Berikut ini prioritas dari semua operator tersebut ditampilkan pada tabel di bawah ini dengan prioritas yang lebih tinggi ditempatkan pada baris yang lebih atas:

Operator	Operasi
**, NOT	Pemangkatan dan negasi logika
+, -	Tanda positif dan negatif
*, /	Perkalian dan pembagian
+, -, ||	Penjumlahan., pengurangan Dan konkatinasi
=, <, >, <=, >=, <>, !=, IS NULL, LIKE, BETWEEN, IN	Perbandingan
AND	Konjungsi
OR	inklusi

Mencetak Keluaran pada Layar SQL*Plus
Untuk mencetak sebuah nilai pada layar SQL*Plus dapat digunakan procedure

PUT, PUT_LINE dan NEW_LINE yang terdapat dalam package DBMS_OUTPUT. Package ini merupakan salah satu package yang telah built in pada Oracle.

Procedure PUT dan PUT_LINE membutuhkan sebuah argumen berupa NUMBER, VARCHAR2 ataupun DATE. Kedua procedure tersebut akan menyimpan argumen tersebut ke dalam buffer

dan akan ditampilkan di layar bila procedure tersebut dijalankan.

Procedure NEW_LINE tidak membutuhkan argumen apapun. Procedure ini berfungsi untuk menyimpan karakter new line ke dalam buffer.

Namun sebelum procedure tersebut dijalankan, harus dijalankan perintah "SET SERVEROUTPUT ON" untuk mengaktifkan pencetakan ke layar dengan menggunakan procedure yang ada pada package DBMS_OUTPUT.

```
SET SERVEROUTPUT <ON|OFF> [SIZE n] [FOR[MAT]
      <WRA[PPED] | WOR[D_WAPPED] | TRU[NCATED]]
```

Pilihan SIZE n bertujuan untuk menentukan jumlah byte maksimum yang dapat ditampung oleh buffer. Nilai n ini tidak boleh kurang dari 2000 dan tidak boleh lebih dari 1.000.000. Nilai defaultnya 2000. pilihan format bertujuan untuk menetukan format keluaran. WRAPPED akan melanjutkan bagian yang tidak mencukupi dari suatu baris ke baris yang baru. Pilihan TRUNCATE akan memotong bagian yang melampaui ukuran satu baris tepat pada karakter yang berada setelah batas maksimum baris. Ukuran satu baris, yakni jumlah karakter maksimum dalam satu baris, ditentukan oleh sistem variabel LINESIZE.

8.3 STRUKTUR BLOK PL/SQL

Terdapat dua macam blok pada PL/SQL yaitu blok bernama dan blok tidak bernama (anonymous block), dimana blok-blok ini akan membentuk suatu unit PL/SQL. Blok-blok yang bernama bisa disimpan dan dapat berupa procedure, function serta trigger. Sedangkan blok yang tidak bernama tidak dapat disimpan dalam database kecuali jika dipakai sebagai subblok dalam sebuah unit PL/SQL bernama.

Secara umum, satu blok PL/SQL yang lengkap terdiri atas tiga bagian, yaitu: declaration section (bagian deklarasi variabel), executable section (bagian pengeksekusian) serta exception section (bagian penanganan kesalahan). Berikut ini penggambarannya:

```
[DECLARE
     ...]     → DECLARATION SECTION
BEGIN
              →
     ...        EXECUTABLE SECTION
[EXCEPTION
```

```
    ...]    → EXCEPTION SECTION
END;
```

Dengan declaration dan exception bersifat opsional, maka satu bok PL/SQL paling tidak terdiri atas executable section.
Contoh:

```
begin
      null;
end;
```

Catatan: Null dipakai untuk menyatakan nilai yang tidak diketahui, sehingga untuk contoh di atas, blok PL/SQL tersebut tidak melakukan proses apapun.

1) DECLARATION SECTION

Digunakan untuk mendefinisikan atau mendeklarasikan variabel, konstanta, cursor dan seluruh exception yang didefinisikan oleh user yang akan digunakan pada bagian eksekusi. Penulisan blok ini dimulai dengan menulis DECLARE.

Contoh:

```
declare
v_nama      mahasiswa.nama%type;
v_nim       mahasiswa.nim%type;
```

2) EXECUTABLE SECTION

Digunakan untuk mengeksekusi atau menjalankan blok perintah PL/SQL seperti pengulangan, percabangan, perintah SQL dan perintah cursor. Berisi statement SQL untuk memanipulasi data pada basis data dan statement PL/SQL untuk memanipulasi data dalam blok.

Contoh:

```
declare
v_nama      mahasiswa.nama%type;
v_nim       mahasiswa.nim%type;
begin
      select nim, nama into v_nim,
      v_nama from pegawai
      where nim=30108001
      dbms_output.put_line(v_nama);
      exception
  when no_data_found then
  dbms_output.put_line('gak ada');
end;
```

3) EXCEPTION SECTION

Merupakan bagian yang akan diaktifkan bila terjadi kesalahan atau pengecualian pada saat menjalankan program PL/SQL. Exception section terdiri atas predefined dan user defined. Sebagai contoh exception predefined

NO_DATA_FOUND akan diaktifkan bila perintah DML SQL tidak menemukan data dalam klausa WHERE.

Contoh:

```
Declare
v_nama      mahasiswa.nama%type;
v_nim       mahasiswa.nim%type;
Begin
      select nim, nama into v_nim,
      v_nama from pegawai
      where nim=30108001
      dbms_output.put_line(v_nama);
      exception
         when no_data_found then
         dbms_output.put_line('gak ada');
end;
```

8.4 STRUKTUR KONDISIONAL

Perintah IF terdiri atas tiga bentuk, yaitu IF THEN, IF THEN ELSE, serta IF THEN ELSEIF. Struktur dari ketiganya ditampilkan dalam satu rumusan umum sebagai berikut:

```
IF kondisi 1 THEN
      Baris perintah...
[ELSIF kondisi 2 THEN
      Baris perintah...]
...
[ELSE
      baris perintah..] END IF;
```

Baris pada *baris perintah* dapat berupa perintah IF sehingga akan membentuk blok IF bersarang. Bagian ELSIF bisa muncul beberapa kali sesuai dengan kebutuhan sedangkan bagian ELSE biasanya dipakai untuk menangani

kondisi jika semua kondisi pada kalang IF... THEN atau ELSIF... THEN tidak terpenuhi. Namun bagian ELSE ini bisa saja tidak digunakan.

contoh:

```
declare
   penuh exception;
   n1 number;
   n2 number;
begin
   if b1>n2 then
   raise penuh;
else
      dbms_ouput.put_line('bisa');
end if;
end;
```

8.5 STRUKTUR ITERASI

Pernyataan Loop

Untuk perintah LOOP, akan dilakukan pengulangan terus-menerus. Bentuk umum dari pernyataan LOOP sebagai berikut:

```
LOOP
      //Baris perintah
END LOOP;
```

Karena tidak mempunyai kondisi untuk keluar dari iterasi, maka perlu digunakan perintah EXIT. Perintah EXIT dapat digunakan dengan cara seperti berikut:

```
EXIT WHEN kondisi;
```

Contoh:

```
DECLARE
   x number;
BEGIN
   x := 0;
   LOOP
      x := x + 1;
      EXIT WHEN x > 5; -- exit loop
   immediately END LOOP;
   dbms_output.put_line('Hasil looping :
   '||x); END;
```

Bisa juga digunakan format seperti ini:

```
IF kondisi THEN
      EXIT;
END IF;
```

Contoh:

```
DECLARE
   x number;
BEGIN
   x := 0;
   LOOP
      x := x + 1;
      IF x > 5 THEN
            EXIT; -- exit loop
      immediately END IF;
      dbms_output.put_line('Hasil looping ke-
      '||x); END LOOP;
END;
DECLARE
   vno number; BEGIN
vno:=1; LOOP
      insert into coba(no) values
      vno; vno:=vno+1;
      if vno > 10 then
            exit;
      end
if; END
LOOP; END;
```

Pernyataan LOOP bisa diberi label atau nama, sintaksnya sebagai berikut :

```
<<label_name>>
LOOP
      sequence_of_statements END
LOOP label_nama; //optional
```

Contoh:

```
<<outer>>
LOOP
  ...
  LOOP
      ...
      EXIT outer WHEN ... -- exit both
  loops END LOOP;
  ...
END LOOP outer;
```

Pada contoh diatas, saat **EXIT** maka akan keluar dari kedua looping yang ada.

Pernyataan While - Loop

Perintah WHILE-LOOP akan terus melakukan iterasi (memproses baris perintah secara berulang) selama KONDISI bernilai TRUE. Bentuk umum dari pernyataan LOOP sebagai berikut:

```
WHILE kondisi LOOP
      //Baris perintah
END LOOP;
```

Contoh:

```
DECLARE
   x number;
BEGIN
   x := 0;
   WHILE x <= 5 LOOP
      x := x + 1;
      dbms_output.put_line('Hasil looping ke-
      '||x); END LOOP;
END;
```

Selain dapat digunakan pada perintah LOOP, perintah EXIT ini juga dapat digunakan pada WHILE-LOOP untuk menambahkan kondisi tertentu.

Namun perintah EXIT ini hanya bisa digunakan dalam loop saja.

Contoh:

```
DECLARE
   vno number;
BEGIN
vno:=1;
WHILE vno <= 10 LOOP
      insert into coba(no) values
      vno; EXIT WHEN vno = 10;
      vno:=vno+1;
END LOOP;
END;
```

Pernyataan For - Loop

Struktur pengulangan *For* digunakan untuk menghasilkan pengulangan sejumlah kali tanpa penggunaan kondisi apapun. Stuktur ini menyebabkan aksi diulangi sejumlah beberapa kali

(tertentu). Bentuk umum struktur *for* ada dua macam yaitu : menaik (*ascending*) atau menurun (*descending*). Sintaksnya sebagai berikut :

```
FOR  counter  IN  [REVERSE]  i_terendah  ..  i_teratas
LOOP
      Baris perintah
END LOOP;
```

Perintah FOR-LOOP melakukan iterasi selama nilai COUNTER berada dalam range nilai *i_terendah* dan *i_teratas*. Pada FOR-LOOP, *counter* tidak perlu dideklarasikan. Penggunaan kata kunci RESERVE akan menyebabkan nilai counter dimulai dari i_teratas ke i_terendah. Dua titik antara i_terendah dan i_teratas merupakan operator rentang nilai. i_terendah maupun i_terkecil bisa berupa nilai integer ataupun variabel yang bernilai integer yang sudah dideklarasikan sebelumnya. i_upper harus lebih besar dari i_lower dan jika tidak maka iterasi tidak akan dilakukan.

Contoh:

```
BEGIN
FOR vno IN 1..10 LOOP
      insert into coba(no) values vno;
      dbms_output.put_line('Hasil looping ke- '||x);
END LOOP;
END;
```

```
BEGIN
FOR vno IN REVERSE 1..10 LOOP
      insert into coba(no) values vno; dbms_output.put_line('Hasil looping ke-
      '||x);
END LOOP; END;
```

Ruang Lingkup Pernyataan FOR – LOOP

Contoh :

```
DECLARE
  ctr INTEGER; -- global
variable BEGIN
  ...
  FOR ctr IN 1..25 LOOP
      ...
      IF ctr > 10 THEN ... -- refers to loop
      counter END IF;
  END
  LOOP; END;
```

Untuk menuju ke variabel global, harus ditambahkan label dan notasi *dot*.

Contoh :

```
<<main>>
DECLARE
  ctr INTEGER;
  ...
BEGIN
  ...
  FOR ctr IN 1..25 LOOP
      ...
      IF main.ctr > 10 THEN -- refers to
global variable
        ...
      END IF;
      END LOOP;
END main;
```

Hal ini juga berlaku untuk *nested loop.*

Contoh :

```
<<main>> DECLARE
  ctr INTEGER;
  ...
```

```
BEGIN <<outer>>
  FOR step IN 1..25 LOOP FOR step IN 1..10 LOOP
        ...
        IF outer.step > 15 THEN
            ...
        END IF; END LOOP;
  END LOOP outer; END main;
```

Selain dapat digunakan pada perintah LOOP, perintah EXIT ini juga dapat digunakan pada FOR-LOOP untuk menambahkan kondisi tertentu. Namun perintah EXIT ini hanya bisa digunakan dalam loop saja.

