JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA YOGYAKARTA

MODUL 11
BAHASA QUERY

Membuat dan Mengelola Tabel dengan *Data Definition Language (DDL)*

A. TUJUAN

Mahasiswa dapat:

1. Mendeskripsikan beberapa obyek basisdata
2. Membuat tabel
3. Mendeskripsikan tipe data yang dapat digunakan untuk menspesifikasikan definisi kolom
4. Mengubah definisi tabel
5. Melakukan *drop, rename* dan *truncate* tabel
6. Memahami bagaimana *constraint* dibuat pada saat membuat tabel.

B. LANDASAN TEORI & LANGKAH PRAKTIKUM

B.1. Obyek Basisdata

Beberapa obyek basisdata yang terdapat dalam Oracle adalah *table, view,* dan *index.* Table adalah unit simpanan dasar, terdiri atas baris dan kolom. Sedangkan view merupakan *subset* dari satu atau lebih tabel. Berikut adalah aturan penamaan tabel dan kolom dalam Oracle:

1. Harus dimulai dengan huruf
2. Panjang karakter antara 1 – 30 karakter
3. Hanya boleh memuat karakter A-Z, a-z, 0-9, _, $, dan #
4. Tidak boleh sama dengan obyek lain yang dimiliki oleh *user* yang sama
5. Tidak boleh berupa kata kunci dalam Oracle

Sebaiknya nama tabel dan obyek lain dalam basisdata cukup deskriptif dan bisa menggambarkan isi dari obyek tersebut. Nama obyek tidak *case sensitive*.
Terdapat 2 jenis table dalam Oracle, yaitu*:*

1. *User table*:
   - Adalah kumpulan table yang dibuat dan dirawat oleh user.
   - Berisi informasi tentang user
2. *Data dictionary*:
   - Merupakan kumpulan table yang dibuat dan dirawat oleh server Oracle.
   - Berisi informasi tentang basisdata.

Untuk melihat nama tabel-tabel yang dimiliki oleh user, digunakan perintah berikut:

```
SELECT  table_name
FROM   user_tables;
```

B.2. Membuat tabel dengan perintah CREATE TABLE

Membuat tabel untuk menyimpan data dapat dilakukan dengan perintah SQL CREATE TABLE. Perintah ini merupakan salah satu perintah yang termasuk kelompok *Data Definition Language (DDL).* Perintah-perintah DDL adalah kumpulan perintah yang digunakan untuk membuat, memodifikasi, atau menghapus struktur basisdata Oracle. Perintah-perintah ini memiliki dampak langsung pada basisdata, dan sekaligus juga menyimpan informasinya dalam kamus data.

Untuk dapat membuat tabel, pengguna harus mempunyai hak (*privilege*) untuk CREATE TABLE dan sebuah media simpanan yang dipakai untuk menyimpan obyek tabel. Hak ini biasanya diberikan dan diatur oleh *Database Administrator (DBA)*. Matakuliah Administrasi Basisdata akan membicarakan topik tentang hal tersebut lebih lanjut.

B.2.1. Sintaks perintah CREATE TABLE

Sintaks dari perintah CREATE TABLE adalah sebagai berikut:

```
CREATE TABLE [schema.] table
            (column datatype [DEFAULT expr] [, …]);
```

Keterangan:

- *schema* adalah nama pemilik tabel
- *table* adalah nama tabel
- *DEFAULT expr* menspesifikasikan nilai default jika nilai kolom tersebut tidak diisi pada saat memakai perintah INSERT
- *column* adalah nama kolom
- *datatype* adalah tipe data dan panjang data kolom

Tipe data yang didukung oleh Oracle adalah sbb:

TIPE DATA	DESKRIPSI
VARCHAR2 (*size*)	Data karakter dengan panjang bervariasi (ukuran maksimum *size* harus didefinisikan, panjang minimum 1; maksimum 4000)
CHAR [(*size*)]	Data karakter dengan panjang tertentu (ukuran default dan minimum *size* 1, maksimum 2000)
NUMBER [(p,s)]	Bilangan dengan presisi *p* dan *scale s* Presisi adalah jumlah digit desimal, dan *scale* adalah jumlah digit di sebelah kanan titik desimal. Presisi dapat berkisar dari 1 sampai 38, sedangkan *scale* dapat berkisar dari -84 sampai 127.
DATE	Tanggal dan waktu hingga detik terdekat, bernilai antara January 1, 4712 SM hingga December 1, 9999 M
LONG	Karakter dengan panjang yang bervariasi, dapat mencapai panjang hingga 2GB
CLOB	Karakter dengan panjang mencapai 4 GB

