**Modul I**

**Sekilas Oracle**

Oracle merupakan salah satu Relational Database Management System (RDBMS) yang handal yang dikembangkan oleh Oracle Corporation yang didirikan pada tahun 1977 di RedWood, California oleh Larry Ellison, Bob Miner dan Eat Oates. Pada mulanya Oracle Corporation dikenal sebagai Relational Software Incorporated (RSI). Mereka berhasil mengembangkan sebuah prototype, yaitu versi 1 menggunakan bahasa C dan SQL Interface.

Oracle Server merupakan RDBMS yang mengontrol :

* Penyimpanan Data pada area perancangan basis data.
* Akses Data untuk aplikasi tertentu, dengan menggunakan Teknik optimisasi.
* Keamanan Basis Data dan pengaturan izin suatu operasi untuk pemakai tertentu.
* Konsistensi dan proteksi data, termasuk perintah yang diizinkan dan mekanisme locking.
* Komunikasi dan integritas data untuk basis data tersebar.

**SQL, SQL\*Plus, dan PL/SQL**

*SQL* merupakan bahasa yang digunakan untuk mengakses Basis Data relational termasuk Oracle, dapat digunakan pada setiap tools Oracle yang membutuhkan akses pada file Basis data

*PL/SQL* merupakan bahasa procedural pada Oracle untuk menulis aplikasi lojik dan untuk memanipulasi data diluar basis data. Salah satu bagiannya adalah SQL dan tersedia pada setiap produk CDE serta pada Oracle Server.

*SQL\*Plus* merupakan produk Oracle yang menggunakan SQL dan PL/SQL yang ditambah dengan sintak bahasa tambahan untuk mengontrol kelakuan produk, dan format output dari SQL Queries.

**Pengantar Basis Data Relasional**

Basis Data Relasional merupakan kumpulan Objek atau Tabel yang saling berelasi secara kompak dan bergantung satu sama lainnya, dimana tersusun secara normal memenuhi kaidah tertentu.

**Entity Relationalship Diagram (ERD)**

ERD merupakan skema pembentuk Basis Data Relasional, dimana sebelum kita membangun tabel-tabel maka kita perlu menggambarkan terlebih dahulu skema entitas pembangun tabel-tabel tersebut. Karena pada dasarnya sebuah tabel adalah sebuah object, dimana sebuah object merupakan wujud dari sebuah entitas atau lebih. Baca lebih lanjut tentang pembangun tabel dari ERD pada buku *Sistem Basis Data*.

Mapping ERD ke Tabel :

1.

R1: PK1

M

R2 : PK2

N

R3

Jika sebuah relasi R1 (atribut kunci PK1) memiliki kardinalitas Banyak(M) berhubungan (melalui relasi R3) dengan relasi R2 (atribut kunci PK2) memiliki kardinaltias Banyak (N). maka tabel yang terbentuk adalah :

* Tabel R1.
* Tabel R2.
* Tabel R3 (dengan mengikutsertakan atribut kunci R1:PK1 dan atribut kunci R2: PK2 didalam atribut tabelnya.

2.

R1: PK1

M

R2 : PK2

1

R3

Jika sebuah relasi R1 (atribut kunci PK1) memiliki kardinalitas Banyak(M) berhubungan (melalui relasi R3) dengan relasi R2 (atribut kunci PK2) memiliki kardinalitas satu (1). maka tabel yang terbentuk adalah :

* Tabel R1 (mengikutsertakan atribut kunci R2:PK2 didalam atribut tabelnya sebagai foreign key).
* Tabel R2.

3.

R1: PK1

1

R2 : PK2

1

R3

Jika sebuah relasi R1 (atribut kunci PK1) memiliki kardinalitas Satu(1) berhubungan (melalui relasi R3) dengan relasi R2 (atribut kunci PK2) memiliki kardinalitas Satu (1). maka tabel yang terbentuk adalah :

* Tabel R1.
* Tabel R2.
* Mengikutsertakan atribut kunci R2:PK2 ke tabel R1 mengikutsertakan atribut kunci R1:PK1 ke tabel R2 didalam atribut tabelnya sebagai foreign key.

**Normalisasi**

Proses me-normal-kan Tabel menjadi lebih efisien. Beberapa kaidah normalisasi yaitu *First Norm-Form, Second Norm-Form, Third Norm-Form.* Penjelasan mengenai normalisasi akan diterangkan pada saat praktikum berlangsung.

**Tipe Data**

**1. varchar2 (size)** string yang memiliki panjang karakter variable dengan panjang maximal sebesar size.

**2. char(size)**  string dengan panjang karakter tetap sebesar ukuran size

**3. number(p,s)** tipe data number memiliki presisi p dan s digit dibelakang koma Jika kita abaikan p dan s (contoh number), berarti dianggap sebagai tipe number floating point.

