



Modul Pelatihan

# **Oracle Fundamental**

---

Relational Database Management System

# **Oracle 8i** **SQL**

SYARAT : Database Relational, ER-Model  
BAB I - SQL Basic (1 pertemuan)  
BAB II - SQL Function (1 pertemuan)  
BAB III - SubQueries

---

**V2.00a - 2005**

## BAB I

### SQL Basic

#### A. SQL Statement

Untuk menampilkan data dari database, kita menggunakan perintah SQL select.

Dengan menggunakan SELECT statement kita dapat melakukan hal2 berikut :

1. Selection : Kita dapat menggunakan SELECT statement untuk memilih row/data pada table sesuai yang kita inginkan dengan query.
2. Projection : dengan projection kita dapat menampilkan kolom yang kita inginkan dari table dengan query.
3. Join : Dengan join kita dapat menampilkan data dari beberapa table yang berhubungan dengan membuat link antar table tersebut.

Syntax Select Statement dasar :

```
SELECT      [DISTINCT]    {*, column [alias], . . . }
FROM        table;
```

Keterangan :

SELECT	List satu atau beberapa kolom
DISTINCT	Menampilkan hanya satu data jika ada duplikasi
*	Menampilkan Semua kolom
<i>column</i>	Menampilkan kolom dengan nama <i>column</i>
alias	Memberikan nama berbeda untuk kolom
FROM <i>table</i>	Nama table yang di query

#### Menampilkan semua kolom dari tabel departement

```
SQL> SELECT *
2 FROM dept;
```

```
DEPTNO      DNAME          LOC
-----
          10 ACCOUNTING     NEW YORK
          20 RESEARCH     DALLAS
          30 SALES        CHICAGO
          40 OPERATIONS   BOSTON
```

untuk menampilkan semua kolom kita bisa menggunakan asterik (\*) atau select semua kolom dari table yang di query. Untuk kasus table department ada tiga kolom yaitu DEPTNO,DNAME dan LOC, maka ketika kolom tersebut akan kita gunakan dalam Select statement.

```
SQL> SELECT deptno, dname, loc
2 FROM dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

### Arithmetic Expression

Jika pada suatu saat kita ingin melakukan modifikasi data hasil query, seperti melakukan penambahan, pengurangan dll, kita bisa menggunakan operator arithmetik untuk melakukannya

```
SQL> SELECT ename, sal, sal+200
2 FROM emp;
```

ENAME	SAL	SAL+200
ALLEN	1600	1800
WARD	1250	1450
JONES	2975	3175
MARTIN	1250	1450

Urutan eksekusi operator adalah \* / + -

```
SQL> SELECT ename, sal, 12*sal+10
2 FROM emp;
```

ENAME	SAL	12*SAL+10
ALLEN	1600	19210
WARD	1250	15010
JONES	2975	35710
MARTIN	1250	15010

### Mendefinisikan kolom ALias

Untuk memberikan nama kolom yang berbeda pada tampilan kita menggunakan alias, contoh:

```
SQL> SELECT ename as "NAMA", sal "GAJI"
2 FROM emp;
```

NAMA	GAJI
------	------

```
-----
ALLEN          1600
WARD           1250
JONES          2975
MARTIN         1250
```

Selain menggunakan tanda petik (""), kita juga bisa menggunakan keyword AS. Tapi jika alias yang diberikan terdiri dari dua kata kita harus menggunakan tanda petik. Keyword as dan tanda petik bisa digunakan bersama-sama.

## Operator Penggabungan (concatenation)

Digunakan untuk menggabungkan kolom, karakter atau string dengan kolom yang lain. Digambarkan dengan dua garis tegak (||).

Sebagai contoh kita ingin menampilkan data karyawan x adalah seorang y.

```
SQL> SELECT ename || ' Adalah Seorang ' || job as "DETAIL
KARYAWAN"
  2 FROM emp;

DETAIL KARYAWAN
-----
ALLEN Adalah Seorang SALESMAN
WARD Adalah Seorang SALESMAN
JONES Adalah Seorang MANAGER
MARTIN Adalah Seorang SALESMAN
```

## Duplicate Row

Secara default hasil query menampilkan semua data tanpa mengeliminasi data yang duplikat. Untuk menangani itu kita gunakan keyword DISTINCT.