RAW (*size*)	Data biner mentah dengan panjang *size*. Nilai *size* harus dispesifikasikan, dengan *size* maksimum 2000.
LONG RAW	Data biner mentah dengan panjang bervariasi hingga mencapai 2 GB
BLOB	Data biner hingga mencapai 4 GB
BFILE	Data biner yang disimpan dalam file eksternal, bisa mencapai 4 GB
ROWID	Sistem bilangan basis 64 yang merepresentasikan alamat unik baris dalam tabel

Sebagai catatan:

- Sebuah kolom bertipe LONG tidak dapat dicopy saat tabel dibuat dengan menggunakan *subquery.*
- Sebuah kolom bertipe LONG tidak dapat dimasukkan dalam klausa GROUP BY maupun HAVING BY.
- Hanya satu kolom LONG yang dapat digunakan dalam satu tabel.
- Kolom LONG tidak dapat dikenai *constraint.*
- Lebih baik menggunakan kolom bertipe CLOB daripada LONG.

B.2.2. Penggunaan perintah CREATE TABLE

Contoh 1

- Membuat tabel DEPT dengan kolom terdiri atas deptno, dname, dan loc.

```
CREATE TABLE dept
            (deptno     NUMBER(2),
             dname      VARCHAR2(14),
             loc        VARCHAR2(13));
```

Hasil pembuatan tabel dapat dicek dengan menggunakan perintah DESCRIBE dept (perintah dapat disingkat dengan DESC dept).

B.3. *Constraints* (Kendala)

Server Oracle menggunakan *constraint* (kendala) untuk mencegah data yang tidak valid masuk ke dalam tabel. Kita dapat menggunakan *constraint* untuk melakukan hal-hal berikut:

- Memaksakan diimplementasikannya aturan terhadap data dalam tabel setiapkali suatu baris data akan disisipkan (*insert*), diubah (*update*), atau dihapus (*delete*) dari tabel. *Constraint* harus dipenuhi agar operasi-operasi tersebut berhasil dilakukan.
- Mencegah pengahapusan dari tabel jika terdapat ketergantungan terhadap tabel lain.
- Menyediakan aturan terhadap alat bantu (*tools*) Oracle, seperti Oracle Developer.

Kendala integritas data mencakup hal-hal berikut:

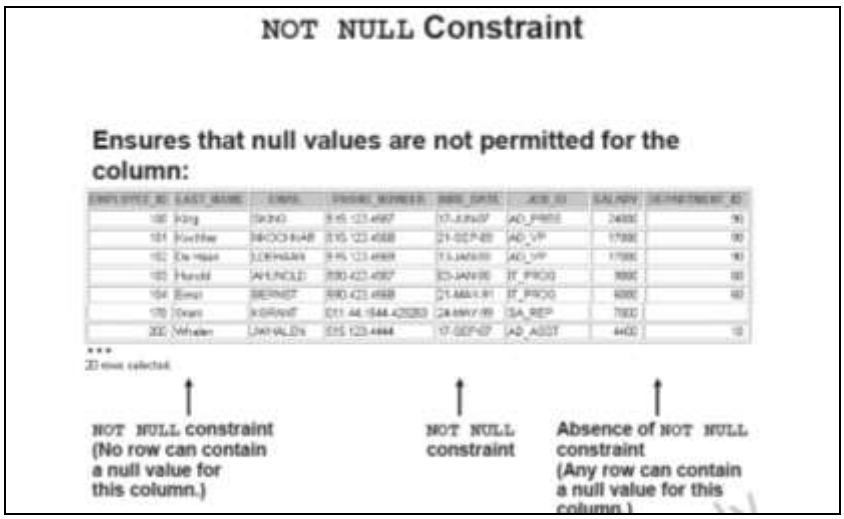
CONSTRAINT	DESKRIPSI
NOT NULL	Menspesifikasikan bahwa kolom tidak boleh berisi nilai null. Kolom tanpa *constraint* NOT NULL, secara