**4. long** data karakter dengan ukuran panjang bervariasi, hingga mencapai 2 gigabytes. (tipe data ini tidak dapat digunakan sebagai key).

**5. date** tipe data tanggal

**6. raw(size)** tipe data berupa binary data dengan ukuran maksimal 255 bytes

**7. long raw** tipe data berupa binary data dengan ukuran maksimal 2 gigabytes tidak dikonversi oleh oracle (data mentah apa adanya).

**8. lob** (large object binary), untuk menyimpan data yang tidak terstruktur seperti gambar, video, dokumen, internasional karakter set atau external data dan lainnya. Lob menampung data sampai 4GB.

**Data Definition Language**

**Pengertian**

DDL merupakan bagian dan' perintah SQL untuk membuat, memodifikasi atau menghapus struktur basis data Oracle.

**Perintah - Perintah DDL**

**Create Table** ( Membuat Tabel )

Sintaks:

**CREATE TABLE** nama\_tabel

(

Nama\_atribut1 tipe1 constraint\_pk\_nama\_tabel PRIMARY KEY,

Nama\_atribut2 tipe2 [unique][notnull/null][default] <nilai default> [check],

............

);

Contoh:

CREATE TABLE dokter

(

Id\_dokter char(5) constraint pk\_dokter primary key,

nama varchar(20)

);

Untuk melihat struktur (nama dan tipe atribut) sebuah tabel dapat digunakan perintah **DESCRIBE**

Contoh:

SQL> Describe dokter;

**Constraint**

Constraint yaitu aturan - aturan yang terdapat pada atribut tabel. Beberapa tipe constraint sbb:

* PRIMARY KEY.
* FOREIGN KEY.
* NULL / NOT NULL.
* CHECK.
* UNIQUE.

Primary Key (PK)

Primary Key secara implisit mernbetuk keunikan kolorn atau kornbinasinya dan indeks unik dibuat untuk menangani key ini. Hanva ada satu key primer yang diperbolehkan untuk setiap tabel. Secara implisit PK adalah NOT NULL.

Contoh:

CREATE TABLE dokter

(

Id\_dokter char(5) **constraint pk\_dokter primary key**,

nama varchar (20)

);

Foreign Key

Menyediakan acuan terintegrasi dalam satu tabel atau antar tabel dan digunakan sebagai hubungan antar primary atau unique key lain. Biasanya sebuah foreign key pada suatu tabel merupakan primary key tabel yang lain.

Contoh :

create table obat

(

id\_obat char(5),

id\_dokter char(5) **constraint fk\_obat\_dokter references** dokter **ON DELETE CASCADE**,

jumlah number(3),

dosis varchar2(10),

constraint pk\_obat primary key(id\_obat,id\_dokter)

);

**On Delete Cascade Option -** Digunakan untuk membentuk keterkaitan antara tabel yang terhubung. Jika suatu baris data pada tabel parent dihapus maka, kolom pada tabel child yang terhubung juga akan dihapus.

Not Null / Null

Jika sebuah kolom pada database tidak boleh kosong (harus terisi), maka constraint NOT NULL harus diberikan. Dengan demikian pada saat tabel diisi dengan data (misalnya melalui instruksi INSERT), maka kolom tersebut harus terisi. Jika tidak system akan menolaknya. Begitu pula dengan NULL

Check

Mendefinisikan suatu kondisi nilai tiap baris harus tepat / harus ada. Menjamin bahwa niIai kolom adalah dalam range nilai tertentu. Kondisi tersebut digunakan untuk bentukan query, dengan beberapa pengecualian:

* Subquery tidak diijinkan.
* Acuan pseudo-colomns seperti SYSDATE tidak diijinkan.

Unique

Constraint ini dirancang untuk kolom atau kombinasi kolom sebagai suatu key unik. Artinva untuk suatu kolom yang sama tidak boleh memiliki nilai yang sama. Key unique menjamin bahwa nilai kolom adalah tunggal (tidak memiliki duplikat). Primary key vs Unique, keduanya adalah kunci kata yang mempunyai arti sama. Oracle membedakan UNIQUE sebagai sebuah field yang dapat mempunyal nilai "NULL" (kosong), namun bila mempunyai nilai, maka field tersebut harus tunggal.

Contoh:

create table apotek

(

kode\_apotek char(5)constraint pk\_apotek primary key,

id\_obat char(5)constraint fk\_apotek\_obat

references obat(id\_obat),

nama varchar2(20) NOT NULL,

alamat varchar(30) constraint ck\_alamat check (alamat=’Buah batu’),

no\_kontrak char(5) constraint unq\_apotek\_kontrak unique(kode\_apotek,no\_kontrak)

);

**Create View ( Membuat Tabel Semu )**

Untuk mendefinisikan suatu tabel semu, (view) dari kombinasi beberapa tabel yang dihubungkan dengan kesamaan kondisi tertentu

**CREATE** [or replace] [force] [noforce] **VIEW** nama\_view [(nama\_koloml), ....]

**As** subquery [with check option]

Sintaks:

Ket:

* **or replace** - mendefinisikan kembali view yang sudah ada.
* **noforce** - view hanya, akan dibuat j ika tabel induk telah dibuat.
* **force** - view dapat dibuat walaupun tabel induk belum dibuat.
* **with check option** - view akan menvalidasi data yang diinsert atau diupdate ke view.