```
SQL> SELECT deptno
  2 FROM emp;

DEPTNO
-----
  10
  20
  10
  30
  30

SQL> SELECT DISTINCT deptno
  2 FROM emp;

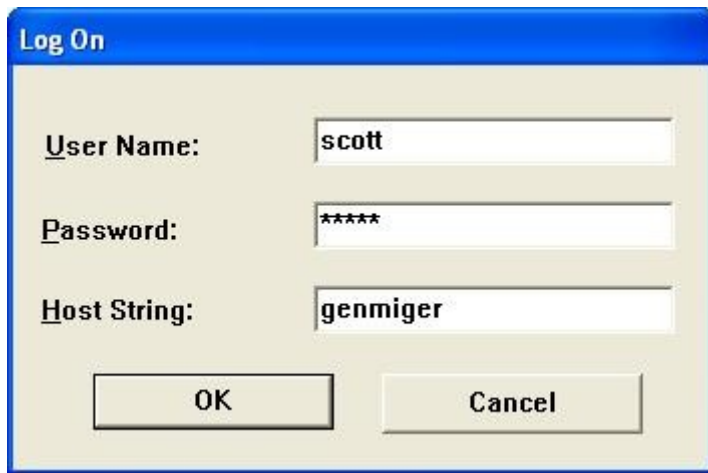
DEPTNO
-----
  10
```

## B. SQL \*Plus

SQL \*Plus adalah suatu environment yang padanya kita dapat melakukan eksekusi perintah SQL, Format, print hasil query untuk keperluan report, membuat script SQL yang bisa digunakan untuk waktu berikutnya.

### Loggin ke SQL \*Plus

Untuk windows environment



User Name : User database  
 Password : Password user database  
 Host String : Database Connection String

Untuk command line

```
CMD> sqlplus [ username[/password[@database]]]
```

Untuk keamanan pada sqlplus masukkan hanya username dengan database, untuk password diisi dari prompt password. Jika berhasil SQL \*Plus akan memberikan informasi seperti dibawah ini :

```
Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
SQL>
```

### Menampilkan Struktur Table

Dalam SQL \*Plus untuk menampilkan struktur table digunakan perintah DESC atau DESCRIBE

```
SQL> DESCRIBE dept;
Name                                     Null?    Type
-----
DEPTNO                                  NOT NULL NUMBER(2)
DNAME                                              VARCHAR2(14)
LOC                                                VARCHAR2(13)
```

*Null?* Mengindikasikan isi dari kolom, NOT NULL berarti kolom harus berisi data.

*Type* Informasi tipe data dari kolom.

### SQL \*Plus Command Edit

Command	Deskripsi
A[PPEND] <i>text</i>	Menambahkan <i>text</i> ke akhir dari baris
C[HANGE] / <i>old</i> / <i>new</i>	Mengganti <i>text old</i> dengan <i>new</i>
C[HANGE] / <i>text</i> /	Menghapus <i>text</i> dari baris
CL[EAR] BUFF[ER]	Menghapus buffer
DEL	Menghapus baris
I[NPUT]	Menambahkan sejumlah baris
I[NPUT] <i>text</i>	Menambahkan satu baris berisi <i>text</i>
L[IST]	Menampilkan semua SQL di buffer
L[IST] <i>n</i>	Menampilkan baris ke <i>n</i> dari SQL Buffer
L[IST] <i>m n</i>	Menampilkan baris antara <i>m</i> dan <i>n</i> dari SQL Buffer
<i>n</i>	Membuat baris ke <i>n</i> menjadi baris sekarang
<i>n text</i>	Replace baris ke <i>n</i> dengan <i>text</i>

### SQL \*Plus File Command

Command	Deskripsi
SAV[E] <i>filename</i> [.ext] [RE]PLACE[APP[END]]	Menyimpan SQL Buffer kedalam <i>filename</i>
GET <i>filename</i> [.ext]	Menuliskan isi <i>filename</i> ke SQL Buffer
STA[RT] <i>filename</i> [.ext]	Menjalankan <i>filename</i>
ED[IT]	Memanggil editor untuk edit SQL Buffer
SPO[OL] <i>filename</i> [.ext]	Meyimpan hasil query ke <i>filename</i>
EXIT	Keluar dari SQL *Plus

### C. SQL \*Plus Vs SQL Statement

SQL adalah bahasa yang digunakan untuk berkomunikasi dengan database Oracle server dari sebuah tool atau aplikasi. Oracle SQL

terdiri dari beberapa bagian, ketika kita menjalankan perintah SQL, sql akan disimpan dalam suatu memory yang disebut SQL Buffer dan akan tetap berada dalam memory sampai ada statemen baru. SQL \*Plus adalah tool oracle yang mengenali dan menjalankan perintah SQL ke Oracle Server.

Berikut dalah tabel perbedaan antara SQL dan SQL \*Plus

SQL	SQL *Plus
Bahasa untuk komunikasi	Suatu Environment
ANSI Standard	Khusus untuk Oracle
Manipulasi Data di database	Tidak bisa memanipulasi data
Di disimpan di SQL Buffer	Tidak ada SQL Buffer
Keyword tidak dapat disingkat	Keyword bisa disingkat
Menggunakan fungsi untuk Format data	Menggunakan command untuk format data

#### D. WHERE Clause

Untuk menampilkan data dari database dengan kondisi yang kita inginkan misalnya hanya menampilkan employee yang mempunyai gaji lebih dari 1000 kita menggunakan WHERE clause.

