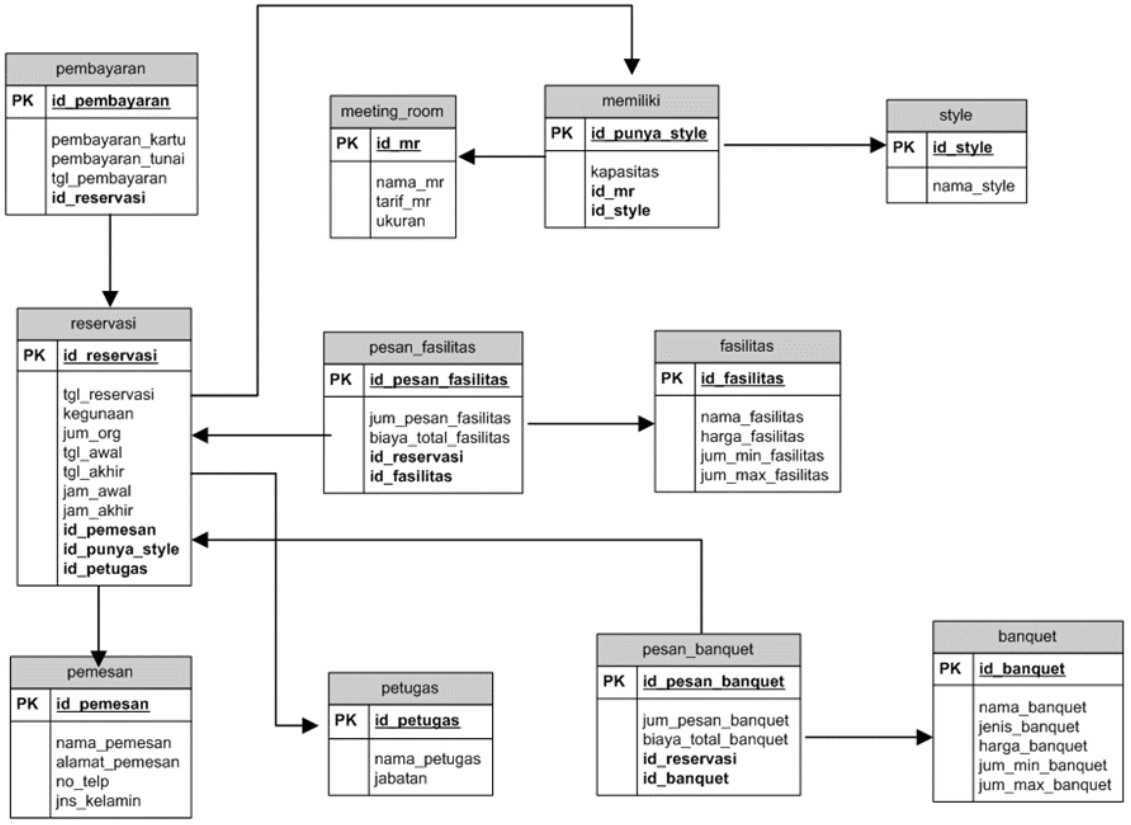
MODUL 2

DML (DATA MANIPULATION LANGUAGE)

DAN QUERY DASAR

Studi Kasus Perhotelan :



I. DATA MANIPULATION LANGUANGE (DML)

Data Manipulation Language merupakan bahasa yang memungkinkan pengguna untuk mengakses dan mengubah data yang sesuai dengan model datanya selain itu DML juga merupakan konsep yang menerangkan bagaimana menambah, mengubah dan menghapus baris tabel.

- **Insert**

  Berfungsi untuk menambahkan baris baru ke tabel

  Sintaks :

```
INSERT INTO <nama_tabel> (nama_kolom1, nama_kolom2, ….)
Values (nilai1, nilai2, …);
```

Misal ada table sebagai berikut :

```
create table reservasi(
   id_reservasi varchar(10),
   tgl_reservasi date,
   kegunaan varchar(30) not null,
   jum_org integer, tgl_awal date,
   tgl_akhir date,
   jam_awal char(6),
   jam_akhir char(6),
   constraint pk_reservasi primary key (id_reservasi)
)
create table penyewa(
   id_reservasi varchar(10),
   tgl_reservasi date,
   kegunaan varchar(20) not null,
   constraint fk_penyewa foreign key (id_reservasi) references  reservasi
(id_reservasi) on delete cascade
)

INSERT INTO penyewa (id_reservasi, tgl_reservasi, kegunaan) values
('82900','2009-04-22','meeting')
```