Contoh:

CREATE VIEW view\_dokter as

Select a.id\_dokter, a.nama\_depan, a.nama\_belakang, b.nama

From dokter a, obat b

Where a.id\_dokter=b.id\_obat

**Create Sequence**

Digunakan untuk membangkitkan serangkaian nilai serial yang unik.

Sintaks:

CREATE SEQUENCE nama-sequence

[increment by integer]

[start with integer]

[maxvalue integer | nomaxvalue]

[minvalue integer | nominvalue]

[cycle I nocycle]

[chace integer | nochace]

Sequence mempunyal 6 karakteristik antara lain:

**Nama** - nama dari sequence yang akan didefinisikan

**Starting value** - nilal awal yang dihasilkan oleh sequence

**Increment** - rentang atau selisih nilai yang dihasilkan oleh sequence

**Minimum value -** nilal terendah yang dapat dihasilkan oleh sequence

**Maximum value -** nilal tertinggi yang dapat dihasilkan oleh sequence, baik minimum maupun maximum value bersifat optional artinya boleh diisi boleh tidak.

**Cycle** - sequence akan kembali ke nilai terendah yang ditetapkan minimum.

**Chace** - menentukan berapa banyak nilai dari sequence yang akan dialokasikan ke memori.

Contoh:

CREATE SEQUENCE seq\_bulan

INCREMENT BY 1

START WITH 1

MAXVALUE 12;

**Create Synonim**

Synonim adalah nama lain (alias) dari suatu tabel, view, sequence.

Sintaks:

**CREATE** [PUBLIC] **SYNONIM** nama\_synonim

For object;

Option PUBLIC dipakai agar semua user dapat mengakses synonim yang dibuat. Jika public tidak ditulis maka sebagai defaultnya SQL akan menggunakan PRIVATE yang berarti hanya dapat diakses pada skema tersebut.

Maksud penggunaan synonim adalah:

* Untuk mengganti atau menyingkat narna yang sulit diingat dari sebuah objek.
* User dapat mengakses data pada tabel yang berisi informasi yang sensitive dan private melalui synonym tanpa harus mengetahui nama tabel aslinya sehingga anda tidak dapat melakukan perubahan pada struktur tabel.

Contoh:

CREATE SYNONIM medicine for obat;

**Create Index**

Berfungsi untuk membuat index pada satu atau lebih kolom dari suatu tabel guna mempercepat pengaksesan data pada tabel.

Sintaks:

**CREATE INDEX** nama\_index

**On** nama-table(nama\_atributl, .....);

Contoh:

CREATE INDEX dokter\_idx on dokter(id\_dokter);

**Create User**

Perintah CREATE USER digunakan untuk mendefinisikan suatu user beserta passwordnya.

Sintaks:

**CREATE USER** nama\_user

**IDENTIFIED BY** password | **EXTERNALLY**;

Option **EXTERNALLY** - oracle akan memverifikasi akses user melalui sistem operasi.

Contoh:

CREATE USER praktikan

IDENTIFIED BY orclst;

**Drop Table**

Perintah ini digunakan untuk menghapus definisi tabel ORACLE yeitu sernua data dan indeks tabel.

Sintaks :

**DROP TABLE** nama\_tabel [CASCADE CONSTRAINT];

Contoh :

DROP TABLE dokter;

Option **CASCADE CONSTRAINT** - akan menghapus pula konstrain referensi integrity yang terhubung.

**Drop Index**

Perintah DROP INDEX digunakan untuk menambah / menghilangkan indeks: telah dibuat di database.

Sintaks :

**DROP INDEX** nama\_indeks;

Contoh :

DROP INDEX dokter\_idx;

**Drop Sequence**

Perintah DROP SEQUENCE digunakan untuk membuang & menghilangkan sequence yang telah dibuat pada data base.

Sintaks :

**DROP SEQUENCE** nama\_sequence;

Contoh :

DROP SEQUENCE seq\_bulan;

**Drop Synonim**

Untuk menghilangkan synonim yang telah dibuat, dapat menggunakan pefintah DROP SYNONYM.

Sintaks :

**DROP SYNONIM** nama\_synonim;

Contoh :

DROP SYNONIM medicine;

Drop User

Perintah DROP USER digunakan untuk menghilangkan pendefinisian terhadap user.