```
SELECT      [DISTINCT]  { * | column [alias] , . . . }
FROM        table
[WHERE     condition(s) ]
```

**WHERE** Membatasi data hasil query yang sesuai dengan kondisi yang diberikan.  
**condition(s)** merupakan gabungan dari kolom, ekspresi, atau operator perbandingan.

```
SQL> SELECT  ename, job, deptno
2 FROM      emp
3 WHERE     job='CLERK';

ENAME      JOB              DEPTNO
-----
SMITH      CLERK             20
ADAMS      CLERK             20
JAMES      CLERK             30
MILLER     CLERK             10
```

Pada contoh diatas kita ingin menampilkan semua employee yang mempunyai pekerjaan sebagai CLERK. Pada kondisi diatas CLERK ditulis huruf besar karena String dalam data di oracle case sensitive.

Catatan :

- Nilai Character string dan date ditempatkan diantara tanda petik 1 " ' "
- Nilai Character string adalah case sensitive dan nilai Date format sensitive.
- Default datae format adalah DD-MON-YY

Operator	Arti
=	Sama Dengan
>	Lebih Dari
>=	Lebih Dari Sama Dengan
<	Kurang Dari
<=	Kurang Dari Sama Dengan
<>	Tidak Sama Dengan
BETWEEN x1 AND x2	Antara x1 dan x2
IN (list)	Yang sesuai dengan nilai list
LIKE	Sesuai dengan pola karakter
IS NULL	Nilai Null

Syntax

.. WHERE expr operator value

contoh :

```

... WHERE hiredate = '01-JAN-95'
... WHERE sal >= 1500
... WHERE ename = 'SMITH'
    
```

query untuk menampilkan employee yang mempunyai gaji lebih kecil atau sama dengan komisi.

```

SQL> SELECT  ename, sal, comm
      2 FROM    emp
      3 WHERE   sal < = comm;

ENAME          SAL          COMM
-----
MARTIN          1250          1400
    
```

menampilkan semua employee yang mempunyai gaji antara 1000 dan 1500 dolar.

```

SQL> SELECT  ename, sal
      2 FROM    emp
      3 WHERE   sal BETWEEN 1000 AND 1500;

ENAME          SAL
-----
WARD            1250
    
```

MARTIN	1250
TURNER	1500
ADAMS	1100
MILLER	1300

## Menggunakan operator LIKE

```
SQL> SELECT  ename
      2 FROM emp
      3 WHERE ename LIKE '_A%';
```

```
ENAME
-----
WARD
MARTIN
JAMES
```

Symbol	Deskripsi
%	Merupakan gabungan beberapa karakter atau bias juga null
_	Bersesuaian dengan satu karakter

## Rule of precedence

Order	Operator
1	Semua operator Perbandingan
2	NOT
3	AND
4	OR

## E. ORDER BY Clause

Dengan `order by` kita dapat mengurutkan data hasil query ascending atau descending. `Order By` harus diletakkan di statement paling belakang.

### Syntax:

```
SELECT      expr
FROM        table
[WHERE      condition(s)]
[ORDER BY  {column, expr} [ASC | DESC ]]
```

default urutan adalah ascending yaitu dari kecil ke besar. Untuk mengurutkan dari besar ke kecil anda harus menggunakan kata kunci DESC.

## E. SQL Basic Workshop

1. Tuliskan perintah SQL untuk menampilkan struktur table DEPARTMENTS dan menampilkan semua data.
2. Buat sebuah query untuk menampilkan nama, job dan hiredate dan nomor karyawan dari table employee, dan simpan SQL tersebut dengan nama `b1q2.sql`.
3. Jalankan file sql `b1q2.sql` yang sudah anda simpan.
4. Buat query untuk menampilkan jenis pekerjaan dari table EMP (unik).
5. Load file `b1q2.sql` ke SQL Buffer, ganti tampilan kolom dengan Emp #, Employee, Job dan Hire Date. Jalankan query!
6. Tampilkan semua data EMP, pisahkan untuk setiap kolom dengan koma (,), dan beri nama kolom DAFTAR KARYAWAN.
7. Tampilkan nama dan gaji karyawan yang gajinya lebih dari 2850. Simpan SQL dengan nama `b1q7.sql`.
8. Edit file `b1q7.sql` untuk menampilkan nama dan gaji keryawan dimana gaji tidak termasuk dalam range anantara 1500 dan 2850. Simpan ulang dengan nama `b1q8.sql`.
9. Tampilkan nama dan deptno yangg deptno 10 dan 30 urutkan hasil berdasarkan nama.
10. Tampilkan nama dari karyawan dimana huruf ke tiga adalah A.
11. Tampilkan nama karyawan yang mempunyai dua huruf L dan lokasi kerja di dept 30 atau managernya 7782.
12. Tampilkan nama, job, salary untuk semua karyawan dimana pekerjaannya Clerk tau Analyst dan gaji tidak sama dengan 1000, 3000 , atau 5000.
13. Tampilkan Nama, gaji, dan komisi dari semua karyawan dimana komisi lebih besar dari gaji yang sudah dinaikkan 10%.

## BAB II

### SQL FUNCTION

Tujuan :

- Mengetahui type dan fungsi dalam SQL
- Menggunakan Fungsi Char, Number, dan Date dalam SELECT statement
- Menjelaskan penggunaan Fungsi Konversi.

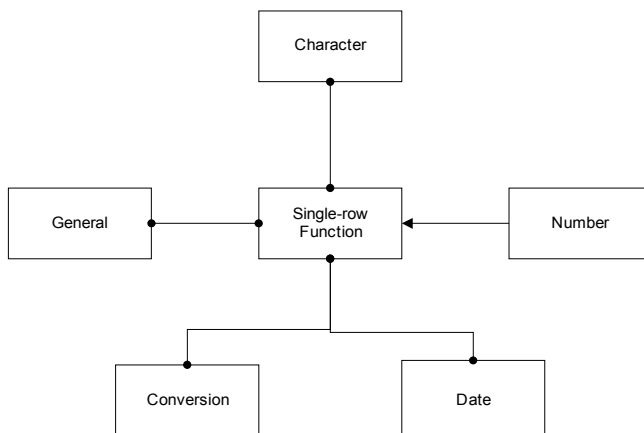
#### A. Single Row Function

Dalam kelompok ini inputnya adalah satu row dan menghasilkan satu hasil per-row.

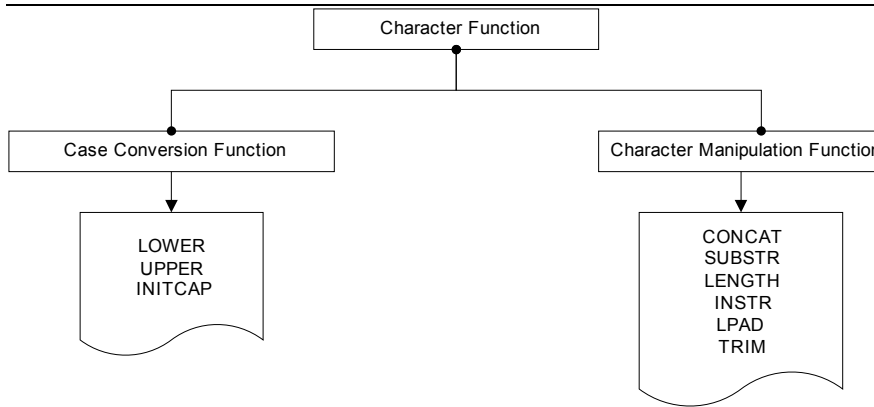
Untuk single row function dapat dikategorikan sebagai berikut :

- Character
- Number
- Date
- Conversion

Single row function digunakan untuk memanipulasi data. Menerima input satu atau lebih argumen dan mengembalikan sebuah nilai untuk setiap row yang dihasilkan query. Inputan dapat berupa konstanta, variable, nama kolom atau ekspresi.



#### A.1 Character Function



Function	Purpose
LOWER (column   expression)	Merubah karakter menjadi lowercase
UPPER (column   expression)	Merubah karakter menjadi uppercase
INITCAP (column   expression)	Mengubah Huruf pertama dari setiap kata menjadi uppercase
CONCAT(column1  expression1,column2  expression2)	Menggabungkan expression1 dan expression2. (ekuivalen dengan   )
SUBSTR(column  expression,m[n])	Mengembalikan karakter dari input mulai dari karakter pada posisi m. n adalah panjang karangter yang diambil. Jika m negative proses pengambilan dari belakang. Jika n tidak disisi maka semua karakter dari posisi m ke akhir akan diambil
LENGTH(column expression)	Mengembalikan jumlah karakter dari input
INSTR(column expression,m)	Mengembalikan posisi dari m karakter
LPAD(column  expression,n,'string')	Membuat rata kanan dengan menambahkan string jika input kurang dari n
TRIM(expression)	Menghilangkan spasi pada string input.

### Case Conversion Function

Function	Result
LOWER('SQL Fundamental')	sql fundamental
UPPER('SQL Fundamental')	SQL FUNDAMENTAL
INITCAP('SQL Fundamental')	Sql Fundamental

Contoh penggunaan character function