	default dapat menerima nilai NULL. *Constraint* ini harus didefinisikan pada level kolom.
UNIQUE	Menspesifikasikan sebuah kolom atau kombinasi kolom yang nilainya harus unik untuk semua baris dalam tabel. Tanpa *constraint* NOT NULL, kolom yang diberi *constraint* UNIQUE tetap dapat menerima nilai NULL. *Constraint* ini dapat didefinisikan pada level tabel maupun kolom.
PRIMARY KEY	Kunci primer yang secara unik mengidentifikasi tiap baris dalam tabel.
FOREIGN KEY	Menetapkan dan memaksakan hubungan kunci tamu (*foreign key*) antara suatu kolom dengan kolom dalam tabel yang dirujuk (*referenced table*).
CHECK	Menspesifikasikan suatu kondisi yang harus dipenuhi. Sebuah kolom dapat berisi beberapa *constraint* CHECK sekaligus. *Constraint* ini didefinisikan pada level kolom atau tabel.

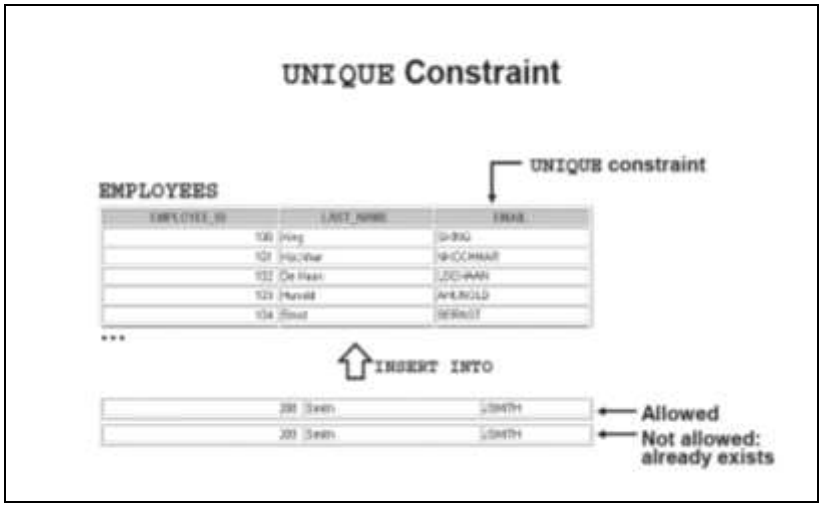
Beberapa hal yang perlu diperhatikan terkait dengan *constraint*:

- Semua *constraint* harus disimpan dalam kamus data.
- Aturan pemberian nama *constraint* sama dengan aturan pemberian nama obyek dalam basisdata (lihat B.1.).
- Sebaiknya beri nama *constraint* yang bermakna untuk memudahkan pemakaiannya.
- Jika kita tidak memberi nama *constraint*, server Oracle akan menghasilkan nama dengan format SYS_C*n*, dimana *n* adalah suatu integer sedemikian hingga nama *constraint* itu unik.
- *Constraint* dapat dibuat pada saat:
  - Pada saat yang bersamaan dengan saat tabel dibuat, atau
  - Setelah tabel dibuat
- *Constraint* dapat didefinisikan pada level kolom atau level tabel.
- *Constraint* yang didefinisikan pada level tabel dapat dilihat pada tabel kamus data USER_CONSTRAINTS.

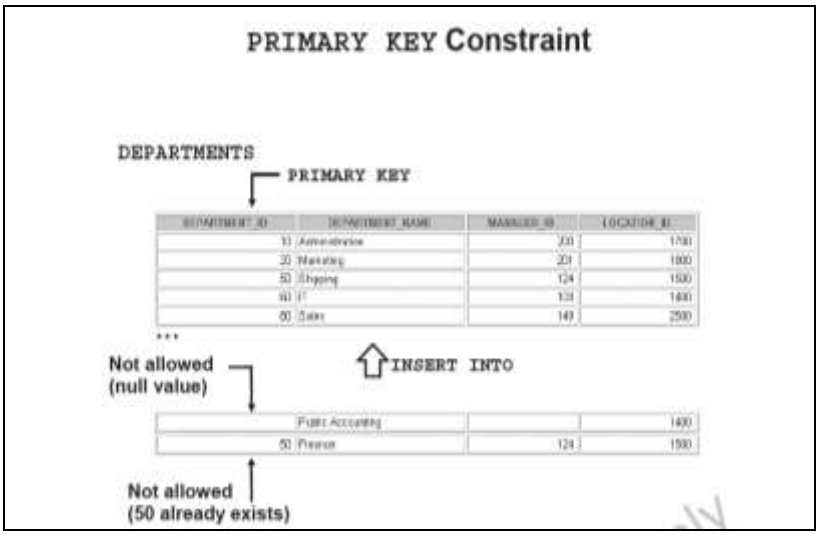
Gambar-gambar berikut menunjukkan ilustrasi berbagai macam jenis *constraint*.