Jika ada data-data yang ada di tabel lain dan ingin digunakan untuk di insertkan ke tabel baru maka querynya adalah:

```
INSERT INTO <nama_tabel> [(nama_kolom1, nama_kolom2,…)] <query>;
```

Contoh:

```
INSERT INTO penyewa
Select id_reservasi, tgl_reservasi, kegunaan from reservasi
Where id_reservasi='82900'
```

- **Update**

  Berfungsi untuk memodifikasi nilai pada baris tabel.
  Sintaks :

```
UPDATE <nama_tabel>
```

```
Set <nama_kolom1>=<nilai_ekspresi1>, <nama_kolom2>=<nilai_ekspresi2> ,
….
Where <kondisi>;
```

Apabila klausa “where” tidak digunakan maka semua data atribut yang dimodifikasi pada tabel tersebut akan berubah.

Contoh :

```
UPDATE reservasi
set kegunaan='resepsi pernikahan'
where id_reservasi='82900';
```

Sama dengan perintah **insert**, data-data yang ada di tabel lain dapat digunakan untuk mengembalikan data sebagai data nilai baru.

Contoh:

```
UPDATE reservasi
set kegunaan=(select kegunaan from reservasi where id_reservasi='82900')
where id_reservasi='83900';
```

- **Delete**

  Berfungsi untuk menghapus baris tunggal atau lebih dari satu baris berdasarkan kondisi tertentu.

  Sintaks :

```
DELETE FROM <nama_tabel>
WHERE <kondisi>;
```

  Contoh :

```
DELETE FROM reservasi
WHERE id_reservasi=’82900’;
```

  Jika klausa “where” tidak digunakan maka semua data pada tabel akan terhapus. Sebagai sintaks alternative untuk menghapus semua data pada table digunakan perintah.

  Sintaks:

```
TRUNCATE TABLE <table>
```

- **Merge**
  Berfungsi untuk melakukan update maupun insert ke suatu tabel tertentu berdasarkan kondisi dari tabel lain. Dengan perintah merge ini, data akan diinsertkan apabila data tersebut tidak ada di tabel tujuan dan akan diupdatekan apabila data tersebut telah ada di tabel tujuan.

  Sintaks:

```
MERGE INTO <nama_tabel> <nama_alias_tabel>
USING (table|view|sub_query) alias
ON (join condition)
        WHEN MATCHED THEN
                UPDATE SET [(column1=column_val1),
(column2=column_val2),…]
        WHEN NOT MATCHED THEN
                INSERT (column_list)
                VALUES (column_values);
```

  Contoh:

```
MERGE INTO copy_reservasi cr
        USING reservasi r
        ON cr.id_reservasi=r.id_reservasi
                WHEN MATCHED THEN
                        UPDATE SET r.kegunaan=cr.kegunaan
                WHEN NOT MATCHED THEN
                        INSERT VALUES (cr.kegunaan);
```

II. KONSEP DASAR TRANSAKSI

Sebuah transaksi adalah sebuah unit eksekusi program yang mengakses dan mungkin mengubah beberapa item data.

Dalam transaksi, DBMS harus menjamin 4 sifat :

- **Atomicity**

Jika semua operasi pada sebuah transaksi sukses maka transaksi akan dianggap sukses dan begitu juga sebaliknya.

- **Consistency**

Ada beberapa operasi yang tidak dapat diakses. Hal ini bertujuan untuk menjaga kekonsistenan database.

- **Isolation**

Jika ada beberapa transaksi berlangsung bersamaan maka masing-masing transaksi tidak boleh mengetahui transaksi lain yang sedang berlangsung dan hasil sementara masing-masing transaksi harus disembunyikan.

- **Durability**

Apabila sebuah transaksi sukses dilakukan maka perubahan-perubahan yang dibuat terhadap database akan bersifat permanen, bahkan jika terjadi kesalahan.

Perintah Dasar Transaksi

Commit

Perintah ini berfungsi untu mengakhiri suatu transaksi yang telah dirubah menggunaka perintah DML. Data-data akan bersifat permanen setelah menggunakan perintah “commit”. Jika pengguna tidak menggunakan perintah ini, maka masih dimungkinkan untuk mengembalikan (undo) semua modifikasi terakhir kali. Perintah commit ini dapat dilakukan secara eksplisit ataupun implisit.
Cara Penggunaan:

```
> SELECT * FROM pegawai;

+----------+-------+--------------------+

| nip      | nama  | alamat             |

+----------+-------+--------------------+

| 93032000 | arie  | Payakumbuh Utara   |

| 93032100 | alid  | Payakumbuh Selatan |

| 93032100 | halwa | Payakumbuh Barat   |

| 93032300 | aldam | Payakumbuh Timur   |

+----------+-------+--------------------+
> START TRANSACTION;

> INSERT INTO pegawai VALUES (93032400,"cesa","Payakumbuh Tengah");

> COMMIT;
```

```
> SELECT * FROM pegawai;
+----------+-------+--------------------+
| nip      | nama  | alamat             |
+----------+-------+--------------------+
| 93032000 | arie  | Payakumbuh Utara   |
| 93032100 | alid  | Payakumbuh Selatan |
| 93032100 | halwa | Payakumbuh Barat   |
| 93032300 | aldam | Payakumbuh Timur   |
| 93032400 | cesa  | Payakumbuh Tengah  |
+----------+-------+--------------------+
```

