3. MEMBUAT VIEW

Objektif :

Setelah menyelesaikan pelajaran ini, diharapkan dapat melakukan hal-hal sebagai berikut:

1. Menjelaskan apa itu *View*
2. Membuat *View*
3. Mencari data melalui *View*
4. Merubah struktur *View*
5. Melakukan *Insert*, *Update* dan *Delete* Data melalui *View*
6. Menghapus *View*

3.1. Obyek Database

Objek-objek database adalah sebagai berikut :

Obyek	**Deskripsi**
Table	Unit paling dasar dari Database, yang berisi baris dan kolom
View	**Merupakan sebuah tabel logika, dan merupakan bagian data dari satu atau lebih tabel asal**
Sequence	Menghasilkan nilai primary key
Index	Meningkatkan kemampuan dari beberapa query
Synonym	Nama alternatif untuk Obyek

3.2. View

View adalah Tabel logika yang berdasarkan pada sebuah Tabel atau dari *View* lainnya. Sebuah *View* tidak mempunyai data dari *View* itu sendiri. Tabel yang ada pada *View* dinamakan "Tabel Asal". *View* disimpan seperti Perintah SELECT pada data dictionary.

3.3. Keuntungan View Keuntungan

View adalah :

- *View* dapat mengamankan data, karena *View* bisa menampilkan kolom-kolom tertentu dari Tabel Asal.
- *View* memperbolehkan user untuk membuat query sederhana untuk mendapatkan hasil dari query yang rumit. Sebagai contoh, *View* membolehkan user untuk mendapatkan informasi dari banyak Table tanpa harus mengetahui bagaimana untuk membuat perintah Join.
- *View* menyediakan data independen bagi user sementara dan program aplikasi. Sebuah *View* dapat digunakan untuk mendapatkan data dari beberapa Tabel.
- View menyediakan Grup User untuk mengakses data menurut kriteria mereka masingmasing.

3.4. Simple View Vs Complex View

View dibagi menjadi dua; *Simple* dan *Complex.* Perbedaan mendasar adalah pada operasi DML (*insert,update* dan *delete*) :

- *Simple View* :

  Data turunan yang berasal dari hanya satu Tabel

  Tidak berisi fungsi atau grup data

Dapat menampilkan DML melalui *View*

- *Complex View* :

  Data turunan yang berasal dari banyak Tabel

  Berisi fungsi atau grup data

  Tidak selalu dapat menampilkan DML melalui *View*

3.5. Membuat View

Syntax :

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
    [ (alias[, alias]…)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constrain]]
[WITH READ ONLY];
```

Pada syntax :

OR REPLACE	Membuat kembali sebuah View bila view itu sudah ada sebelumnya
FORCE	Membuat View walaupun Tabel Asalnya tidak ada
NOFORCE	Mebuat View hanya jika Tabel Asalnya ada
View	Nama dari View
Alias	Nama field-field yang akan di dipilih pada Tabel Asal (Namanya boleh tidak sama dengan Tabel Asal tapi jumlahnya harus sama dengan Tabel Asal)
Subquery	Perintah SELECT

WITH CHECK OPTION Menjelaskan bahwa hanya baris yang ada pada View yang bisa dilakukan operasi Insert dan Update *constraint* Nama yang ada pada CHECK OPTION constraint

WITH READ ONLY Memastikan bahwa tidak ada operasi DML yang dapat dilakukan pada View ini.

Petunjuk dalam membuat View

- *Subquery* yang didefinisikan pada *View* dapat berisikan SELECT yang kompleks, termasuk Join, Group, dan Subquery.
- *Subquery* yang didefinisikan pada *View* tidak dapat berisikan klausa ORDER BY. ORDER BY dapat digunakan pada saat kita mencari data dari VIEW tersebut.
- Bila tidak mencantumkan nama constraint untuk *View* yang kita buat dengan CHECK OPTION, maka system akan otomatis memberikan nama dalam format SYS_Cn.
- Dapat digunakan pilihan OR REPLACE untuk mengubah definisi dari *View*

Dapat mengontrol nama kolom dengan mengikut sertakan nama alias dari kolom pada subquery.

Contoh :