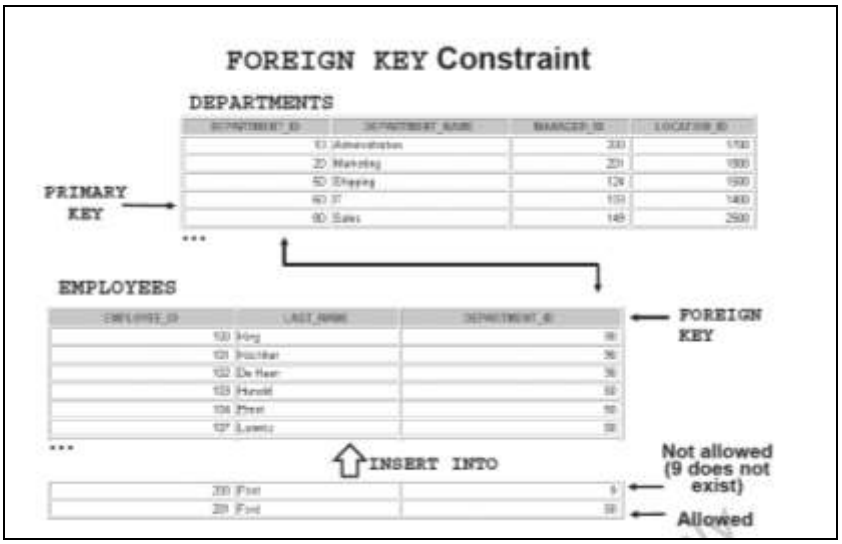
Gambar 11.1 Ilustrasi NOT NULL *Constraint*



Gambar 11.2 Ilustrasi UNIQUE *Constraint*



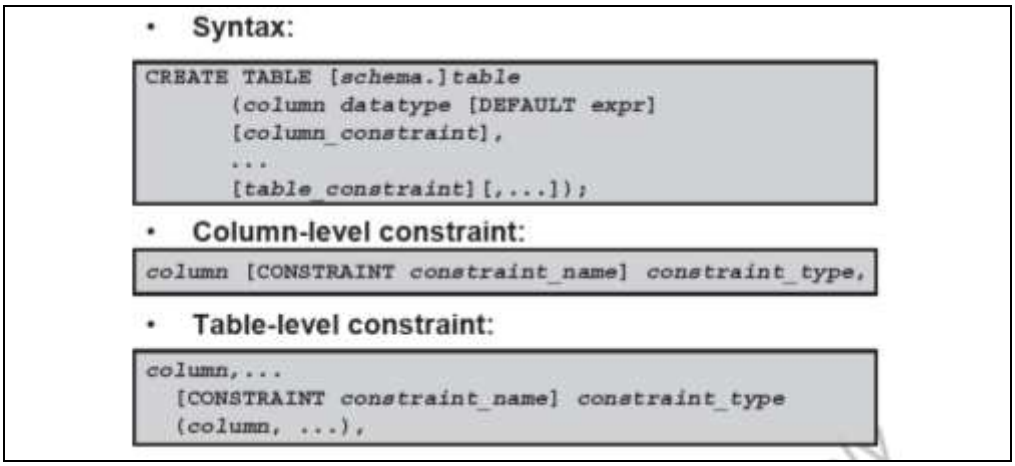
Gambar 11.3 Ilustrasi PRIMARY_KEY *Constraint*



Gambar 11.4 Ilustrasi FOREIGN_KEY *Constraint*

B.3.1 Mendefinisikan *constraint* pada saat membuat tabel

Sintaks dari cara ini adalah sebagai berikut:



Keterangan:

- *schema* adalah nama pemilik tabel
- *table* adalah nama tabel
- *DEFAULT expr* menspesifikasikan nilai default jika nilai kolom tersebut tidak diisi pada saat memakai perintah INSERT
- *column* adalah nama kolom
- *datatype* adalah tipe data dan panjang data kolom
- *column_constraint* adalah kendala integritas (*integrity constraint*) yang menjadi bagian dari definisi kolom
- *table_constraint* adalah kendala integritas (*integrity constraint*) yang menjadi bagian dari definisi tabel.