Rollback

Data-data yang telah dirubah dengan perintah DML masih bisa dikembalikan ke kondisi awal transaksi. Pengembalian tersebut dapat menggunakan perintah rollback.

Cara Penggunaan:

```
> SELECT * FROM pegawai;
+----------+-------+--------------------+
| nip      | nama  | alamat             |
+----------+-------+--------------------+
| 93032000 | arie  | Payakumbuh Utara   |
| 93032100 | alid  | Payakumbuh Selatan |
| 93032100 | halwa | Payakumbuh Barat   |
| 93032300 | aldam | Payakumbuh Timur   |
| 93032400 | cesa  | Payakumbuh Tengah  |
+----------+-------+--------------------+


> DELETE FROM pegawai WHERE nip="93032400";
> SELECT * FROM pefawai;
```

```
+----------+-------+--------------------+
| nip      | nama  | alamat             |
+----------+-------+--------------------+
| 93032000 | arie  | Payakumbuh Utara   |
| 93032100 | alid  | Payakumbuh Selatan |
| 93032100 | halwa | Payakumbuh Barat   |
| 93032300 | aldam | Payakumbuh Timur   |
+----------+-------+--------------------+
> ROLLBACK;
> SELECT * FROM pegawai;
+----------+-------+--------------------+
| nip      | nama  | alamat             |
+----------+-------+--------------------+
| 93032000 | arie  | Payakumbuh Utara   |
| 93032100 | alid  | Payakumbuh Selatan |
| 93032100 | halwa | Payakumbuh Barat   |
| 93032300 | aldam | Payakumbuh Timur   |
| 93032400 | cesa  | Payakumbuh Tengah  |
+----------+-------+--------------------+
```

Savepoint

Berfungsi untuk membagi-bagi suatu transaksi menjadi tahapan serta memberikan nama ataupun tanda pada bagian yang dimaksud. Dengan adanya Savepoint, perintah Rollback dapat digunakan untuk membatalkan perintah-perintah DML.

Cara penggunaan:

>START TRANSACTION;

>SAVEPOINT SavePoint1;

III. QUERY DASAR

Query merupakan statement dalam SQL yang berfungsi untuk mendapatkan atau mengambil data dari database (satu atau beberapa tabel/view) berdasarkan kriteria-kriteria tertentu yang diberikan[3]. Sebuah query selalu diawali dengan SELECT statement. Secara umum sintaks query sederhana dapat dituliskan seperti di bawah ini [3].

Sintaks query sederhana :

```
SELECT [ALL|DISTINCT] (nama_kolom1, nama_kolom2, ….)

FROM (nama_tabel_sumber1, nama_tabel_sumber2, …. )

WHERE (kondisi )
```

Keterangan :

- SELECT
  Untuk menspesifikasikan nama-nama kolom yang akan ditampilkan. Nama-nama kolomnya dituliskan setelah klausa ini.
- DISTINCT
  Untuk menampilkan record-record hasil query yang nilai atau valuesna berbeda. Tujuan penggunaan klausa ini untuk menghilangkan redundancy hasil query.
- ALL
  Untuk menampilkan seluruh record meskipun ada beberapa record yang nilainya sama. Secara default setiap statement SELECT menggunakan klausa ALL di dalamnya.
- FROM
  Untuk mendefinisikan nama tabel yang mengandung kolom yang terdaftar dalam SELECT
- WHERE
  Untuk mendefinisikan kondisi atau kriteria untuk menyaring data yang diambil dari tabel sumber. Terdiri dati 3 elemen, yaitu nama kolom, kondisi perbandingan dan nama konstanta atau daftar nilai.