```
CREATE VIEW salvu30 AS SELECT empno EMPLOYEE_NUMBER, ename NAME, sal SALARY FROM emp
WHERE deptno = 30;
```

Lihat video

3.6 . Mendapatkan Data dari View

Dapat diperoleh data dari *View* seperti didapatkan dari Tabel asal. Dapat pula menampilkan isi dari seluruh *View* atau *View* dari baris atau kolom tertentu.

Contoh :

```
SELECT *
FROM salvu30
```

3.7. View pada Data Dictionary

Sekali membuat sebuah View maka dapat dicari melalui Tabel Data Dictionary yang bernama USER_VIEWS untuk melihat nama dari View dan definisi dari View. Text dari perintah SELECT mengatur View yang ada pada kolom LONG.

3.8. Akses Data menggunakan View

Ketika mengakses data, menggunakan View, Server Oracle menampilkan operasi sebagai berikut :

1. Mendapatkan definisi *View* dari Tabel Data Dictionary USER_VIEWS
2. Mencek hak akses untuk *View* dari Tabel Asal.
3. Mengkonversi *query View* ke dalam operasi yang ekuivalen pada satu atau lebih Tabel Asal yang dirujuk. Dengan kata lain, data didapatkan dari atau perubahan data diperoleh dari Tabel Asal.

3.9. Memodifikasi / Mengubah View

Option OR REPLACE memungkinkan View dapat diperbaharui walaupun namanya sama, jadi perbaruan yang terjadi menggantikan struktur yang lama. Ini berarti perubahan *View* dapat dilakukan tanpa harus men-*drop View*, membuat ulang, ataupun memberi hak kembali pada obyek Database.

Catatan : Ketika membuat kolom alias pada klausa CREATE VIEW, perhatikan bahwa alias tersebut juga harus ada pada subquery.

Contoh :

```
CREATE OR REPLACE VIEW      salvu30 (employee_number,
                                     employee_name, job_title)
        AS SELECT           empno, ename, job
          FROM              emp
         WHERE              deptno = 30;
```

Lihat Video

3.10. Membuat Complex View

Contoh :

```
CREATE VIEW  dept_sum_vu (name, minsal, maxsal, avgsal)
    AS SELECT          d.dname, MIN(e.sal), MAX(e.sal),
                       AVG(e.sal)
      FROM             emp e, dept d
      WHERE            e.deptno = d.deptno
```

Lihat Video

Contoh di atas adalah contoh dari pembuatan *View* yang kompleks dimana *View* tersebut menampilkan fungsi arimatika dimana *View* tersebut adalah *View* yang berisikan nama departemen, gaji minimum, gaji maksimum dan gaji rata-rata dari departemen tersebut. Perhatikan bahwa nama kolom alternatif telah dispesifikasikan pada *View* tersebut. Hal ini diperlukan bila kolom pada View tersebut diturunkan dari kolom fungsi atau sebuah ekspresi. Dapat dilihat struktur dari *View* tersebut dengan menggunakan perintah DESCRIBE pada SQL*Plus. Dan bila ingin menampilkan isi dari *View* dapat dilakukan dengan perintah SELECT.

3.11. Melakukan Operasi DML pada View

Dapat dilakukan operasi DML pada data melalui *View* bila operasi tersebut mengikuti aturan-aturan yang sudah ditetapkan. Pengguna dapat menghapus baris dari *View* bila setidaknya baris tersebut berisi seperti dibawah ini :

- Group Fungsi
- Klausa GROUP BY
- Keyword pseudocolumn ROWNUM

3.12. Menggunakan klausa WITH CHECK OPTION

Dapat dimungkinkan untuk melakukan refrential integrity checks melalui *View*. Dapat pula memberikan constraint pada setiap level Database. *View* dapat digunakan untuk mempertahankan integritas data, tapi kegunaannya sangatlah terbatas.

Klausa WITH CHECK OPTION menjelaskan bahwa INSERT dan UPDATE melalui *View* tidak dapat dibenarkan bila baris yang kita isikan atau kita *update* bukan merupakan bagian dari *View* yang kita buat.

Contoh :