Contoh:

```
BEGIN
   FOR j IN 1..10 LOOP FETCH
      c1 INTO mhs_rec; EXIT
      WHEN c1%NOTFOUND;
      ...
  END
  LOOP; END;
```

```
BEGIN <<outer>>
  FOR i IN 1..5 LOOP
       ...
       FOR j IN 1..10 LOOP FETCH
         c1 INTO mhs_rec;
         EXIT outer WHEN c1%NOTFOUND; -- exit
         both FOR loops
         ...
       END LOOP;
       END LOOP outer;
-- control passes here

END;
```

Perintah GOTO

Perintah ini digunakan untuk mengarahkan proses ke baris yang ditandai dengan label tertentu. Bentuk umum pemakaian perintah ini adalah:

```
GOTO nama_label;
```

Untuk memberikan label pada suatu baris tertentu, gunakan format penamaan label seperti berikut ini:

```
<<nama_label>>
```

Penggunaan perintah GOTO dalam jumlah yang banyak akan menyebabkan suatu blok PL/SQL menjadi tidak terstruktur. Karena itu sebaiknya penggunaan GOTO ini dihindari.

Contoh:

```
DECLARE
   x number;
BEGIN
   x := 0;
   LOOP
      x := x + 1;
      IF x = 5 THEN
            GOTO
      EndOfLoop; END IF;
   END LOOP; <<EndOfLoop>>dbms_output.put_line
   ('Hasil looping
: '||x);
END;
```

Contoh:

```
create or replace procedure coba
       (v_nim mahasiswa.nim %type) is
```

```
      cursor mhs_cur is
        select nim, nama, alamat from mahasiswa
        where nim=v_nim;

mhs_rec mhs_cur%rowtype;




begin
 open mhs_cur;

 <<iterasi>>
 fetch mhs _cur into mhs _rec;

 if mhs _cur%notfound then
      goto lbl_end;
 end if;

 dbms_output.put_line(mhs _rec.nama_pegawai||'
 '|| mhs_rec.alamat||' '|| mhs_rec.gaji);

 goto iterasi;
```

Dengan adanya perintah "goto iterasi", proses berikutnya menuju baris "<<iterasi>>" yang berada beberapa sebelum baris goto tersebut. Selanjutnya, perintah-perintah yang mengikutinya akan diproses sesuai dengan urutannya (sekuensial). Sedangkan perintah "<<lbl_end>>" menentukan proses berikutnya adalah baris "<<lbl_end>>" yang berada setelah perintah goto tersebut.

Namun demikian, pada saat menggunakan perintah goto harus diperhatikan hal-hal berikut:

- perintah goto tidak boleh menuju label yang berada dalam perintah IF, LOOP, blok lain dan blok yang menjadi sub bloknya.
- Sebuah label harus diikuti oleh paling tidak sebuah perintah eksekusi PL/SQL. Kata atau reserved word seperti END, END IF dan END

LOOP tidak termasuk sebagai perintah eksekusi. Tetapi NULL termasuk perintah eksekusi.

8.6 LATIHAN

1. Buatlah tampilan seperti ini pada SQL Plus:

1

3

5

7

9

11

2. Buatlah tampilan seperti ini pada SQL Plus:

   1 3 5 7 9 11

3. Jelaskan maksud blok PL/SQL dibawah ini, jika nilai variabel input_nilai = 8:

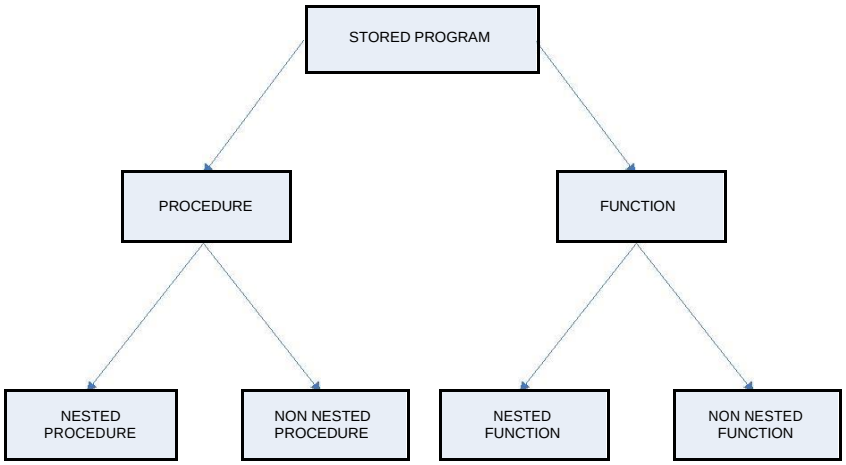
```
DECLARE nilai1
number;
nilai2
number; nilai3
number; hasil
number; BEGIN
nilai1:=1;
nilai2:=1;
nilai3:='&input_nilai';
dbms_output.put_line(nilai1||' ');
dbms_output.put_line(nilai2||' ');
loop
      hasil:=nilai1+nilai2;
      if hasil>=nilai3 then
            goto
      endloop; end if;
      dbms_output.put_line(hasil||' '); nilai1:=nilai2;
      nilai2:=hasil;
end loop; <<endloop>>
dbms_output.put_line(' ');
END;
/
```

4. Jelaskan maksud blok PL/SQL dibawah ini, jika nilai variabel input_nilai = 8:

```
DECLARE
      batas_b number;
      batas_a number;
  BEGIN
      batas_b:='&batas_bawah';
      batas_a:='&batas_atas';
```

```
      while batas_b<=batas_a loop if
            mod(batas_b,2)=1 then
                  dbms_output.put_line(batas_b||'
');
            end if;
            batas_b:=batas_b+1;
      end loop; dbms_output.put_line(' ');
END;
/
```

9. STORED PROCEDURE



9.1 PROCEDURE

Non Nested Procedure

Non nested Procedure atau stored procedure merupakan sekumpulan blok PL/SQL yang tersimpan di dalam skema database dan dapat dieksekusi secara berulang kali jika user memiliki privilege untuk mengeksekusi procedure tersebut. Nama procedure yang dibuat nantinya akan menjadi objek dengan tipe procedure. Procedure akan dieksekusi pada saat pemanggilan setelah sebelumnya dibuat terlebih dahulu.

Sintaks Pendeklarasian:

```
CREATE [OR REPLACE] PROCEDURE nama_procedure
      [argumen1 tipe_data, argumen2 tipe_data,...]
AS
      [deklarasi variabel lokal]
BEGIN
      badan_prosedur
EXCEPTION END;
```

Keterangan

nama_procedure	: nama dari procedure yang akan dibuat
argumen	: parameter-parameter yang dipakai saat pemanggilan procedure
badan_prosedur	: tempat blok PL/SQL yang memuat baris kode.

Contoh :

```
CREATE OR REPLACE PROCEDURE tambahMahasiswa
(
      v_nim       mahasiswa.nim%type;
      v_nama      mahasiswa.nama%type;
      v_alamat    departemen.lokasi%type
)

IS
BEGIN
      insert into mahasiswa
      values(v_nim,v_nama,v_alamat);
END;
```

Setelah procedure dibuat, dijalankan dengan sintaks berikut.

```
EXECUTE nama_procedure(parameter_1,parameter_2,...);
```

Contoh :

```
EXECUTE tambahMahasiswa
('30108002','paimen','sukapura');
```

Untuk menghapus procedure yang telah kita buat, dapat menggunakan sintaks berikut

```
DROP PROCEDURE nama_procedure;
```

Contoh :

```
DROP PROCEDURE tambahMahasiswa;
```

Nested Procedure

Adalah procedure yang dideklarasikan di dalam Declaration Section suatu blok PL/SQL yang anonim. Karena dideklarasikan di dalam blok PL/SQL yang anonim maka nested procedure tidak disimpan dalam basis data dan tidak dapat dijalankan oleh blok lain kecuali dideklarasikan kembali oleh blok tersebut.

Contoh :

```
DECLARE
  PROCEDURE CETAK(kata varchar2, n number) IS
      BEGIN
        FOR i IN 1..n LOOP
            DBMS_OUTPUT.PUT_LINE(to_char(i)||
            '. '||kata);
        END LOOP;
      END CETAK;

BEGIN
      CETAK('&v_kata', &v_n);
END;
```

Jenis-jenis Procedure pada Oracle

PROCEDURE TANPA PARAMETER/ARGUMEN
Procedure yang tidak memiliki parameter/argument biasanya bersifat statis

(outputannya selalu sama) setiap kali dieksekusi.

Contoh:

```
CREATE OR REPLACE PROCEDURE lihat_mahasiswa
IS
      vnama mahasiswa.nama%type;

BEGIN
SELECT nama INTO vnama FROM mahasiswa
WHERE nim = '30108002';
DBMS_OUTPUT.PUT_LINE('Nama mahasiswa dengan nim
      30108002 adalah '||vnama);
END;
/
```

Hasil setelah dieksekusi adalah sbb,

```
SQL> EXECUTE lihat_mahasiswa;
```

Nama mahasiswa dengan nim 30108002 adalah paimen

Procedure lihat_mahasiswa di atas bersifat statis, dieksekusi kapanpun memiliki output yang sama.

PROCEDURE DENGAN PARAMETER/ARGUMEN

Perbedaan dengan procedure tanpa parameter yaitu procedure dengan parameter memiliki output yang dinamis sesuai dengan nilai yang diberi pada parameter procedure tersebut. Default argumen pada Oracle adalah IN.

Jenis-Jenis parameter/argumen dari Procedure adalah : Parameter Masukan (Input)

Ditandai dengan atribut IN, dimana nilai dari parameter ini merupakan inputan untuk sebuah procedure.

Contoh:

```
CREATE OR REPLACE PROCEDURE lihat_mahasiswa
(vnim IN mahasiswa.nim%type)

IS
      vnama mahasiswa.nama%type;

BEGIN
SELECT nama INTO vnama FROM mahasiswa
WHERE nim = vnim;
DBMS_OUTPUT.PUT_LINE('Nama mahasiswa dengan nim
'||vnim||' adalah '||vnama);
```

```
END;
/
```

Parameter Keluaran (Output)

Ditandai dengan atribut OUT, dimana parameter ini merupakan variabel penampung untuk output sebuah procedure.

Contoh:
Procedure yang menggunakan parameter masukan dan juga parameter keluaran adalah sebagai berikut :

```
CREATE OR REPLACE PROCEDURE lihat_mahasiswa
(vnim IN mahasiswa.nim%type,
vnama OUT mahasiswa.nama%type)

IS BEGIN
SELECT nama INTO vnama FROM mahasiswa
WHERE nim = vnim;
DBMS_OUTPUT.PUT_LINE('Nama mahasiswa dengan nim
'||vnim||' adalah '||vnama);

END;
/
```

Untuk mengeksekusi procedure lihat_mahasiswa :

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  vnama mahasiswa.nama%type;
  3  BEGIN
  4        lihat_mahasiswa ('30108002',vnama);
  5        DBMS_OUTPUT.PUT_LINE('Nama mahasiswa teladan bulan ini adalah '||vnama);
  6  END;
  7  /
```

Nama mahasiswa teladan bulan ini adalah paimen

Parameter Masukan/Keluaran (Input/Output)

Ditandai atribut IN OUT, dimana parameter tersebut dianggap sebagai masukan kemudian diproses dan ditampilkan kembali sebagai keluaran.

Contoh:

```
CREATE OR REPLACE PROCEDURE k
uadrat (x IN OUT number)

IS
BEGIN
```

```
        x:=x*x;

END;
/
```

Untuk mengeksekusi procedure kuadrat :

```
SQL> SET VERIFY OFF;
SQL> DECLARE
  2  bil number := '&input_angka';
  3  n number;
  4  BEGIN
  5         n := bil;
  6         kuadrat(bil);
  7         DBMS_OUTPUT.PUT_LINE('kuadrat dari '||n||' adalah '||bil);
  8  END;
  9  /
Enter value for input_angka : 5
kuadrat dari 5 adalah 25
```

9.2 FUNCTION

Perbedaan mendasar antara function dan procedure adalah bahwa function harus mengembalikan nilai tertentu kepada pemanggilnya. Nilai ini dikembalikan dengan menggunakan sintaks RETURN.

Non Nested Function

Disebut juga stored function yang mirip dengan stored procedure tetapi harus memberikak output sebuah nilai. Function ini tersimpan dalam basis data.

Sintaks:

```
CREATE [OR REPLACE] FUNCTION nama_function
       [(argumen [IN|OUT|IN OUT] tipe_data,
       argumen [IN|OUT|IN OUT]tipe_data,
         ...)]
RETURN tipe_data {IS|AS}
       [deklarasi variabel lokal]

BEGIN
       badan fungsi
END;
```

RETURN adalah nilai yang dikembalikan oleh function. Jika terdapat

RETURN dalam badan fungsi, maka itu berfungsi untuk mengembalikan kontrol kepada pemanggil fungsi bersama nilai yang dikembalikan fungsi.

Sintaks:

```
RETURN ekspresi;
```

Dalam satu fungsi dimungkinkan penggunaan RETURN yang lebih dari satu tetapi bila di dalam badan fungsi tidak terdapat pernyataan RETURN maka akan terjadi error.

Contoh:

```
CREATE OR REPLACE FUNCTION tambah
(n1 number, n2 number)
RETURN number IS

BEGIN
      return (n1+n2);
END;
```

Cara mengeksekusi :

```
SQL > select tambah (1,8) from dual;

tambah (1,8)
------------
           9
```

Contoh :

```
CREATE OR REPLACE FUNCTION
cariMahasiswa (vnim mahasiswa.nim%type)
RETURN number IS
      v_nim mahasiswa.nim%type;

BEGIN
      SELECT nim FROM mahasiswa WHERE nim =
      vnim; IF SQL%FOUND THEN
            RETURN 1;
      END IF;
EXCEPTION
      WHEN NO_DATA_FOUND THEN
            RETURN 0;
```

```
END cariPegawai;
```

Contoh pemanggilan fungsi:

```
declare begin
   IF(cariMahasiswa('30108002') = 1) THEN;
      dbms_output.put_line('mahasiswa terdaftar');
   ELSE
      dbms_output.put_line('mahasiswa tidak
      terdaftar');
   END IF;

end;
```

Untuk meng-compile ulang sebuah function digunakan sintaks berikut

```
ALTER FUNCTION nama_fungsi COMPILE;
```

Sedangkan untuk menghapus funsi digunakan sintaks berikut

```
DROP FUNCTION nama_fungsi;
```

Nested Function

Seperti nested procedure, fungsi ini adalah fungsi yang dideklarasikan dalam suatu blok PL/SQL, tidak tersimpan dalam basis data dana hanya bisa dipanggil oleh blok dimana fungsi dideklarasikan.

Contoh:

```
DECLARE
      jum number;
      FUNCTION tambah(n1 number, n2 number)
      RETURN number IS
      BEGIN
            RETURN(n1+n2); END tambah;
BEGIN
      jum:=tambah(2,3);
      dbms_output.put_line('Hasil penjumlahan antara
         2 dan 3 adalah '||TO_CHAR(jum));
END;
```

JENIS-JENIS FUNCTION :

1. FUNCTION TANPA PARAMETER

Function ini tidak memiliki parameter/argumen biasanya bersifat statis setiap kali dieksekusi.
Contoh :

```
DECLARE

      FUNCTION jumMahasiswa
      RETURN number IS jum number;
      BEGIN
         select count(nim) into jum from mahasiswa;
         RETURN(jum); END jumMahasiswa;
BEGIN
      dbms_output.put_line('Jumlah mahasiswa saat ini adalah
         '||TO_CHAR(jumMahasiswa));
END;
```

2. FUNCTION BER-PARAMETER

Sama seperti Procedure berparameter, Function ber-parameter memiliki outputan yang dinamis sesuai dengan nilai yang diassign ke parameter pada function tersebut. Adapun parameter yang dimaksud disini adalah :

a. Parameter masukan

Contoh :

```
CREATE OR REPLACE FUNCTION
luas_persegi_panjang
(p number,l number)
RETURN number
IS

BEGIN
      RETURN (p*l);
END;
```

Salah satu cara untuk mengeksekusinya dengan menggunakan klausa SELECT seperti di bawah ini :

```
SQL> SELECT luas_persegi_panjang (8,3) FROM dual;

LUAS_PERSEGI_PANJANG(8,3)
-----------------------------------------
                24
```

b. Parameter keluaran

Contoh :

```
CREATE OR REPLACE FUNCTION volume_tabung
(r IN number, t IN number,luas OUT number)
RETURN number IS pi
   number:=3.14;
   vol number;

BEGIN luas:=pi*r*r;
   return (luas*t);

END;
```

Misal digunakan blok PL/SQL untuk menjalankan fungsi diatas :

```
declare
   L_alas number;

begin
   dbms_output.put_line('Volume tabung paijo
   adalah '||volume_tabung(2,5,L_alas));
   dbms_output.put_line('Luas alas lingkaran
    pada tabung paijo adalah '||L_alas);
end;
```

Output :

Volume tabung paijo adalah 62.8
Luas alas lingkaran pada tabung paijo adalah 12.56

Contoh di atas menunjukan bahwa function dapat mengembalikan lebih dari satu nilai lewat parameter OUT.

c. Parameter masukan/keluaran

Contoh :

```
CREATE OR REPLACE FUNCTION volume_tabung_ku
(x IN OUT number, t IN number)
RETURN number IS
        pi number:=3.14;
        vol number;

BEGIN
      x:=pi*x*x;
```

```
      vol:= x*t;
      return (vol);
END;
```

digunakan blok PL/SQL berikut untuk menjalankan fungsi diatas :

declare
y number:='&jari_alas';

```
begin
   dbms_output.put_line('Volume tabung
   paimen adalah '||volume_tabung2(y,5));
   dbms_output.put_line('Luas alas
   lingkaran tabung paimen adalah '||y);
end;
```

Output :

Enter value for jari_alas: 2
old 2: L number:='&jari_alas'; new 2: L number:='2';
Volume tabung paimen adalah 62.8
Luas alas lingkaran tabung paimen adalah 12.56

9.3 LATIHAN

1. Buat sebuah procedure untuk melakukan enkripsi sebuah string yang diinputkan oleh user. jika string yang dihasilkan tetap sama maka keluarkan perintah gagal enkripsi.

   contoh :

```
SQL> execute enkripsi('oh semoga nilaiku bagus');
======================
enkripsi berhasil
======================
string sebelum enkripsi : oh semoga nilaiku bagus
string sesudah enkripsi : 0h s3m0g4 n1l41ku b4gus

PL/SQL procedure successfully completed.

SQL> execute enkripsi('kyt kmnwxz zzbfr');
======================
enkripsi gagal
```

```
======================

PL/SQL procedure successfully completed.
```

2. Buatlah fungsi untuk menghasilkan nilai biner dari suatu angka desimal. Tuliskan juga cara mengeksekusinya.

   Contoh:

   ```
   SQL> select biner(10) from dual; BINER(10)
   ```

3. Jelaskan maksud dari procedure dibawah ini

```
PROCEDURE calc_bonus (emp_id IN INTEGER, bonus
OUT REAL) IS
   hire_date DATE;
   bonus_missing EXCEPTION;
BEGIN
   SELECT sal * 0.10, hiredate INTO bonus,
hire_date FROM emp
      WHERE empno = emp_id;
   IF bonus IS NULL THEN
      RAISE
   bonus_missing; END IF;
   IF MONTHS_BETWEEN(SYSDATE, hire_date) > 60 THEN
      bonus := bonus + 500;
   END IF;
   ...
EXCEPTION
   WHEN bonus_missing THEN
      ...
END calc_bonus;
```

4. Jalankanlah Package di bawah ini !

```
CREATE OR REPLACE PACKAGE lihat IS
   function is_kartu(id
   pembayaran.id_pembayaran%type)
   return number;
   procedure all_kartu;
END;

create or replace package body lihat
   is function is_kartu(id
   pembayaran.id_pembayaran%type) r
   eturn number is
   kartu
```

```
        pembayaran.pembayaran_kartu%type;
           tunai
        pembayaran.pembayaran_tunai%type;
           begin
                    select pembayaran.pembayaran_kartu,
                    pembayaran.pembayaran_tunai
                    into kartu, tunai
                    from pembayaran
                    where
              id=pembayaran.id_pembayaran;

                    if kartu>=tunai then return (1);
                    else return (0); end if;
              end;

              procedure all_kartu is cursor
                 c_pembayaran is
                       select *
                       from pembayaran
                       where is_kartu
                       (pembayaran.id_pembayaran)=1;


           nama pemesan.nama_pemesan%type;
           id pembayaran.id_pembayaran%type;
           begin
            dbms_output.put_line ('---
                 ----------------------
                 ----------------------
                 --------------------'); for x in c_pembayaran loop
                 id:=x.id_reservasi; select c.nama_pemesan
            into nama
                 from reservasi b, pemesan c
                 where id=b.id_reservasi and b.id_pemesan=c.id_peme san;
                 dbms_output.put_line (c_pembayaran%rowcount ||'.
                 '||x.id_pembayaran||' pada tanggal '||x.tgl_pembayaran||' oleh
                 '||nama);
            end loop;
      end;
end;