Contoh :

❖ Menampilkan semua data (gunakan tanda ‘*’) dari tabel pemesan

```
SELECT * FROM pemesan;
```

Menampilkan beberapa kolom yang spesifik (penggunaan klausa WHERE)

Contoh :

- Menampilkan kolom id_reservasi dan kegunaan dari tabel reservasi yang jumlah orangnya 100 (pengkondisian).

```
SELECT id_reservasi, kegunaan
FROM reservasi
WHERE jum_org=100;
```

- Keterangan :
  a. Jika pengkondisian bertipe data string atau date maka gunakan tanda petik satu (‘_’)

```
SELECT id_reservasi, tgl_reservasi
FROM reservasi
WHERE kegunaan = ‘resepsi pernikahan’;
```

  b. Nilai pada tipe data string harus case sensitive
  c. Nilai pada tipe data harus format sensitive. Default dari format date adalah DD-MM-YY

❖ Menampilkan nama style yang ada di tabel style (penggunaan klausa DISTINCT)

```
SELECT DISTINCT nama_style
FROM style ;
```

Coba bandingkan hasilnya dengan

```
SELECT nama_style
FROM style ;
```

❖ Menampilkan jumlah petugas berdasarkan jenis jabatannya dari tabel petugas (penggunaan klausa GROUP BY)

```
SELECT jabatana, COUNT (jabatan)
FROM petugas
GROUP BY jabatan ;
```

❖ Menampilkan semua nama_mr dan tariff_mr terurut secara DESCENDING berdasarkan tariff_mr (penggunaan klausa ORDER BY), default dari order by adalah ASCENDING, jadi klausa ascending boleh ditulis atau tidak .

```
SELECT nama_mr, tarif_mr
FROM meeting_room
ORDER BY tarif_mr DESC;
```

Beberapa hal yang perlu diperhatikan dalam penulisan statement dalam SQL :

a. SQL statement tidak case sensitive
b. SQL statement dapat dibuat dalam satu baris atau lebih
c. Keyword tidak dapat disingkat
d. Klausa biasanya ditempatkan di baris yang terpisah
e. Spasi digunakan untuk mempermudah

Ada kemungkinan untuk menyisipkan item lain pada klausa SELECT yaitu :

1. Ekspresi Aritmatik
   Terdiri atas : * / + -
   Contoh :

```
SELECT tgl_reservasi, jum_org+2
FROM reservasi
WHERE id_reservasi='83900';
```

   Keterangan :
   a. Perkalian dan pembagian didahulukan daripada penjumlahan dan pengurangan
   b. Operator dengan prioritas yang sama (misal perkalian dan pembagian) akan dieksekusi dari operator yang paling kiri (pertama ditulis)
   c. Tanda kurung '()' digunakan untuk menandakan bahwa statement di dalamnya harus dieksekusi terlebih dahulu.

2. Kolom alias
   Digunakan untuk mengganti heading dari hasil query atau nama tabel. Kolom alias ditulis setelah klausa SELECT. Secara default, alias ditampilkan dengan huruf capital dan tidak ada spasi. Dibutuhkan tanda kutip dua ("_") jika nama alias mengandung spasi atau karakter khusu atau jika kasusnya sensitive. Dapat juga menggunakan keyword **AS.**
   Contoh :

```
SELECT nama_pemesan Nama, alamat_pemesan AS alamat, no_telp
"nomor telepon"
FROM pemesan
WHERE id_pemesan='98000';
```

3. Kolom konkat
   Operator konkat (||) atau penggabungan digunakan untuk menghubungkan suatu kolom dengan kolom lain, ekspresi aritmatik atau nilai konstan untuk membentuk ekspresi karakter.
   Contoh 3:

```
SELECT nama_petugas ||'  '|| jabatan  "Data Petugas"
FROM petugas ;
```

4. Literal
   Merupakan karakter, ekspresi atau bilangan yang terdapat pada klausa SELECT namun bukan merupakan salah satu field atau kolom pada tabel yang digunakan (tabel setelah klausa FROM) dan ingin ditampilkan pada setiap baris hasil query.
   Contoh 4:

```
SELECT nama_petugas, 'menjabat sebagai', jabatan
FROM petugas ;
```

IV. VARIABEL DALAM MYSQL

Ada dua cara untuk menetapkan nilai ke variabel yang ditentukan pengguna. Cara pertama adalah dengan menggunakan SET, dan cara kedua untuk menetapkan nilai ke variabel adalah dengan menggunakan pernyataan SELECT.