Dalam sintaks di atas, *constraint* pada level kolom dimasukkan pada saat pendefinisian kolom. Sedangkan *constraint* pada level tabel dimasukkan pada bagian akhir definisi tabel dan harus merujuk pada kolom atau kolom-kolom yang terkait dengan *constraint* tersebut, yang ditempatkan diantara tanda kurung. *Constraint* NOT NULL harus didefinisikan pada level kolom. Sedangkan *constraint* yang diterapkan terhadap lebih dari satu kolom harus didefinisikan pada level tabel.

Khusus untuk *constraint* FOREIGN_KEY yang sering disebut juga **integritas referensial (*referential integrity*),** terdapat beberapa kata kunci penting yaitu:

- FOREIGN KEY: untuk mendefinsikan kolom dalam tabel anak pada *constraint* level tabel.
- REFERENCES: mengidentifikasi tabel dan kolom dalam tabel induk.
- ON DELETE CASCADE: menghapus baris data yang terkait, yang terdapat dalam tabel anak, ketika baris tersebut dihapus dari tabel induk. Tabel anak adalah tabel yang memiliki foreign key.
- ON DELETE SET NULL: mengubah nilai foreign key dalam tabel anak menjadi NULL ketika baris yang terkait dengan foreign key tersebut dihapus dari tabel induk.
- Tanpa ada klausa ON DELETE CASCADE atau ON DELETE SET NULL, maka suatu baris dalam tabel induk tidak boleh dihapus selama masih ada tabel anak yang merujuknya. Hal ini disebut sebagai RESTRICT RULE dan menjadi perilaku *default*.

Contoh 2

- Berikut ini adalah contoh pembuatan tabel EMPLOYEE2 yang menggunakan *constraint* NOR NULL, PRIMARY KEY, UNIQUE KEY, dan FOREIGN KEY.

```
CREATE TABLE employees2 (
       employee_id    NUMBER(6)
           CONSTRAINT emp2_id_pk            PRIMARY KEY
       ,first_name    VARCHAR2(20)
       ,last_name          VARCHAR2(25)
           CONSTRAINT emp2_last_name_nn     NOT NULL
       ,email         VARCHAR2(25)
           CONSTRAINT emp2_email_nn         NOT NULL
           CONSTRAINT emp2_email_uk         UNIQUE
       ,phone_number  VARCHAR2(20)
       ,hire_date     DATE
           CONSTRAINT emp2_hire_date_nn     NOT NULL
       ,job_id        VARCHAR2(10)
           CONSTRAINT emp2_job_nn           NOT NULL
       ,salary        NUMBER(8,2)
           CONSTRAINT emp2_salary_ck        CHECK (salary>0)
       ,commission_pct NUMBER(2,2)
       ,manager_id    NUMBER(6)
       ,department_id NUMBER(4)
           CONSTRAINT emp2_dept_fk          REFERENCES
                 departments(department_id));
```

Lihatlah hasil tabel yang telah dibuat dengan menggunakan perintah :

```
DESC employees2;
```

B.3.2. Pelanggaran *Constraint*

Jika suatu kolom berisi *constraint,* maka akan muncul error saat kita mencoba melanggar aturan tersebut.

Contoh 3

- Jika kita ingin menghapus record yang memiliki nilai yang terkait dengan referential integrity constraint, maka akan muncul error.

```
DELETE from departments
WHERE  department_id = 60;
```

Contoh di atas akan memunculkan pesan error karena department number digunakan sebagai *foreign key* dalam tabel EMPLOYEES.

B.4. Membuat tabel dengan menggunakan sintaks *subquery*

Cara lain untuk membuat tabel adalah dengan menggunakan klausa *AS subquery* yang akan membuat tabel sekaligus mengisi baris-baris data dengan menggunakan *subquery*.