EXECUTE lihat.all_kartu;
```

Apa output dari package tersebut dan
Analisalah package tersebut !!

10. AKSES DATABASE ORACLE MELALUI PHP

10.1 SEKILAS TENTANG APLIKASI WEB MENGGUNAKAN PHP DAN ORACLE

Saat ini hampir semua aplikasi Web yang di kembangkan saat ini membutuhkan tekhnologi database untuk melakukan penyimpanan dan pengelolaan data yang di gunakan di dalam nya.PHP memberikan dukungan terhadap banyak database,baik yang bersifat komersial atau pun yang tidak,termasuk Oracle.

Di indonesia ,banyak terdapat perusahaan-perusahaan raksasa (seperti Pertamina,Telkom,PLN) ataupun perusahaan-perusahaan yang lain nya yang menggunakan database Oracle untuk menyimpan dan mengelola data mereka,misalnya,data pelanggan,anggaran akuntansi dan keuangan,dan sebagainya.dan ternyata untuk pengembangan aplikasi web nya

pun,mereka menggunakan PHP.Disini saya mencoba untuk berbagi konsep-konsep dasar yang di perlukan dalam mengembangkan aplikasi web menggunakan PHP dan Oracle.

10.2 INSTALASI ORACLE DATABASE XE

Memperoleh *software* Oracle *Database* XE

Software Oracle Database XE (Oracle Database 10g Express Edition),Oracle versi gratis dapat anda peroleh di situs *Oracle Technology Network*: *http://otn.oracle.com/xe*.

Melakukan Instalasi Oracle Database XE

Sebelum anda mempelajari Oracle,anda perlu memasang software Oracle Database XE terlebih dahulu,yang akan kita gunakan sebagai sarana untuk mempelajari Oracle dan sebagai server database pada saat mempelajari materi tentang akses database melalui PHP. Berikut cara-cara nya.

1. Buka direktori *Software/OracleXE*
2. Jalankan *file OracleXEUniv.exe.*
3. Klik *next* untuk beralih ke form berikutnya,pilih opsi ***I accept the terms in teh license agreement***,lalu klik *next*
4. Biarkan Oracle di instal pada lokasi default (**C:\OracleXE\).**Jika anda ingin menentukan direktori lain untuk tujuan instalasi,klik ***Browse*** dan tentukan direktori yang anda inginkan.lalu klik next untuk melanjutkan.
5. Tentukan password untuk ***user sys*** dan ***system*** lalu klik *next.*
6. Klik ***install*** untuk memulai instalasi.
7. Tunggu beberapa saat sampai semua proses selesai,pada tahap ini yang di lakukan adalah melakukan instalasi,membuat sekaligus menjalankan service Oracle dan melakukan konfigurasi database.
8. Hilangkan tanda cheklist pada opsi ***launch the database hompage***,lalu klik ***finish*** untuk meng akhiri proses instalasi.

Untuk catatan: Pada Oracle Database 10g Expres Edition,database secara otomatis akan di buat dengan nama XE.Dengan demikian kita tidak perlu lagi membuat database secara manual,kita hanya perlu membuat user nya saja.

Membuat User Baru ke dalam Database XE

Untuk keperluan pembelajaran ***Oracle Essential*** kita perlu membuat user khusus,tujuannya adalah agar tabel-tabel yang akan di buat nanti tidak tercampur dengan tabel-tabel milik user ***sys*** maupun ***system***,caranya adalah sebagai berikut:

1. Klik ***Start / Program / Oracle Database 10g Expres Edition.***
2. Pilih Go To Database Home page

3. Isikan ***SYSTEM*** untuk ***Username*** dan masukan password yang anda kehendaki pada saat proses instalasi,lalu klik ***Login***
4. Pilih ***Administration / Database User / Create User***
5. Isikan nama user (misal: Amien) dan tentukan password (dua kali) untuk user tersebut,untuk keperluan pembelajaran,beri tanda cheklist pada opsi ***DBA(*** pada bagian ***Roles*:)** jika sudah klik create untuk melakukan pembuatan user.
6. Tampak bahwa sekarang dalam database XE terdapat dua user (***Amien*** dan ***HR***)
7. ***selesai !*** klik ***logout*** untuk disconnect dari user SYSTEM.

10.3 INSTALASI DAN KONFIGURASI APACHE

Memperoleh *software Apache*

Apache adalah aplikasi web server yang akan kita gunkan,software versi terbaru dapat anda peroleh disitus resmi Apache: *http:// httpd.Apache/org/.*

Melakukan Instalasi Apache

Untuk melakukan instalasi ***Apache*** berikut caranya:

1. Buka direktori ***software\Apache***
2. Jalankan file ***httpd-2.2.16-win32-x86-openss1-0.9.80.msi.***
3. ***Klik*** untuk beralih ke ***form*** berikut nya.
4. Pilih opsi ***Iacept the terms in the license agreement***,lalu klik ***next***
5. Klik ***next*** untuk melanjutkan proses instalasi
6. Isikan Network Domain dan Server Name dengan ***Localhost***

   Dan Administrator Email Address dengan ***me@localhost*** lalu pilih opsi for ***All User***,lalu klik ***next***
7. Pilih ***Typical***,lalu klik ***next***
8. Klik ***Change*** untuk menentukan lokasi instalasi
9. Lalu tuliskan ***C:\Apache2.2*** pada kotak ***folder Name***,lalu klik ***OK***
10. Klik untuk ke form selanjutnya
11. Klik ***Install*** untuk memulai proses instalasi,tunggu beberapa saat sampai muncul form Berikut nya
12. Klik ***Finish*** untuk mengakhiri proses instalasi.

Melakukan Konfigurasi Apache

Agar software ***Apache*** dapat di gunakan sebagai ***web server*** dari program-progran ***PHP*** yang nanti akan kita buat,kita perlu melakukan sedikit konfigurasi terhadap ***Apache***,bagian ini akan membimbing anda untuk melakukan hal tersebut

1. Klik tombol ***Start > Progams > Apache HTTP Server 2.2 > Configure Apache Server***
2. Pilih ***Edit the Apache httpd.conf Configuation File***
3. Pastikan nilai untuk ***Document Root*** adalah sebgai berikut:

Document Root"C:/Apache2.2/htdocs"

Document Root berfungsi untuk menentukan direktori yang akan di gunakan sebagai tempat anda meletakan file-file kode PHP yang anda buat

4. Ubah nilai ***Directory Index*** menjadi seperti berikut:

   DirectoryIndex index.php index.html

5. Carilah bagian ***(baris 358),***tepatnya pada teks berikut:

   Addtype applicaton/x-compress .z

   Addtype application/x-gzip .gz .tgz

Tambahkan teks dibawah ini kedalamnya:

Addtype application/x-httpd-php .php
Addtype application/x-httpd-php .phtml
Addtype application/x-httpd-php .php3
Addtype application/x-httpd-php .html
Addtype application/x-httpd-php .htm
Addtype application/x-httpd-php-source .phps

6. Simpan perubahan yang telah anda buat di dalam ***file httpd.conf.***