SET @variable_name := value;

SELECT @variable_name := value;

Contoh :
SET @counter := 100;

V. OPERATOR-OPERATOR DALAM MYSQL

1. Operator aritmatika
   - \+ Pertambahan
   - \- Pengurangan
   - * Perkalian
   - / Pembagian
   - % Modulo
2. Operator Bitwise
   - & Bitwise AND
   - | Bitwise OR
   - ^ Bitwise XOR
3. Operator Perbandingan
   - = Sama dengan
   - > Lebih besar dari
   - < Kurang dari
   - >= Lebih dari atau sama dengan
   - <= Kurang dari atau sama dengan
   - <> Tidak sama dengan
4. Operator Gabungan
   - += Tambahkan sama
   - -= Kurangi sama dengan

*= Kalikan sama dengan
/= Bagi sama dengan
%= Modulo sama dengan
&= Bitwise AND sama dengan
^-= Bitwise XOR sama dengan
|*= Bitwise OR sama dengan

5. Operator Logika

ALL	TRUE jika semua nilai subquery memenuhi kondisi
AND	TRUE jika semua kondisi yang dipisahkan oleh AND bernilai TRUE
ANY	TRUE jika salah satu nilai subquery memenuhi kondisi
BETWEEN	BENAR jika operan berada dalam kisaran perbandingan
EXISTS	TRUE jika subquery mengembalikan satu atau lebih record
IN	BENAR jika operan sama dengan salah satu daftar ekspresi
LIKE	BENAR jika operan cocok dengan pola
NOT	Menampilkan catatan jika kondisi TIDAK BENAR
OR	TRUE jika salah satu kondisi yang dipisahkan oleh OR adalah TRUE
SOME	TRUE jika salah satu nilai subquery memenuhi kondisi

VI. FUNGSI-FUNGSI

- Fungsi Karakter dan String

Fungsi Manipulasi Kasus

LOWER(column/expression)	Mengkonversikan column/expression menjadi huruf kecil semua
UPPER(column/expression)	Mengkonversikan column/expression menjadi huruf kapital semua
INITCAP(column/expression)	Mengkonversikan huruf pertama dari setiap kata menjadi huruf kapital

Fungsi Manipulasi Karakter

1. Fungsi CONCAT
   Menggabungkan dua atau lebih string/nilai field.
   1. SELECT CONCAT (str1,[separator], str2,...);
   2. SELECT nama_field (yang akan ditampilkan), CONCAT (nama_field1,[separator], nama_field2,...);
2. Fungsi CONCAT_WS
   Menggabungkan dua atau lebih string (kolom) dengan separator diantara masing-masing string/nilai field.

1. SELECT CONCAT_WS ([separator], str1, str2,...);

2. SELECT nama_field (yang akan ditampilkan), CONCAT_WS ([separator], nama_field1, nama_field2,...);

3. Fungsi SUBSTR
   Mengambil atau memotong string dimulai dari karakter awal sebanyak panjang karakter.
   1. SUBSTR(string, awal, panjang);
   2. SUBSTRING(string, awal, panjang);
   3. SUBSTRING(string FROM awal FOR panjang);
   4. MID(string, awal, panjang);
4. Fungsi LENGTH
   Menghitung panjang string.
   1. LENGTH (string);
   2. OCTET_LENGTH (string);
   3. CHAR_LENGTH (string);
   4. CHARACTER_LENGTH (string);
5. Fungsi LEFT
   Memotong string dari sebelah kiri sebanyak panjang karakter.
   Sintaks : LEFT (string, panjang);
6. Fungsi RIGHT
   Memotong string dari sebelah kanan sebanyak panjang karakter.
   Sintaks: LEFT (string, panjang); RIGHT (string, panjang);
7. Fungsi LTRIM
   Menghilangkan spasi di awal string (kiri).
   Sintaks: LTRIM (string);
8. Fungsi RTRIM
   Menghilangkan spasi di akhir string (kanan).
   Sintaks: LTRIM (string); RTRIM (string, panjang);
9. Fungsi TRIM
   Menghilangkan spasi di awal (kiri) dan di akhir string (kanan).
   Sintaks: TRIM(string);
10. Fungsi REPLACE
    Mengganti suatu string dengan string yang lain.
    REPLACE (string, from_str, to_str);
11. Fungsi REPEAT
    Menduplikasi suatu string sebanyak jumlah.
    Sintaks: REPEAT (string, jumlah);
12. Fungsi REVERSE
    Membalik string.
    Sintaks: REPEAT (string, jumlah); REVERSE (string);