B.4.1. Sintaks perintah membuat tabel dengan menggunakan *subquery*

Sintaks dari cara ini adalah sebagai berikut:

```
CREATE TABLE  table
                [(column, column …)])
AS subquery;
```

Keterangan:

- *table* adalah nama tabel
- *column* adalah nama kolom, nilai default, dan kendala integritas
- *subquery* adalah perintah SELECT yang mendefinisikan baris-baris data yang akan disisipkan dalam tabel

Beberapa hal yang harus diperhatikan dalam cara ini adalah:

- Tabel dibuat dengan menyebutkan nama kolom, dan selanjutnya baris-baris data yang didapat melalui perintah SELECT disisipkan dalam tabel.
- Definisi kolom dapat hanya berisi nama kolom dan nilai default.
- Jika spesifikasi kolom disebutkan, jumlah kolom harus sama dengan jumlah kolom dalam daftar SELECT pada bagian *subquery.*
- Jika spesifikasi kolom tidak disebutkan, maka nama kolom dalam tabel harus sama dengan nama kolom dalam *subquery*.
- Hanya definisi kolom yang dikirimkan/disalin ke tabel baru, sedangkan aturan integritas tidak dikirimkan/disalin ke tabel baru.

B.4.2. Penggunaan perintah membuat tabel dengan *subquery*

Contoh 4

- Membuat tabel DEPT80, yang berisi semua employee yang bekerja di department 80. Data untuk tabel DEPT80 diambil dari tabel EMPLOYEES. Cek hasil pembuatan tabel dengan menggunakan perintah DESCRIBE dan perintah SELECT.

```
CREATE TABLE dept80
AS
   SELECT employee_id, last_name,
          salary*12 ANNSAL,
          hire_date,
   FROM  employees
   WHERE department_id = 80;
```

Lihatlah hasil tabel yang telah dibuat dengan menggunakan perintah :

DESC dept80;

Pastikan untuk menggunakan nama alias kolom jika menggunakan ekspresi. Ekspresi SALARY * 12 diberi nama alias ANNSAL. Jika tidak maka akan terjadi error. **Cobalah dengan menghilangkan ANNSAL dari perintah di atas dan amati hasilnya**.

B.5. Mengubah tabel dengan perintah ALTER TABLE

Setelah kita membuat tabel, kita bisa mengubah struktur tabel dengan menggunakan perintah ALTER TABLE untuk melakukan beberapa hal berikut:

- Menambah kolom baru
- Memodifikasi kolom yang sudah ada
- Mendefinisikan nilai default untuk sebuah kolom baru
- Menghapus kolom

B.5.1. Sintaks perintah mengubah tabel dengan perintah ALTER TABLE

Sintaks dari perintah ALTER TABLE ini untuk menambah, memodifikasi, dan menghapus kolom adalah sebagai berikut:

```
ALTER TABLE   table
ADD           (column datatype [DEFAULT expr]
              [, column datatype] …);
```

```
ALTER TABLE   table
MODIFY        (column datatype [DEFAULT expr]
              [, column datatype] …);
```

```
ALTER TABLE   table
DROP          (column);
```

Keterangan:

- *table* adalah nama tabel
- ADD|MODIFY|DROP adalah tipe modifikasi
- *column* adalah nama kolom, nilai default, dan kendala integritas
- *subquery* adalah perintah SELECT yang mendefinisikan baris-baris data yang akan disisipkan dalam tabel

B.5.2. Penggunaan perintah ALTER TABLE

Contoh 5

- Menambah kolom job_id ke dalam tabel DEPT80. Cek hasil modifikasi tabel dengan menggunakan perintah DESCRIBE.

```
ALTER TABLE dept80
ADD (job_id VARCHAR(9));
```

Contoh 6

- Memodifikasi kolom last_name dalam tabel DEPT80 menjadi VARCHAR(30). Cek hasil modifikasi tabel dengan menggunakan perintah DESCRIBE.

```
ALTER TABLE dept80
MODIFY (last_name VARCHAR(30));
```

Contoh 7

- Menghapus kolom job_id dalam tabel DEPT80. Cek hasil modifikasi tabel dengan menggunakan perintah DESCRIBE.

```
ALTER TABLE dept80
DROP COLUMN job_id;
```

B.6. Mengubah nama obyek basisdata

Perintah DDL tambahan adalah perintah RENAME yang dipakai untuk mengganti nama obyek basisdata (lihat B.1.) seperti tabel, view, sequence, atau synonim.

B.6.1. Sintaks perintah untuk mengubah nama obyek basisdata

Sintaks dari perintah RENAME adalah sebagai berikut:

```
RENAME old_name TO new_name;
```

Keterangan:

- *old_name* adalah nama tabel, view, sequence, atau synonim yang lama.
- *new_name* adalah nama tabel, view, sequence, atau synonim yang baru.