7. ***Restart*** Apache menggunakan ***Start > Program > Apache HTTP Server 2.2 > Control Apache Server > Restart.***

Menjalankan Server Apache

Untuk menjalankan server ***Apache***,lakukan klik ganda(double) pada icon Apache yang berada pada bagian ***task bar***.pada saat muncul Apache Server monitor,***klik Start setelah klik Start bahwa indikasi server telah di aktifkan.***

Untuk memastikan server Apache sudah siap untuk di gunakan,coba anda jalankan web browser.lalu tuliskan: ***http//127.0.0.1/*** atau ***http//localhost/***. Jika konfigurasi benar anda akan memperoleh tampilan yang bertuliskan: **" it work! "**

10.4 INSTALASI DAN KONFIGURASI PHP

Software PHP berfungsi untuk menterjemahkan kode-kode PHP yang kita tulis menjadi suatu halaman web. Software PHP versi terbaru dapat anda peroleh di situs resmi PHP: *http//www.php.net/.*

Melakukan Instalasi PHP

Pada bagian ini anda akan mempraktekan cara melakukan instalasi PHP.

1. Matikan service Apache
2. Buka direktori ***Software\PHP***
3. jalankan file ***php-5.2.14-win32-installer.msi.***
4. klik ***next*** untuk form selanjutnya
5. Beri tanda ***cheklist*** pada opsi ***I accept the terms in the License aggrement***,lalu klik ***next***
6. Tentukan direktori tujuan instalasi,sebgai latihan isikan ***C:\PHP*** lalu klik ***next***
7. lalu pilih ***Apache 2.2.x Module***,lalu klik ***next***
8. Isikan ***C:\Apache2.2\conf***.direktori ini adalah tempat file ***httpd.conf*** berada,klik ***next*** untuk melanjutkan proses
9. Tentukan modul-modul PHP yang akan di instal pastikan pada bagian Extension anda memilih modul tambahan ***Oracle (10)*** dan *PDO| Oracle 10g Client and above.*
10. Klik ***next*** untuk beralih ke form selanjutnya
11. lalu ***klik install*** untuk memulai proses instalasi,tunggu beberapa saat sampai muncul form berikutnya
12. klik ***finish*** untuk mengakhiri proses instalasi PHP.

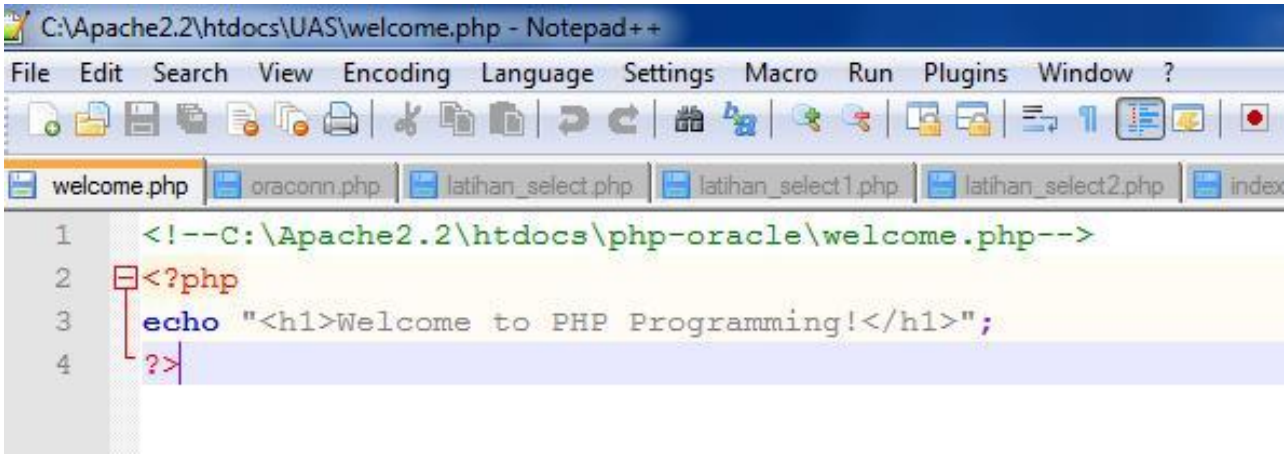
Melakukan Konfigurasi PHP agar Memiliki Dukungan terhadap Oracle

Untuk melakukan konfigurasi agar PHP memiliki dukungan terhadap Oracle kita harus memodifikasi(merubah) sedikit.Bagian ini akan menunjukan kepada anda untuk melakukan hal tersebut.

1. Jalankan aplikasi ***Text Editor***
2. Buka file ***php.ini*** yang ada di direktori ***C:\PHP***
3. Pada ***baris 342***,hilangkan tanda ***titik koma*** di depan text sehingga teks akan tampak seperi berikut:
    a. ***Eror_reporting = E_ALL & -E_NOTICE***
4. Pada ***baris ke 373***,ubah nilai ***display_errors*** menjadi ***On***, sehingga teks akan tampak seperti berikut:
    a. ***Display_errors = On***
5. Pada ***baris 496***,ubah nilai ***magic_quotes_gpc*** menjadi ***On***,sehingga teks akan tampak seperti ini:
    a. ***Magic_quotes_gpc = On***
6. Pada ***baris 535***,isikan nilai ***doc_root***,sehingga tampak seperti ini:
    a. ***Doc_root = "C:/Apache2.2/htdocs"*** ini adalah bagian terpenting.
7. Simpan perubahan yang anda lakukan pada file ***php.ini***
8. ***Copy*** file ***php5ts.dll*** dari ***C:\PHP*** ke ***C:\Apache2.2\bin.***
9. ***Restart*** komputer anda untuk memastikan semua konfigurasi telah terbarui.

Menguji Software PHP dengan membuat Contoh Aplikasi

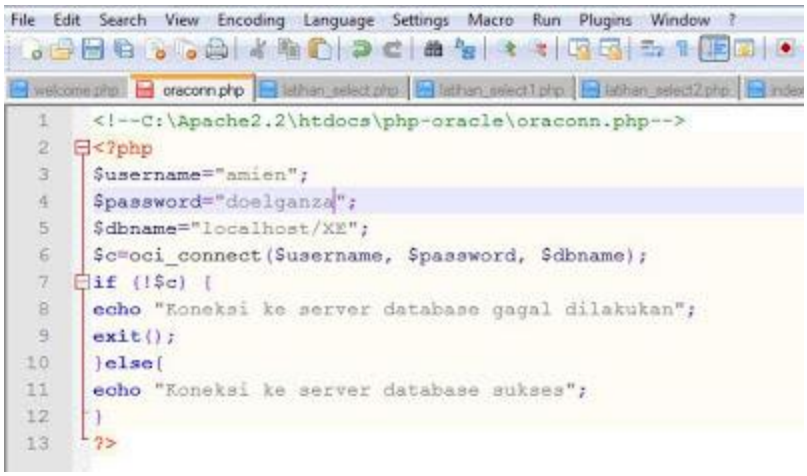
Untuk menguji apakah software PHP terpasang di komputer anda sudah benar apa belum,anda bisa mencoba membuat kode PHP sederhana.contoh nya sebagai berikut:
1.Jalankan aplikasi text editor (C++)
2.tuliskan kode program berikut ini



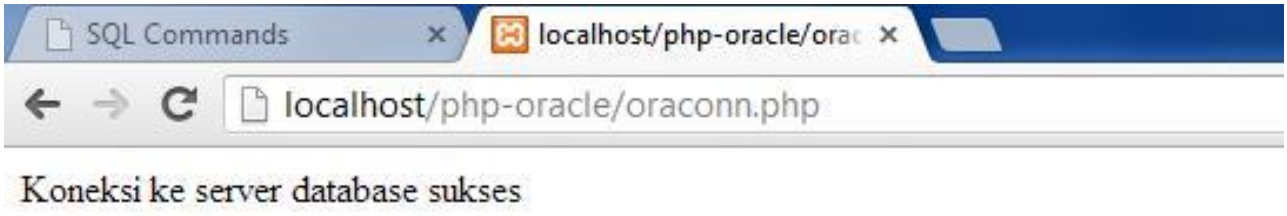
3.Simpan ***file*** dengan nama ***welcome.php*** di dalam direktori ***C:\Apache2.2\htdocs***
4.Jalankan ***web browser*** dan masukan alamat berikut ini: ***http//localhost/welcome.php***
5.Jika semua berjalan dengan benar,maka anda akan melihat tampilan sebagai berikut ini:

10.5 MEMBANGUN KONEKSI ANTARA PHP DAN ORACLE

Untuk membangun koneksi standar kita dapat membut suatu file khusus yang berisi kode sebagai berikut:

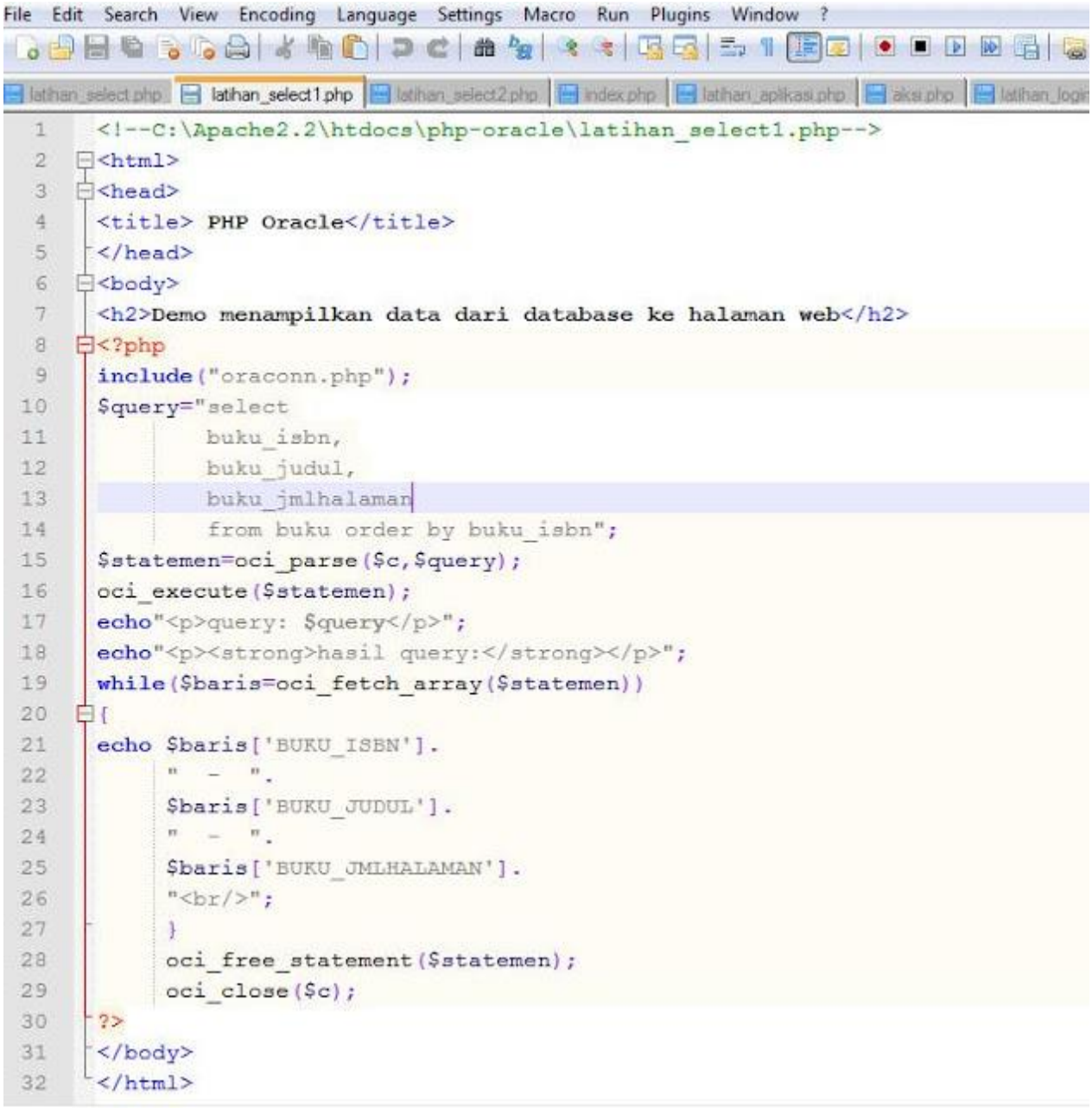


Simpan file tersebut dengan nama (misalkan) ***oraconn.php*** dan tempatkan di direktori kerja anda.Untuk menguji apakah database oracle dapat dibuat atau tidak,anda dapat memenggil file diatas secara langsung melalui web browser,dengan menuliskan URL http://localhost/php-oracle/oraconn.php.
Jika tidak ada masalah dan benar maka hasil yang akan di peroleh adalah sebgai berikut:

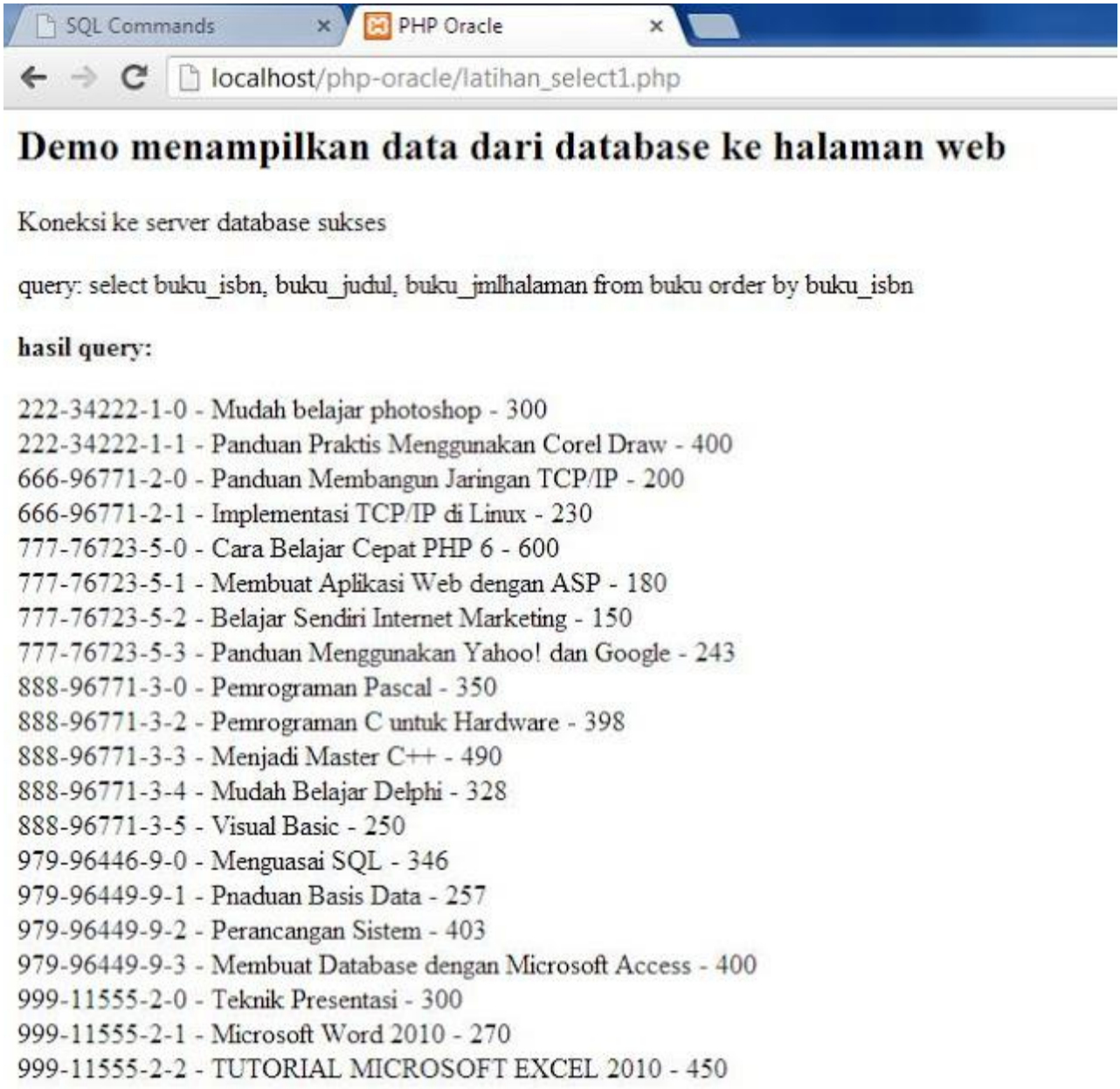


10.6 MENAMPILKAN DATA DARI DATABASE KE HALAMAN WEB

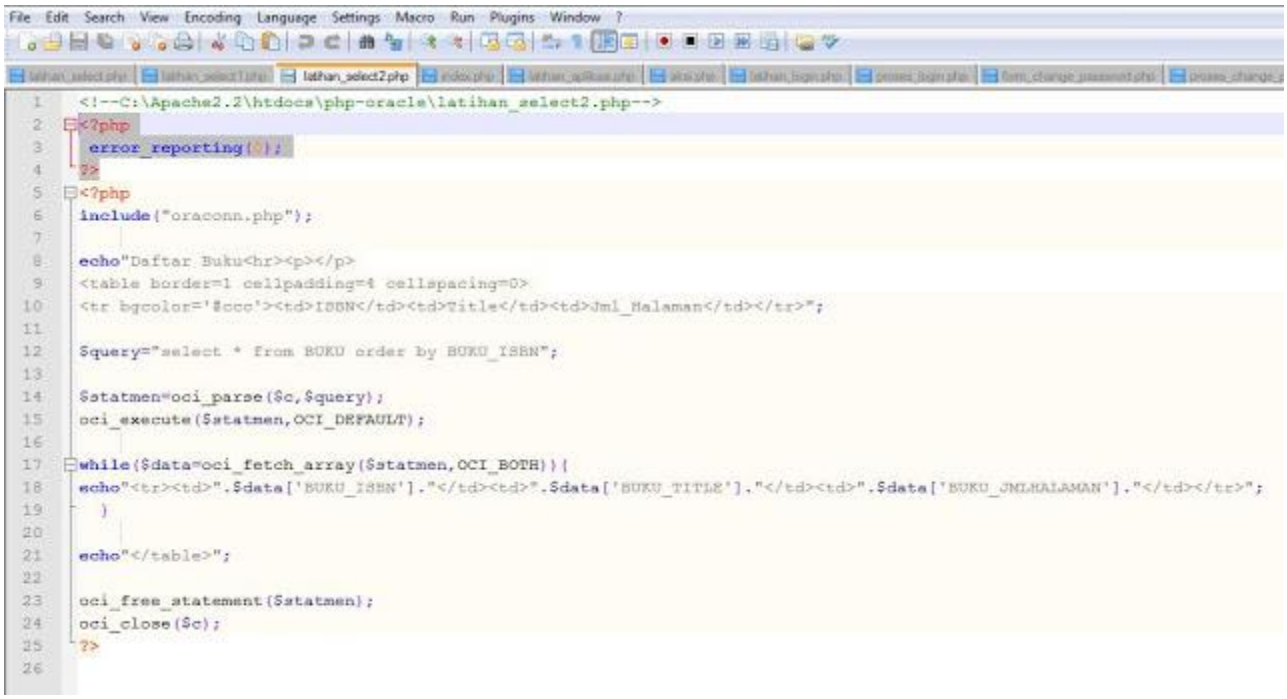
Setelah koneksi berhasil di bangun,langkah selanjutnya yang perlu di lakukan adalah mengakses data di dalam databse,apakah itu untuk di manipulasi atau hanya untuk di tampilkan saja ke dalam halaman web,pada bagian ini kita akan mempraktikan cara mengakses dari database dan menampilkannya ke dalam halaman web.
Contoh:
Berikut ini contoh kode program lengkap yang akan menunjukan konsep atau cara menampilkan data dari suatu tabel di dalam database:


File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
latihan_select.php
latihan_select1.php
latihan_select2.php
index.php
latihan_aplikasi.php
aksi.php
latihan_login
<!--C:\Apache2.2\htdocs\php-oracle\latihan_select1.php-->
<html>
<head>
<title> PHP Oracle</title>
</head>
<body>
<h2>Demo menampilkan data dari database ke halaman web</h2>
<?php
include("oraconn.php");
$query="select
buku_isbn,
buku_judul,
buku_jmlhalaman
from buku order by buku_isbn";
$statemen=oci_parse($c,$query);
oci_execute($statemen);
echo" query: $query ";
echo" hasil query: ";
while($baris=oci_fetch_array($statemen))
{
echo $baris['BUKU_ISBN'].
" - ".
$baris['BUKU_JUDUL'].
" - ".
$baris['BUKU_JMLHALAMAN'].
"
";
}
oci_free_statement($statemen);
oci_close($c);
?>
</body>
</html>

Dan hasil yang akan di berikan adalah sebagai berikut:



Contoh kode latihan *Daftar Buku* lengkap berikut nya sebgai berikut:



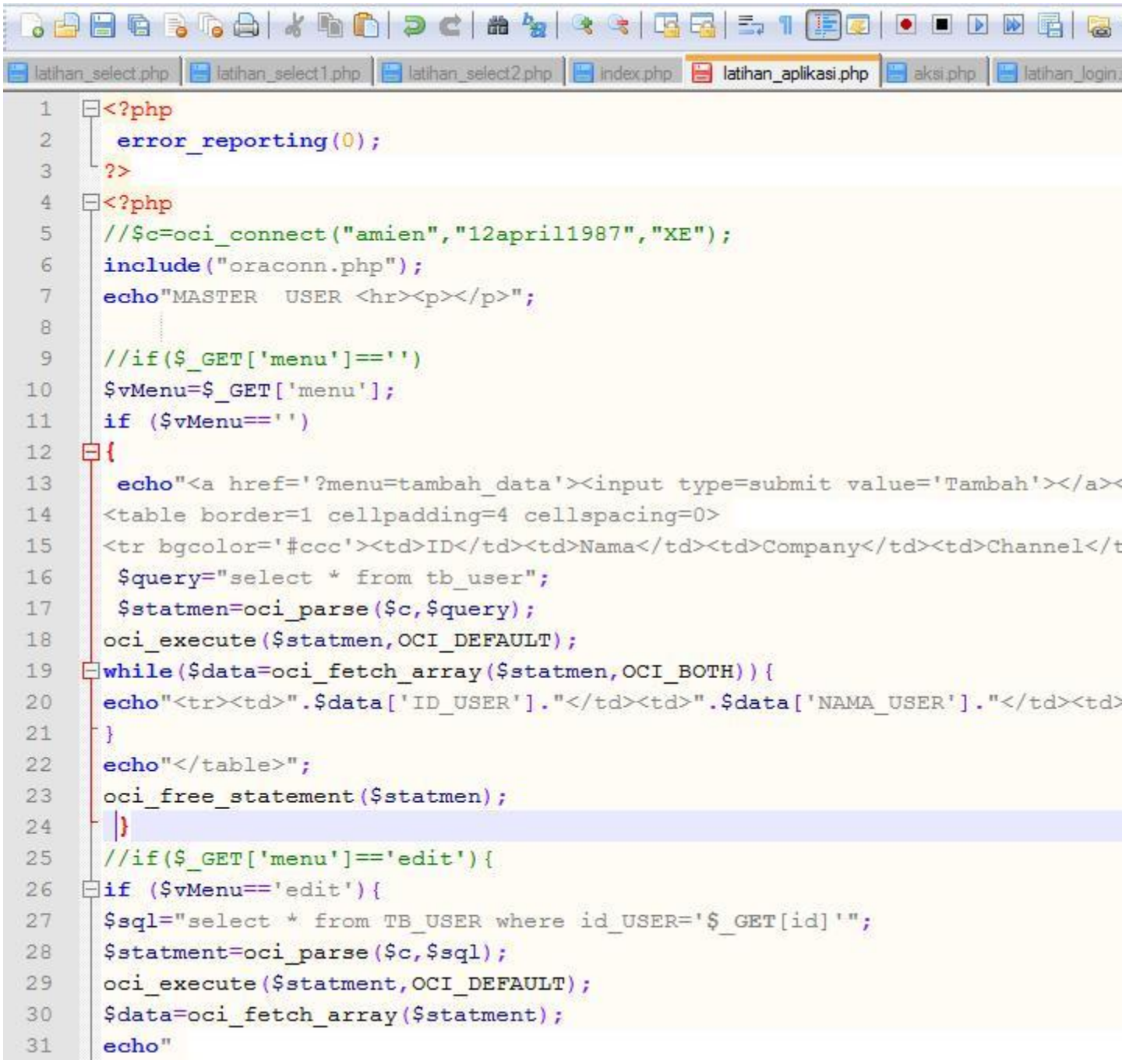
```php
<!--C:\Apache2.2\htdocs\php-oracle\latihan_select2.php-->
<?php
 error_reporting(0);
?>
<?php
include("oraconn.php");