- Fungsi tanggal dan Waktu

1. Fungsi NOW() dan SYSDATE()
   Mendapatkan tanggal dan waktu sistem sekarang.
   Sintaks: NOW();
   SYSDATE();
2. Fungsi MONTH
   Mendapatkan urutan bulan (integer) dari suatu tanggal yang diberikan dalam setahun, dimana 1=Januari, 2=Februari, dst.
   Sintaks: MONTH ('tanggal');
3. Fungsi WEEK
   Mendapatkan urutan minggu (integer) dari suatu tanggal yang diberikan dalam setahun.
   Sintaks:WEEK ('tanggal');;
4. Fungsi YEAR
   Mendapatkan bilangan tahun dari suatu tanggal yang diberikan.
   Sintaks: YEAR (now());
5. Fungsi HOUR
   Mendapatkan bilangan jam dari suatu parameter waktu yang diberikan.
   Sintaks: HOUR (now());
6. Fungsi MINUTE
   Mendapatkan bilangan menit dari suatu parameter waktu yang diberikan.
   Sintaks: MINUTE (now());
7. Fungsi SECOND
   Mendapatkan bilangan detik dari suatu parameter waktu yang diberikan.
   Sintaks: SECOND (now());
8. Fungsi DATE_ADD dan DATE_SUB
   Fungsi-fungsi di bawah ini digunakan untuk menambah suatu tanggal.
   Sintaks: DATE_ADD(date,INTERVAL expr type)
   DATE_SUB(date,INTERVAL expr type)
9. Fungsi DATE_FORMAT
   Fungsi yang digunakan untuk menentukan format tampilan tanggal.
   Sintaks: DATE_FORMAT(date, format)
10. Fungsi TIME_FORMAT
    Fungsi yang digunakan untuk menentukan format tampilan waktu.
    Sintaks: TIME_FORMAT(time, format)
    Berikut ini adalah tampilan tanggal dan waktu serta penggunaannya:
    1. %M : Nama bulan (January ... December)
    2. %W : Nama hari dalam seminggu (Sunday...Saturday)
    3. %D : Urutan hari dalam sebulan
    4. %Y : Tahun, 4 digit
    5. %y : Tahun, 2 digit
    6. %a : Nama hari dalam seminggu (Sunday...Saturday)
    7. %H : Jam, dalam format 24.

8. %i : Menit, 00-59
9. %s : Detik, 00-59

- Fungsi Numerik
  Fungsi-fungsi yang berhubungan dengan operasi numerik.
  Operasi aritmatika dalam MySQL terdiri dari:
  + : Pertambahan
  − : Pengurangan
  ∗ : Perkalian
  / : Pembagian
  % : Sisa hasil bagi, modulus

  1. Fungsi ABS
     Mengambil nilai absolut dari bilangan ***x***.
     Sintaks: ABS(bilangan);
  2. Fungsi MOD
     Mendapatkan hasil pengoperasian *m* modulus *n*
     Sintaks: MOD (m, n);
  3. Fungsi FLOOR
     Mengambil nilai integer terbesar yang tidak lebih besar dari bilangan pecahan yang dituliskan.
     Sintaks: FLOOR(bilangan_pecahan);
  4. Fungsi CEILING
     Mengambil nilai integer terkecil yang tidak lebih kecil dari bilangan pecahan yang dituliskan.
     Sintaks: CEILING(bilangan_pecahan);
  5. Fungsi ROUND
     Mengambil melakukan pembulatan bilangan pecahan ***x*** sebanyak ***d*** tempat presisi.
     Sintaks: ROUND(x,d);
  6. Fungsi POWER
     Mengambil hasil pemangkatan dari ***x n***.
     Sintaks: POWER(x,n);
  7. Fungsi TRUNCATE
     Memotong bilangan ***x*** sepanjang ***d*** tempat desimal.
     Sintaks: TRUNCATE(x,d);
  8. Fungsi GREATEST
     Mengambil nilai terbesar dari suatu kumpulan nilai.
     Sintaks: GREATEST(nilai1, nilai2, nilai3, ...);
  9. Fungsi MAX
     Mendapatkan nilai terbesar dari suatu ekspresi (query).
     Sintaks:
     Fungsi Lainnya SELECT MAX(nama_field) FROM nama_tabel;

VII. EKSPRESI KONDENSIONAL

- **CASE Expression**

Fungsi **CASE** memfasilitasi kebutuhan kondisional dengan mengerjakan pekerjaan yang ada dalam statement **IF-THEN-ELSE**

Contoh :

```
SELECT nama_pemesan
CASE jns_kelamin
        WHEN 'L' THEN 'Pria'
        WHEN 'P' THEN 'Wanita'
         ELSE 'Salah Input'
 END
 FROM pemesan;
```

(Query di atas akan mengecek nilai pada kolom jns_kelamin, apabila ditemukan *L*, maka akan diubah menjadi *Pria*, dan apabila ditemukan *P*, akan diubah menjadi *Wanita*, sedangkan apabila ditemukan selain P atau L, maka akan diisi dengan *salah input*).