Perintah RENAME hanya dapat dilakukan oleh pemilik tabel.

B.6.2. Penggunaan perintah RENAME

Contoh 8

- Ubah tabel DEPT menjadi tabel DETAIL_DEPT. Cek hasil modifikasi tabel dengan menggunakan perintah DESCRIBE dan SELECT.

```
RENAME dept80 TO detail_dept;
```

B.7. Menghapus tabel dengan perintah DROP TABLE

Perintah DROP TABLE akan menghapus definisi dari table dalam Oracle. Perintah ini akan mengakibatkan:

- semua data dalam tabel akan dihapus
- semua indeks yang terkait dengannya akan dihapus
- semua transaksi yang masih berstatus "*pending*" akan *"committed"*

Yang berhak melakukan DROP TABLE adalah user yang membuat tabel tersebut atau yang memiliki *privilege/*hak DROP ANY TABLE. Setelah perintah DROP TABLE dieksekusi, tidak ada fasilitas untuk membatalkannya (*roll back*).

B.7.1. Sintaks perintah menghapus tabel dengan perintah DROP TABLE

Sintaks dari perintah DROP TABLE adalah sebagai berikut:

```
DROP TABLE   table;
```

Keterangan:

- *table* adalah nama tabel

B.7.2. Penggunaan perintah DROP TABLE

Contoh 9

- Hapus tabel detail_dep dan cek hasil modifikasi tabel dengan menggunakan perintah DESCRIBE dan SELECT.

```
DROP TABLE detail_dept;
```

C. TUGAS

1. Buatlah sebuah tabel bernama DEPT1 dengan struktur tabel seperti gambar di bawah ini. Cek hasilnya dengan menampilkan struktur tabel DEPT1 yang baru saja Anda buat.

Column Name	ID	NAME
Key Type	Primary Key	
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

Name	Null?	Type
ID		NUMBER(7)
NAME		VARCHAR2(25)

2. Isi data DEPT1 dengan data yang diambil dari tabel DEPARTMENT. Pilih kolom yang sesuai dengan struktur tabel DEPT1.

3. Buat tabel EMP1 dengan struktur seperti gambar berikut. Cek hasilnya dengan menampilkan struktur tabel EMP1 yang baru saja Anda buat.

Nama Kolom	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Tipe kunci (key type)	PRIMARY_KEY			
Nulls/ Unique				

FK tabel *)				DEPT1
FK kolom **)				ID
Tipe data	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Panjang data	7	25	26	7

4. Modifikasi tabel EMP1 di atas, dengan menambah panjang FIRST_NAME menjadi 50. Cek hasil modifikasinya dengan menampilkan struktur tabel EMP1.

5. Buat tabel EMP2 berdasar struktur tabel EMPLOYEES. Pilih hanya kolom EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, dan DEPARTMENT_ID. Dalam tabel EMP2 beri nama masing-masing kolom tersebut sebagai ID, FIRST_NAME, LAST_NAME, SALARY, dan DEPT_ID.

6. Drop tabel EMP1.

7. Gantilah nama tabel EMP2 menjadi STAFF.

8. Drop kolom FIRST_NAME dari tabel STAFF. Cek struktur tabel STAFF.

9. Cek apakah Anda masih memiliki tabel EMPLOYEES2 yang ada pada contoh 2. Jika belum, buatlah.

10. Buatlah tabel DEPARTMENTS2 dengan struktur sebagai berikut. Cek hasilnya dengan menampilkan struktur tabel DEPARTMENTS2 yang baru saja Anda buat.

Column Name	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
Key Type	PRIMARY_KEY			
Nulls/Unique	NOT NULL	NOT NULL	NOT NULL	NOT NULL
FK Table			EMPLOYEES2	
FK Column			EMPLOYEE_ID	
Data Type	NUMBER	VARCHAR2	NUMBER	NUMBER
Length	10	30	10	10

11. Coba hapus tabel EMPLOYEES2, dan amati apa yang terjadi.
12. Hapus tabel DEPARTMENTS2 dan amati apa yang terjadi.
13. Ulangi langkah 11 dan amati apa yang terjadi.

D. DAFTAR PUSTAKA

1. *Oracle Database 10g : SQL Fundamental I,* Oracle Inc. 2004

☺☺☺ No oil, if olives aren't squeezed
No wine, if grapes aren't pressed
If you feel pressure in life,
God's bring out the best in you !!☺☺☺