echo"Daftar Buku<hr><p></p>
<table border=1 cellpadding=4 cellspacing=0>
<tr bgcolor='#ccc'><td>ISBN</td><td>Title</td><td>Jml_Halaman</td></tr>";

$query="select * from BUKU order by BUKU_ISBN";

$statmen=oci_parse($c,$query);
oci_execute($statmen,OCI_DEFAULT);

while($data=oci_fetch_array($statmen,OCI_BOTH)){
echo"<tr><td>".$data['BUKU_ISBN']."</td><td>".$data['BUKU_TITLE']."</td><td>".$data['BUKU_JMLHALAMAN']."</td></tr>";
  }

echo"</table>";

oci_free_statement($statmen);
oci_close($c);
?>
```

Dan hasil yang akan di berikan adalah sebagai berikut:

Koneksi ke server database suksesDaftar Buku

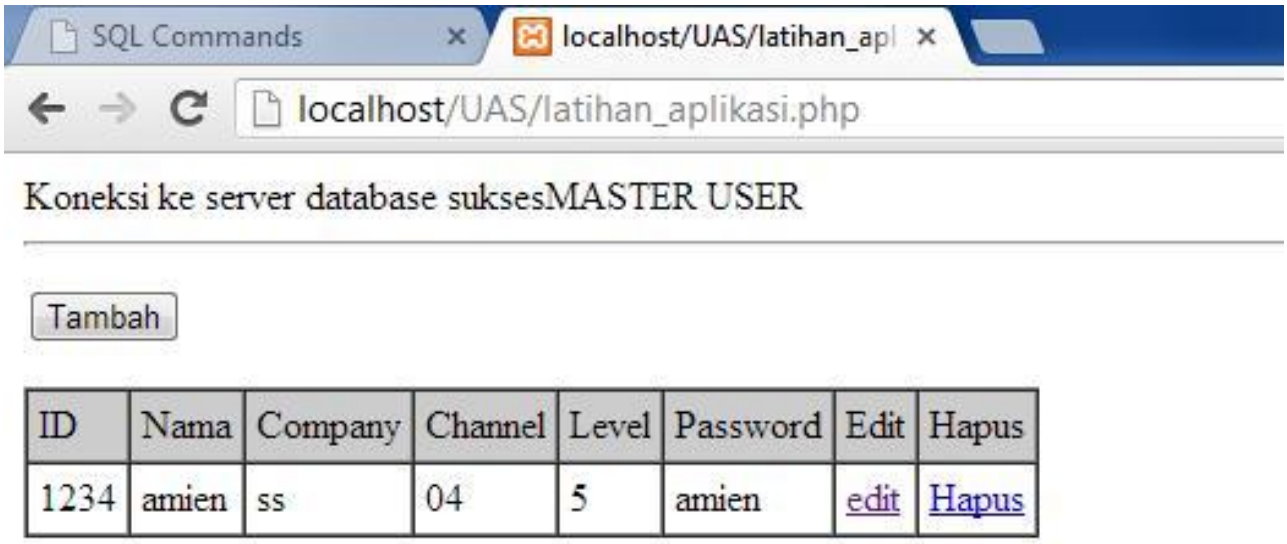
ISBN	Title	Jml_Halaman
222-34222-1-0		300
222-34222-1-1		400
666-96771-2-0		200
666-96771-2-1		230
777-76723-5-0		600
777-76723-5-1		180
777-76723-5-2		150
777-76723-5-3		243
888-96771-3-0		350
888-96771-3-2		398
888-96771-3-3		490
888-96771-3-4		328
888-96771-3-5		250
979-96446-9-0		346
979-96449-9-1		257
979-96449-9-2		403
979-96449-9-3		400
999-11555-2-0		300
999-11555-2-1		270
999-11555-2-2		450

Disini kita akan mencoba membuat latihan aplikasi dan kode lengkapnya sebagi berikut:

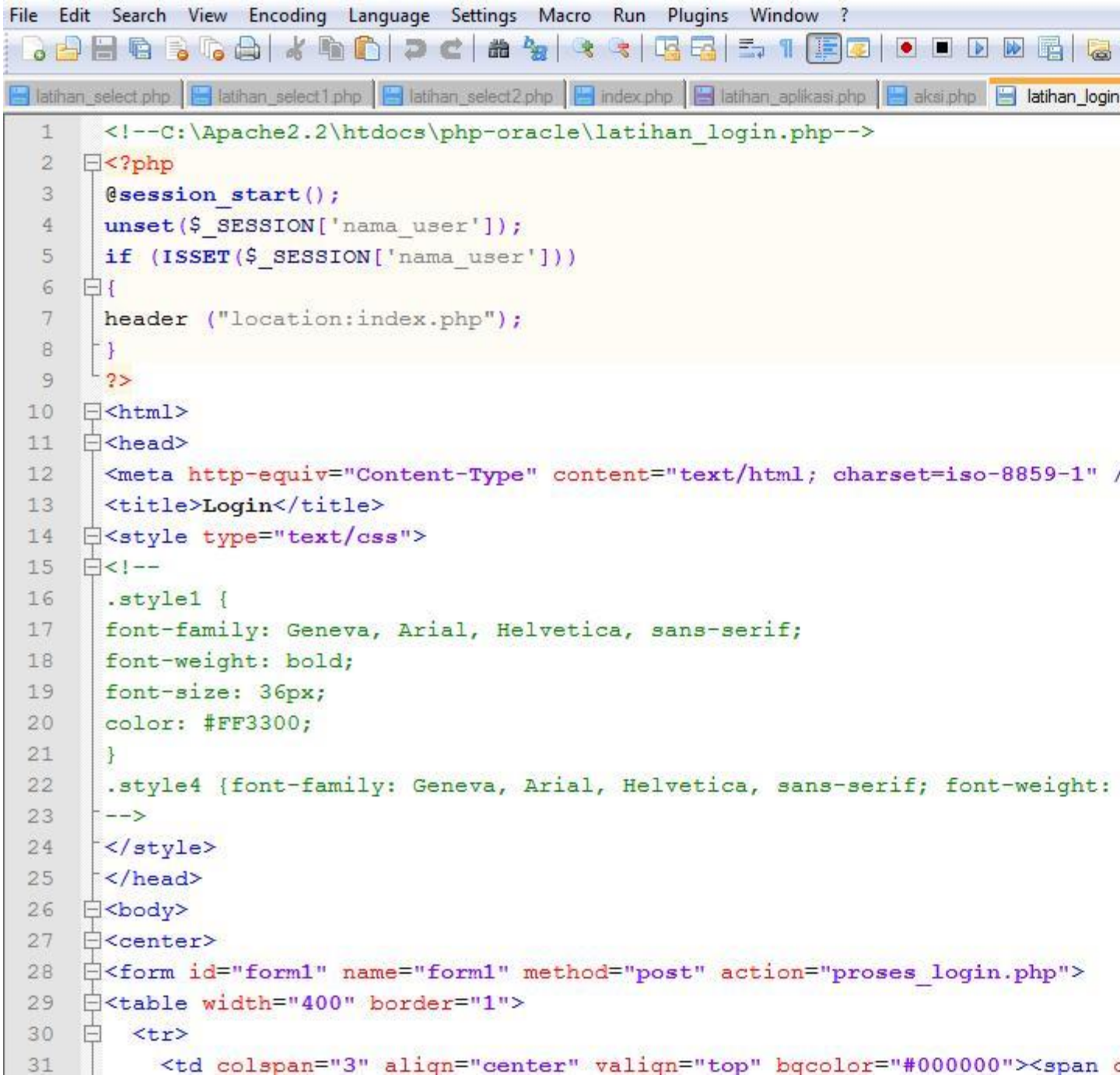