VIII. MENGGUNAKAN FUNGSI AGREGASI

Fungsi agregasi akan melakukan operasi pada kelompok-kelompok baris data.

Beberapa fungsi yang termasuk dalam fungsi agregasi adalah :

AVG([DISTINCT|ALL]expr)	Memperoleh nilai rata-rata dari seluruh nilai pada suatu kolom
COUNT({*|[DISTINCT|ALL]expr})	Memperoleh jumlah baris yang tidak mengandung nilai null dari suatu kolom
MAX([DISTINCT|ALL]expr)	Memperoleh nilai maksimum dari suatu kolom
MIN([DISTINCT|ALL]expr)	Memperoleh nilai minimum dari suatu kolom
SUM([DISTINCT|ALL]expr)	Memperoleh nilai penjumlahan seluruh baris pada suatu kolom
STDDEV([DISTINCT|ALL]expr)	Memperoleh nilai standar deviasi dari suatu kolom
VARIANCE([DISTINCT|ALL]expr)	Memperoleh nilai variansi dari suatu kolom

Contoh :

```
SELECT AVG(harga_fasilitas),
       MAX(harga_fasilitas),
       MIN(harga_fasilitas),
       SUM(harga_fasilitas),
       STDDEV(harga_fasilitas),
       VARIANCE(harga_fasilitas), COUNT(harga_fasilitas)
FROM  fasilitas ;
```

IX. MULTIPLE SELECTION CONDITION

Untuk kasus-kasus query yang memerlukan select dengan kondisi yang banyak, kita dapat menggunakan klausa where dengan penghubung AND atau OR.

Jika kita menggunakan AND di antara dua kondisi maka hasil query adalah query dengan kedua kondisi tersebut terpenuhi. Jika kita menggunakan OR maka oracle hanya akan mengecek kedua kondisi secara terpisah , hasil query kondisi satu akan ditambahkan dengan hasil query kondisi dua.

Contoh :

Menampilkan nama fasilitas yang harganya lebih besar dari 10000 dan jum_max_fasilitasnya lebih kecil dari 10.

```
SELECT nama_fasilitas
FROM fasilitas
WHERE harga_fasilitas > 10000 AND jum_max_fasilitas < 10;
```

Yang ditampilkan adalah yang kondisi dalam klausa where nya terpenuhi kedua-duanya.

Oracle akan mendahulukan AND dibanding OR dan untuk query yang membutuhkan kondisi kompleks, pengunaan “( )” sangat membantu untuk memilah-milah urutan proses.

Contoh :

SELECT nama_fasilitas, count(nama_fasilitas)

FROM fasilitas

WHERE harga_fasilitas > 10000 AND jum_max_fasilitas < 10 OR jum_max_fasilitas > 20 AND nama_fasilitas LIKE ‘K%’ OR nama_fasilitas LIKE ‘A%’;

Bandingkan hasilnya dengan contoh :

SELECT nama_fasilitas, count(nama_fasilitas)

FROM fasilitas

WHERE (harga_fasilitas > 10000 AND (jum_max_fasilitas < 10 OR jum_max_fasilitas > 20)) AND

(nama_fasilitas LIKE ‘K%’ OR nama_fasilitas LIKE ‘A%)’;

X. KLAUSA ORDER BY

Klausa **ORDER BY** digunakan untuk **mengurutkan** hasil query berdasarkan kolom tertentu. Klausa yang mengikutinya adalah **ASC** untuk terurut secara **ASCENDING** (dari kecil ke besar), dan **DESC** untuk terurut secara **DESCENDING** (dari besar ke kecil). Secara **default**, klausa ORDER BY menggunakan **ASC** di dalamnya. Klausa ini bisa dijalankan setelah klausa FROM . Jika terdapat klausa WHERE, maka ORDER BY ditempatkan setelah klausa WHERE tersebut (biasanya ORDER BY ditempatkan di paling akhir suatu query).

Bentuk umumnya adalah :

SELECT [ALL|DISTINCT] *{*|nama_kolom1, nama_kolom2, …}*

FROM *{nama_tabel_sumber1, nama_tabel_sumber2, ...}*

[WHERE *kondisi***]**

[ORDER BY *nama_kolom* [ASC|DESC]**]**

Contoh :
Menampilkan id_reservasi dan tgl_reservasi yang jumlah orangnya lebih besar dari 50, dan terurut secara descending berdasarkan tgl_reservasi dan id_reservasi.