Sintaks :

**DROP USER** nama\_user [CASCADE];

Option CASCADE digunakan untuk menghilangkan user beserta database objectnya.

Contoh :

DROP USER assisten;

**Drop View**

Perintah DROP VIEW digunakan untuk menghapus view yang ada pada database.

Sintaks :

**DROP VIEW** nama\_view;

Contoh :

DROP VIEW view\_obat;

**Alter Table**

Digunakan untuk mengubah tabel dan constraint.yang berlaku. Database object yang dapat dimodifikasi strukturnya dengan perintah ALTER adalah index, sequence, tabel, user, dan view.

Sintaks :

**ALTER TABLE** nama\_tabel

[ADD nama\_atribut tipe\_data constraint],

[ADD table\_constraint],

[MODIFY nama - atribut tipe\_data constraint],

[DROP PRIMARY KEY],

[DROP UNIQUE nama-atribut]

**Add**

Digunakan untuk menambah kolom atau constraint dari suatu. tabel.

* Untuk menambah kolom pada tabel dokter dengan nama gaji maka perintahnya sbb:

ALTER TABLE dokter

ADD (gaji varchar2(8));

* Untuk menambah constraint bahwa gaji tidak boleh lebih dari 15000, maka perintahnya sbb:

ALTER TABLE dokter

ADD(CHECK(gaji<=15000) );

**Modify**

Digunakan untuk mengubah definisi suatu kolom.

Contoh:

Memodifikasi kolom nama agar nama tidak diijinkan kosong

ALTER TABLE dokter

MODIFY(Nama varchar(20) NOT NULL);

**Drop**

Digunakan untuk menghapus bentuk constraint pada suatu tabel

Contoh:

ALTER TABLE dokter DROP constraint ck\_alamat;

ALTER TABLE obat DROP primary key;

**Alter Index**

Perintah ini digunakan untuk memodifikasi alokasi penyimpanan dalam indeks.

Sintaks :

**ALTER INDEX** nama\_indeks

[INITRANS integer]

[MAXTRANS integer]

[STORAGE storage\_clause];

**Alter Sequence**

Perintah ALTER SEQUENCE digunakan untuk memodifikasi karaktersitik sequence sehingga dapat bersifat increment, nilai minimum, nilai maksimum, atau cycle.

Sintaks :

**ALTER SEQUENCE** nama-sequence

[INCREMENT BY integer]

[MAXVALUE integer | NOMAXVALUE]

[MINVALUE integer | NOMINVALUE]

[CYCLE | NO CYCLE];

contoh :

ALTER SEQUENCE seq\_bulan

INCREMENT BY 2;

**Alter User**

Perintah ALTER USER berfngsi untuk merubah password (autentifikasi) user

Sintaks :

**ALTER USER** nama\_user

[IDENTIFIED {BY password | EXTERNALY}];

Contoh :

ALTER USER assisten

IDENTIFIED BY basdat;

**Alter View**

Perintah ALTER VIEW digunakan untuk mengkompilasi ulang sebuah view.

**ALTER VIEW** nama view

**COMPILE**;

Sintaks :

Contoh :

ALTER VIEW view\_dokter

COMPILE;

**Comment**

Perintah COMMENT digunakan utnuk menambah keterangan tentang tabel, view, snapshot atau kolom. Karakter yang digunakan maksinial 255 char.

Sintaks :

**COMMENT ON** [TABLE | COLOUMN][nama\_tabel | namakolom];

Contoh :

* Menambah komentar pada tabel dokter

COMMENT ON TABLE dokter IS 'Informasi dokter';

* Menambah keterangan pada kolom id\_pasien di tabel pasien

COMMENT ON COLOUMN pasien.id\_pasien IS 'nomor pasien’;

* Menghapus komentar pada kolom kamar di tabel pasien

COMMENT ON COLOUMN pasien.kamar IS ‘’;

**Rename**

Perintah RENAME digunakan untuk mengganti narna tabel, view, sequence, dan private synonym.

Sintaks :

**RENAME** nama\_lama **TO** nama\_baru;

Contoh :

RENAME apotek TO toko\_obat;

**Truncate Tuble**

Perintah TRUNCATE TABLE digunakan untuk menghapus semua baris tabel. Perintah ini sama dengan dengan perintah DELETE yang berulang untuk semua baris. Namun perintah ini lebih efisien dibandingkan dengan perintah DELETE.

Sintaks :

**TRUNCATE TABLE** nama\_tabel [REUSE STORAGE];

Option **REUSE STORAGE** - digunakan untuk tetap menyimpan tempat baris tabel yang, dihapus untuk digunakan kembali. Default perintah TRUNCATE adalah bila data dihapus maka ruang datanya pun ikut terhapus.

Contoh

TRUNCATE TABLE pasien;

**Catatan :**