```
<form method=POST action='aksi.php?act=edit_data'>
<input type=hidden name='id_ubah' value='$data[ID_USER]'>
<table border=1 cellpadding=4 cellspacing=0>
<tr><td>ID</td><td><input type=text name='ID_UBAH' value='$data[ID_USER]'
<tr><td>Nama</td><td><input type=text name='NAMA_FUBAH' value='$data[NAMA
<tr><td>Company</td><td><input type=text name='COMPANY_FUBAH' value='$dat
<tr><td>Channel</td><td><input type=text name='CHANNEL_FUBAH' value='$dat
<tr><td>Level</td><td><input type=text name='LEVEL_FUBAH' value='$data[LE
<tr><td>Password</td><td><input type=text name='PASSWORD_FUBAH' value='$d
<tr><td></td><td><input type=submit value='Update'></td><tr>
</table>
</form>
";
}
 //if($_GET['menu']=='tambah_data'){
if ($vMenu=='tambah_data'){
echo"
<form method=POST action='aksi.php?act=tambah_data'>
<table border=1 cellpadding=4 cellspacing=0>
<tr><td>ID</td><td><input type=text name='ID_TMP'></td></tr>
<tr><td>NAMA</td><td><input type=text name='NAMA_TMP'></td></tr>
<tr><td>COMPANY</td><td><input type=text name='COMPANY_TMP'></td></tr>
<tr><td>CHANNEL</td><td><input type=text name='CHANNEL_TMP'></td></tr>
<tr><td>LEVEL</td><td><input type=text name='LEVEL_TMP'></td></tr>
<tr><td>PASSWORD</td><td><input type=text name='PASSWORD_TMP'></td></tr>
<tr><td></td><td><input type=submit value='SIMPAN'></td><tr>
</table>
</form>
";
}
?>
```

Dan hasil yang akan di berikan adalah sebagai berikut:



Kita akan mencoba aplikasi lain nya (misalkan) aplikasi Login,tapi ingat kita harus membuat database nya terlebih dahulu supaya bisa di panggil dan di tampilkan ke halam web.
Contoh membuat database nya:
Create table login (
"username" char (50) not null,
"password" char (40) not null,
primary
Berikut adalah contoh kode lengkap nya:

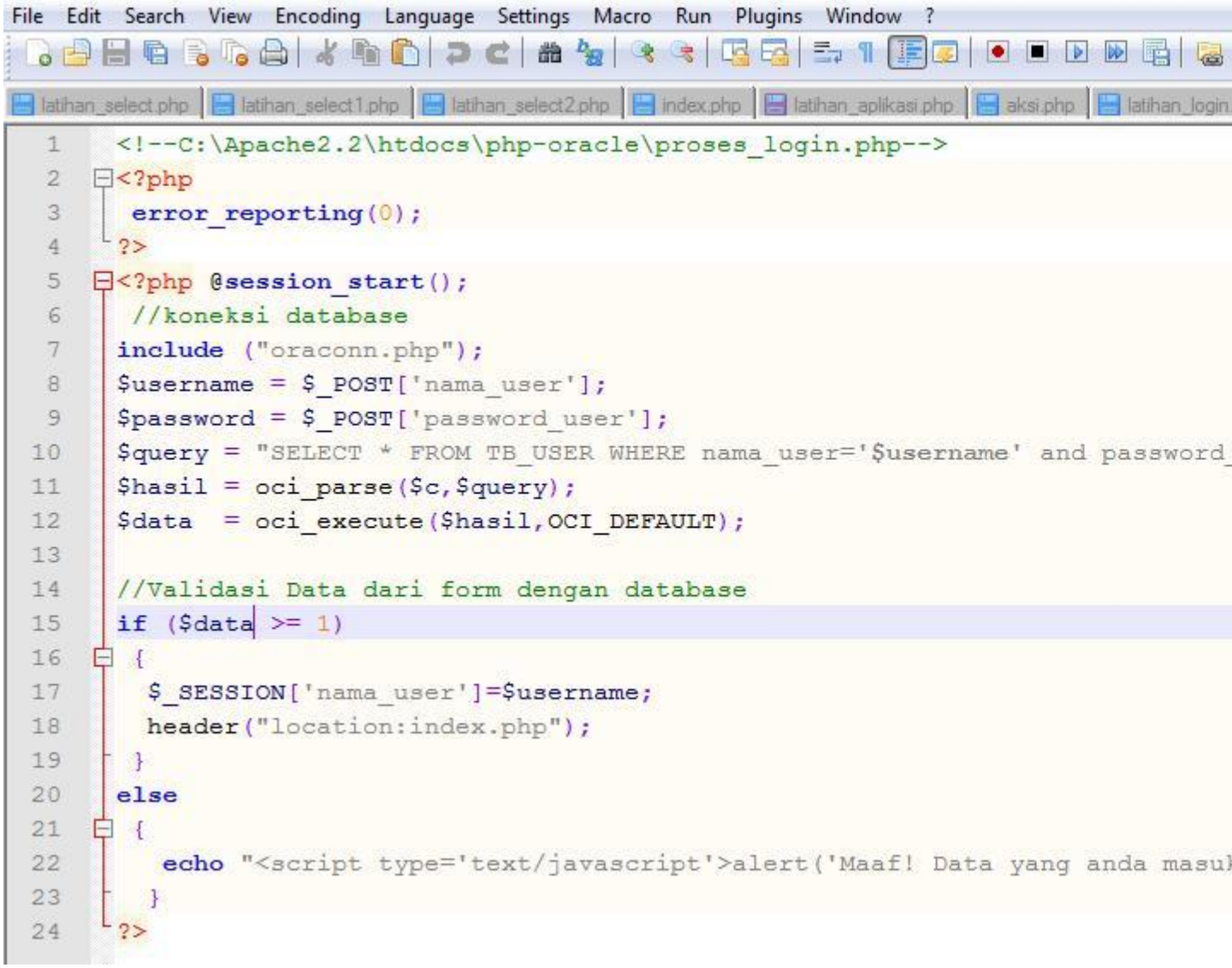

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
latihan_select.php
latihan_select1.php
latihan_select2.php
index.php
latihan_aplikasi.php
aksi.php
latihan_login
<!--C:\Apache2.2\htdocs\php-oracle\latihan_login.php-->
<?php
@session_start();
unset($_SESSION['nama_user']);
if (ISSET($_SESSION['nama_user']))
{
header ("location:index.php");
}
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
<title>Login</title>
<style type="text/css">
<!--
.style1 {
font-family: Geneva, Arial, Helvetica, sans-serif;
font-weight: bold;
font-size: 36px;
color: #FF3300;
}
.style4 {font-family: Geneva, Arial, Helvetica, sans-serif; font-weight:
-->
</style>
</head>
<body>

<form id="form1" name="form1" method="post" action="proses_login.php">
<table width="400" border="1">
<tr>
<td colspan="3" align="center" valign="top" bgcolor="#000000"><span</td></tr></table></form> </body></html>

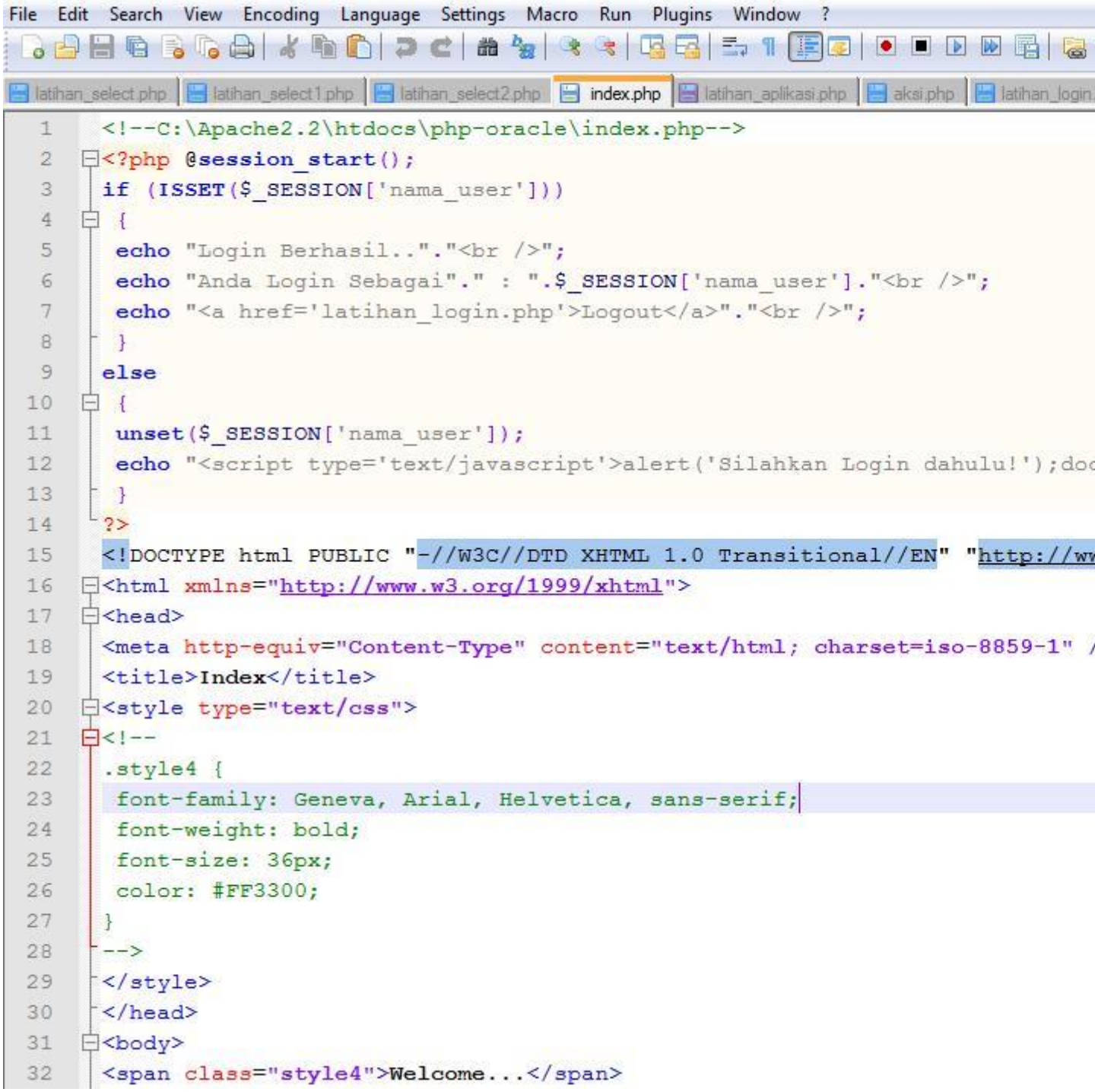
```
32      </tr>
33      <tr>
34        <td width="100"><span class="style4">Username</span></td>
35        <td width="3"><span class="style4">:</span></td>
36        <td width="275"><input name="nama_user" type="text" id="nama_user" />
37      </tr>
38      <tr>
39        <td><span class="style4">Password</span></td>
40        <td><span class="style4">:</span></td>
41        <td><input name="password_user" type="password" id="password_user" />
42      </tr>
43      <tr>
44        <td colspan="3" align="right"><input type="submit" name="Submit" valu
45      </tr>
46  </table>
47  </form>
48  </center>
49  </body>
50  </html>
51
```

Dan hasil yang akan di berikan adalah sebgai berikut:

Disini kita harus mengisi form username dan password nya terlebih dahulu supaya bisa masuk.
Dan kode untuk proses login nya adalah sebgai berikut:

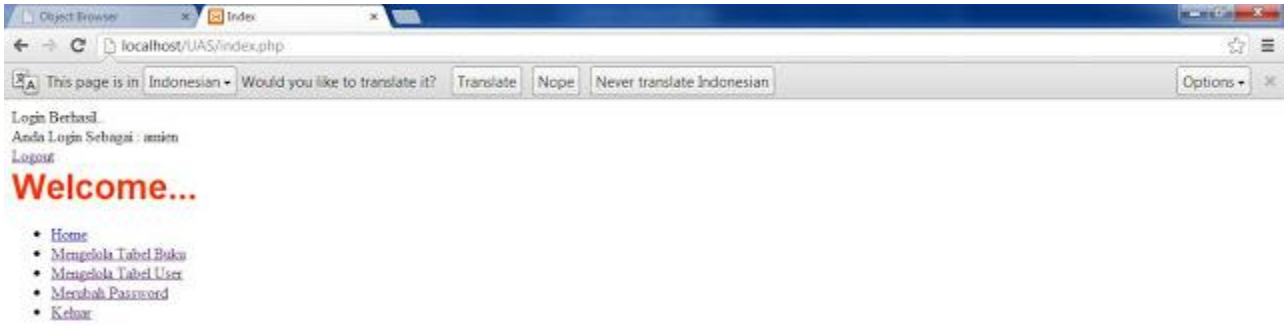


Disini kita harus menyertakan kode *Index* supaya memudahkan dalam melakukan pemnggilan *proses login* nya
Dan kode lengkap nya sebgai berikut:

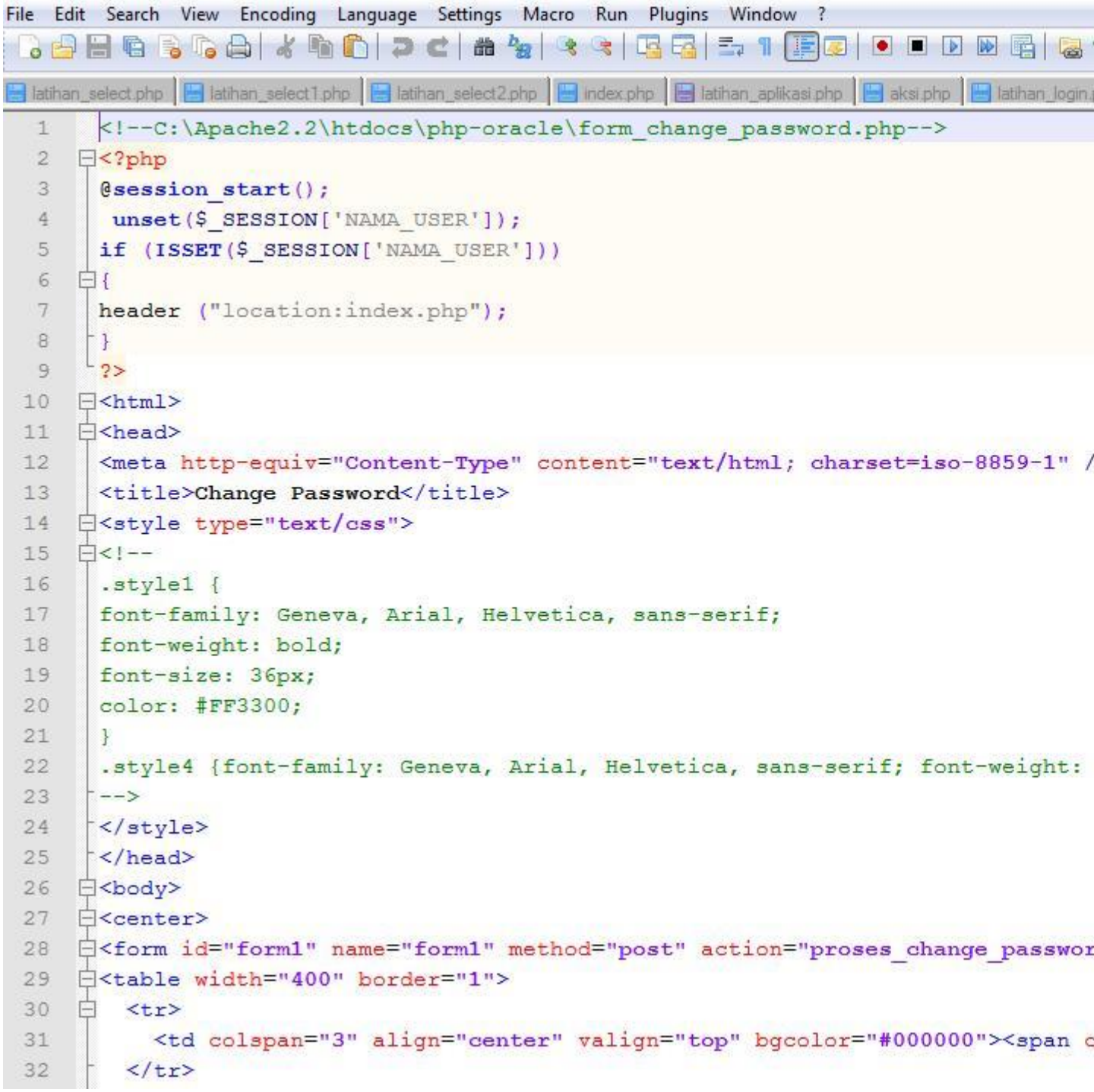

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
latihan_select.php
latihan_select1.php
latihan_select2.php
index.php
latihan_aplikasi.php
aksi.php
latihan_login
<!--C:\Apache2.2\htdocs\php-oracle\index.php-->
<?php @session_start();
if (ISSET($_SESSION['nama_user']))
{
echo "Login Berhasil..."."
";
echo "Anda Login Sebagai"." : ".$_SESSION['nama_user']."
";
echo "Logout"."
";
}
else
{
unset($_SESSION['nama_user']);
echo "<script type='text/javascript'>alert('Silahkan Login dahulu!');doc
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://ww
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
<title>Index</title>
<style type="text/css">
<!--
.style4 {
font-family: Geneva, Arial, Helvetica, sans-serif;
font-weight: bold;
font-size: 36px;
color: #FF3300;
}
-->
</style>
</head>
<body>
Welcome...