SELECT id_reservasi
FROM reservasi
WHERE jum_org > 50
ORDER BY tgl_reservasi DESC;

Pada klausa ORDER BY ini, kita juga bisa mnegurutkan berdasarkan beberapa kolom tertentu. Untuk pengurutan dengan banyak kolom, maka ditambahkan koma setelah nama kolom.
Contoh :

```
SELECT id_reservasi, tgl_reservasi
FROM reservasi
WHERE jum_org > 50
ORDER BY tgl_reservasi, id_reservasi DESC;
```

XI. KLAUSA GROUP BY

Klausa **GROUP BY** digunakan untuk membagi-bagi baris dalam sebuah tabel berdasarkan kelompok tertentu. Eksekusi query dilakukan pada setiap kelompok record. Penggunaan GROUP BY ini dilakukan untuk SELECT yang mengandung **fungsi agregasi** (COUNT, MAX, MIN, SUM, AVG, dll).

Bentuk Umumnya adalah :

SELECT [ALL|DISTINCT] ***{*|nama_kolom1, nama_kolom2, …}***

FROM ***{nama_tabel_sumber1, nama_tabel_sumber2, ...}***

[WHERE ***kondisi*****]**

[GROUP BY ***nama_kolom*****] [HAVING** ***kondisi*****]**

Contoh:

Menampilkan jumlah petugas berdasarkan jenis jabatannya dari table petugas.

```
SELECT jabatan, count(jabatan)
FROM petugas
GROUP BY jabatan;
```

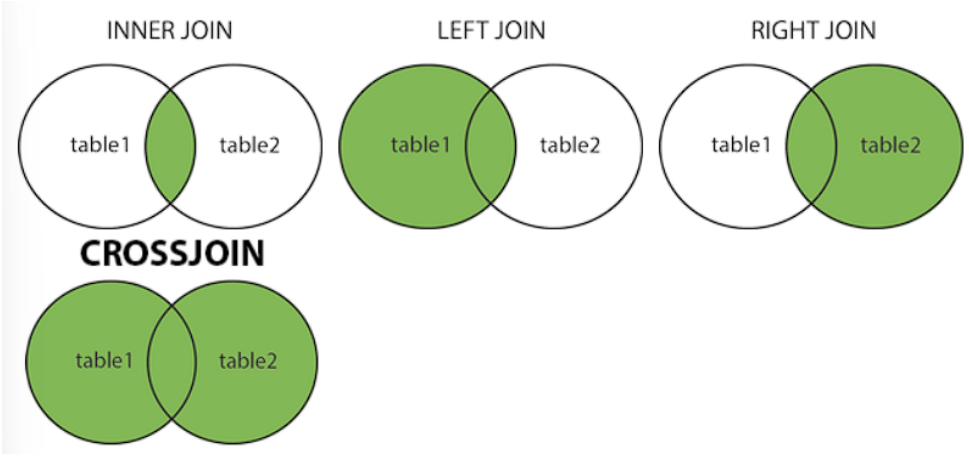
Klausa **HAVING** digunakan untuk mendefinisikan kondisi dari record yang akan ditampilkan dari kelompok group yang dibuat dengan klausa GROUP BY. Penggunaan klausa HAVING dipakai sebagai pengganti klausa WHERE yang tidak dapat untuk menyaring data dalam hubungannya dengan penggunaan GROUP BY. Klausa HAVING dapat digunakan hanya jika terdapat klausa GROUP BY.

XII. MULTI-TABLE ACCESS (JOIN)

Join digunakan untuk mengkombinasikan baris dari 2 tabel atau lebih, berdasarkan kolom yang berelasi di antara tabel-tabel tersebut. terdapat beberapa jenis join yang terdapat pada mysql.

- inner join = mengembalikan hasil yang sama pada kedua tabel

- left join = mengembalikan semua hasil dari tabel kiri dan hasil yang sama pada tabel kanan
- right join = mengembalikan semua hasil dari tabel kanan dan hasil yang sama pada tabel kiri
- cross join = mengembalikan semua hasil dari kedua tabel



Contoh join:
tabel order

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

tabel customer

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

syntax:

```
SELECT Orders.OrderID, Customers.CustomerName,
Orders.OrderDate
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID=Customers.CustomerID;
```

dengan syntax di atas akan menghasilkan seperti berikut:

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

1. Inner join

   syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

   contoh:

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID =
Customers.CustomerID;
```

2. Left join

   syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

   contoh:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID =
Orders.CustomerID
ORDER BY Customers.CustomerName;
```

3. Right join

   syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

   contoh:

```
SELECT Orders.OrderID, Employees.LastName,
Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID =
Employees.EmployeeID
ORDER BY Orders.OrderID;
```

4. Cross join

   syntax:

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

   contoh:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
CROSS JOIN Orders;
```