```
UPDATE    salvu30    SET    deptno=10    WHERE    empno=7788;
```

Lihat Video

Catatan : Tidak ada baris yang di*update* karena bila nomor departemen diubah menjadi 10, *View* tidak lagi dapat melihat pegawai itu lagi. Oleh karena itu dengan klausa WITH CHECK OPTION, *View* hanya dapat melihat pegawai dengan nomor departemen 30 dan tidak diperkenankan merubah nomor departemen para pegawai tersebut melalui *View*.

3.13. Menolak Operasi DML

Dapat dipastikan bahwa tidak ada operasi DML yang dapat dijalankan pada *View* dengan membuat option WITH READ ONLY. Sebagai contoh :

```
                                                                    emp
CREATE OR REPLACE VIEW      salvu30          AS SELEC Tempno
EMPLOYEE_NUMBER, ename NAME, sal SALARY       FROM




WHERE                       deptno = 30

            WITH READ ONLY ;
```

Lihat Video

Dapat digunakan perintah DROP VIEW untuk menghapus *View*. Perintah tersebut akan menghapus definisi *View* dari Database. Menghapus *View* tidak menimbulkan efek pada Tabel Asal. *View-view* atau aplikasi lainnya yang merujuk pada *View* yang telah dihapus akan menjadi invalid atau tidak dapat digunakan lagi. Hanya pembuat dan user yang mendapat hak untuk menghapus *View* yang diperkenankan untuk menghapus *View*.

Syntaxnya :

```
DROP VIEW  nama_view;
```

Lihat Video

3.14. Inline View

Inline *View* pada klausa FROM dari perintah SELECT mendefinisikan sumber data untuk perintah SELECT. Pada contoh di bawah ini Inline *View* b menunjukkan keterangan seluruh nomor departemen dan gaji maksimum untuk tiap-tiap departemen pada Tabel EMP. Klausa WHERE a.deptno b.deptno AND a.sal < b.maxsal pada query utama menampilkan nama pegawai, gaji, nomor departemen dan gaji maksimum untuk seluruh pegawai yang mendapatkan gaji kurang dari gaji maksimum pada departemen mereka masing-masing.

Contoh :

```
SELECT      a.name, a.sal, a.deptno, b.maxsal      FROM      emp a, (SELECT      deptno,
max(sal) maxsal                         FROM      emp
```

```
GROUP BY deptno) b WHERE a.deptno = b.deptno AND a.sal < b.maxsal;
```

Lihat Video

Query "Top-N" sangatlah berguna untuk menampilkan query dimana hanya record-record *npaling atas atau n-paling bawah* pada tabel (tergantung pada kondisi yang diinginkan). Hasilnya dapat digunakan untuk analisa selanjutnya. Sebagai contoh, dengan menggunakan Analisa "TopN" dapat menampilkan tipe *query* sebagai berikut :

- The four most recent recruits in the company (Empat orang pegawai yang paling baru direkrut oleh perusahaan)
- The Top two sales representatives who have sold the maximum number of products (Dua orang sales yang paling banyak menjual produk)

Perintah "Top-N" **:**

```
SELECT [column_list], ROWNUM
FROM (SELECT [column_list] FROM table
ORDER BY Top-N_column)
WHERE ROWNUM <= N ;
```

Query "Top-N" digunakan pada perintah *query* bersarang yang konsisten dengan bagian-bagian yang dijelaskan di bawah ini :

- *Subquery* atau sebuah Inline *View* di gunakan untuk menyortir data. *Subquery* atau Inline *View* tersebut meliputi klausa ORDER BY untuk memastikan pengelompokan sudah sesuai dengan yang diinginkan. Untuk mendapatkan hasil yang ingin menampilkan nilai terbesar, maka parameter DESC diperlukan.
- *Query* yang terluar adalah untuk membatasi jumlah baris pada akhir pencarian hasil. Query terluar termasuk komponen-komponen di bawah ini :

1. *Pseudo-column* ROWNUM dimana nilainya selalu bertambah yang dimulai dengan 1 sampai dengan nilai yang ada pada subquery.
2. Klausa WHERE yang menunjukkan n baris yang ingin dituju nilainya oleh klausa WHERE yang paling luar harus menggunakan operator " <" atau "<=".

Contoh berikut menjelaskan bagaimana menampilkan nama dan gaji dari tiga orang yang mempunyai gaji tertinggi dari Table EMP.

Contoh :

```
                                                    emp
SELECT   ROWNUM AS RANK,ENAME,SAL

        FROM            (SELECT ename, sal  FROM
                                ORDER BY sal DESC)

        WHERE    ROWNUM <= 3 ;
```