```
33  </body>
34  <ul>
35          <li class="current_page_item"><a href="#">Home</a></li>
36          <li><a href="latihan_select2.php">Mengelola Tabel Buku</a></l
37          <li><a href="latihan_aplikasi.php">Mengelola Tabel User</a></
38          <li><a href="form_change_password.php">Merubah Password</a></
39          <li><a href="latihan_login.php">Keluar</a></li>
40          </ul>
41  </html>
```

Dan hasil yang akan di berikan adalh sebagi berikut:



Kita juga akan mencoba membuat aplikasi yang bisa merubah *password*,sebelum nya kita harus membuat form nya terlebih dahulu dan kode lengkap nya sebgai berikut:

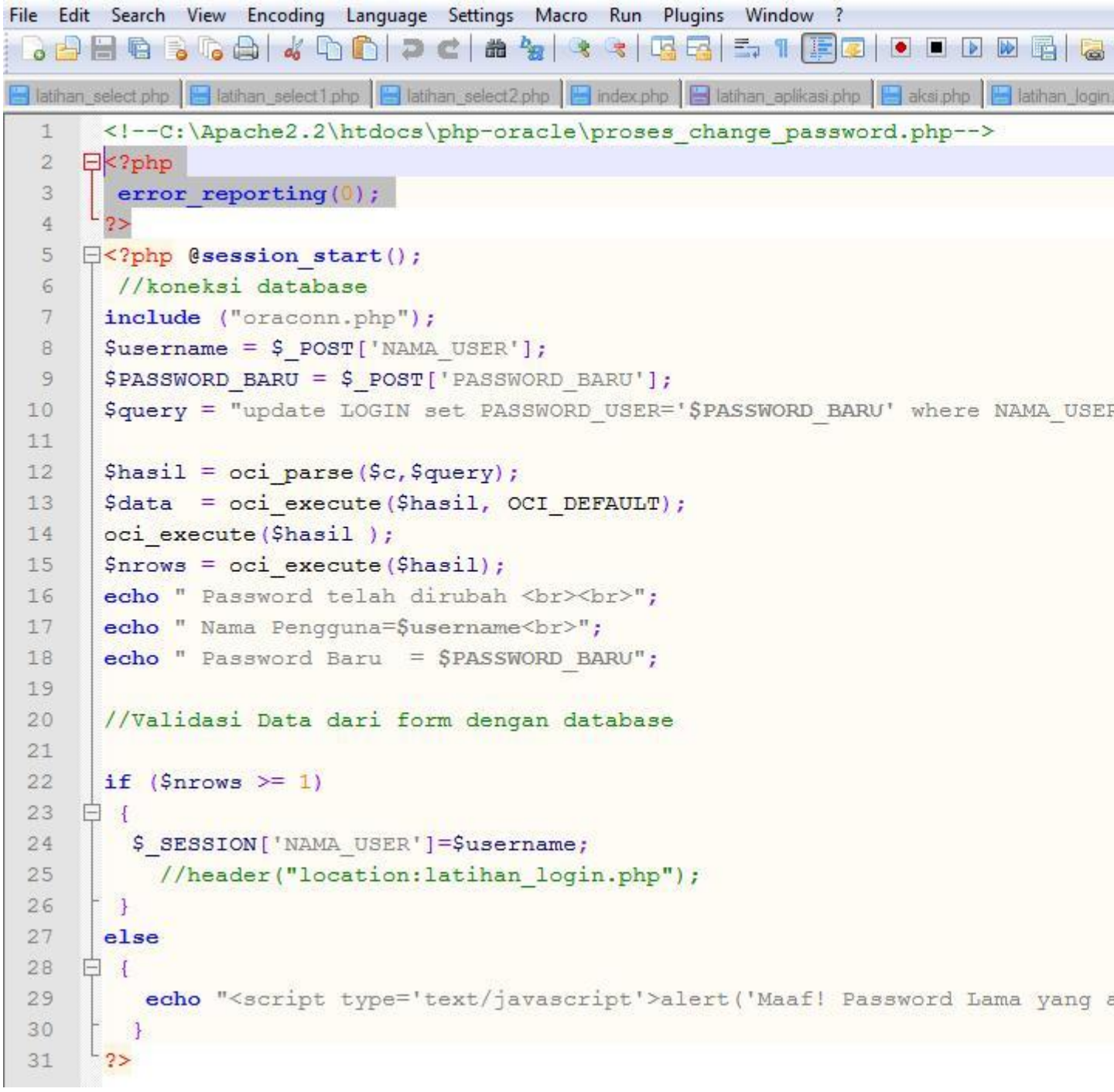

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
latihan_select.php
latihan_select1.php
latihan_select2.php
index.php
latihan_aplikasi.php
aksi.php
latihan_login.
<!--C:\Apache2.2\htdocs\php-oracle\form_change_password.php-->
<?php
@session_start();
unset($_SESSION['NAMA_USER']);
if (ISSET($_SESSION['NAMA_USER']))
{
header ("location:index.php");
}
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
<title>Change Password</title>
<style type="text/css">
<!--
.style1 {
font-family: Geneva, Arial, Helvetica, sans-serif;
font-weight: bold;
font-size: 36px;
color: #FF3300;
}
.style4 {font-family: Geneva, Arial, Helvetica, sans-serif; font-weight:
-->
</style>
</head>
<body>

<form id="form1" name="form1" method="post" action="proses_change_passwor
<table width="400" border="1">
<tr>
<td colspan="3" align="center" valign="top" bgcolor="#000000"><span c
</tr>

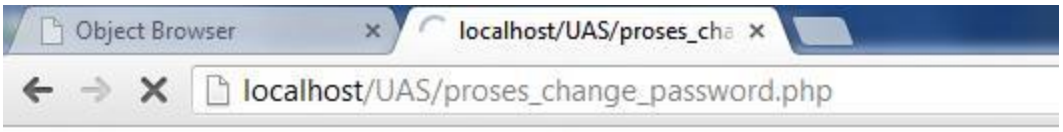
```
33      <tr>
34        <td width="150"><span class="style4">Nama user </span></td>
35        <td width="3"><span class="style4">:</span></td>
36        <td width="275"><input name="NAMA_USER" type="text" id="NAMA_USER" va
37      </tr>
38      <tr>
39        <td><span class="style4">Password Lama</span></td>
40        <td><span class="style4">:</span></td>
41        <td><input name="PASSWORD_LAMA" type="password" id="PASSWORD_LAMA" />
42      </tr>
43      <tr>
44        <td><span class="style4">Password Baru</span></td>
45        <td><span class="style4">:</span></td>
46        <td><input name="PASSWORD_BARU" type="password" id="PASSWORD_BARU" />
47      </tr>
48      <tr>
49        <td colspan="3" align="right"><input type="submit" name="Submit" valu
50      </tr>
51  </table>
52  </form>
53  </center>
54  </body>
55  </html>
```

Dan hasil yang akan diberikan sebagi berikut:

Setelah membuat form nya selesai, pasti kita membutuhkan proses untuk masuk kedalam nya supaya bisa merubah password nya,dan untuk kode lengkap untuk proses change password nya adalah sebgai berikut:



Ingat kita harus mengisi form nya terlebih dahulu,supaya bisa merubah password lama ke password baru,dan hasil yang akan di berikan adalah sebagi berikut:



Koneksi ke server database sukses Password telah dirubah

Nama Pengguna=amien
Password Baru = amien

DAFTAR ISI

1. Meloni, Julie C. (2008). *Teach Yourself PHP. MySQL and Apache All in One*. Sams Publishing. Indianapolis. ISBN: 979-0-672-32976-0
2. Rahmat Wijaya, Dedy (2009). *Praktikum Sistem Manajemen Basis Data*. Politeknik Telkom, Bandung.
3. Raharjo, Budi. (2010). *Belajar Otodidak Pemrograman Web dengan PHP + Oracle*. Informatika, Bandung.
4. http://amien782.blogspot.com/2013/06/tutorial-pemrograman-web-dengan-php.html