```
SQL> SELECT empno, ename, deptno
2 FROM emp
3 WHERE ename='blake';

no rows selected
```

```
SQL> SELECT empno, ename, deptno
2 FROM emp
3 WHERE ename=UPPER('blake');

EMPNO ENAME DEPTNO
-----
7698 BLAKE 30
```

### Character Manipulation Function

Function	Result
CONCAT('SQL','Basic')	SQL Basic
SUBSTR('String',1,3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	*****5000
TRIM('X' FROM 'XXDATA X')	DATA

### Contoh Penggunaan

```
SQL>SELECT      ename,          CONCAT(ename, job) ,          LENGTH(ename) ,
INSTR(ename, 'A')
2 FROM emp
3 WHERE SUBSTR(job,1,5) = 'SALES';

ENAME          CONCAT(ENAME, JOB)          LENGTH(ENAME)          INSTR(ENAME, 'A')
-----
ALLEN          ALLENSALESMAN              5                      1
WARD           WARDSALESMAN               4                      2
MARTIN        MARTINSALESMAN             6                      2
TURNER        TURNERSALESMAN             6                      0
```

### A.2 Number Function

Number Function menerima input numeric dan mengembalikan nilai numeric:

Function	Purpose
ROUND(column expression,n)	Membulatkan kolom atau ekspresi menjadi n dibelakan koma. Jika n null tidak ada nilai dibelakang koma, jika n negative nilai decimal sebelum koma diset nol
TRUNC(column expression)	

MOD(m,n)	Mengembalikan sisa pembagian m terhadap n
----------	---

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
2  ROUND(45.923,-1)
3  FROM DUAL;
```

ROUND(45.923,2)	ROUND(45.923,0)	ROUND(45.923,-1)
-----	-----	-----
45.92	46	50

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
2  TRUNC(45.923,-1)
3  FROM DUAL;
```

TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-1)
-----	-----	-----
45.92	45	40

```
SQL> SELECT ename, sal, comm, MOD(sal, comm)
2  FROM emp
3  WHERE job='SALESMAN';
```

ENAME	SAL	COMM	MOD(SAL,COMM)
-----	-----	-----	-----
ALLEN	1600	300	100
WARD	1250	500	250
MARTIN	1250	1400	1250
TURNER	1500	0	1500

### A.3 Date Function

Oracle menyimpan data dalam numeric format yang merepresentasikan abad, tahun, bulan, hari, jam, menit dan detik. Tampilan default adalah DD-MON-YY, nilai valid dalam oracle adalah antara 1 januari 4712 BC dan 31 Desember 9999.

#### **SYSDATE**

Sysdate adalah fungsi yang mengembalikan tanggal hari ini. Anda dapat menggunakan sysdate seperti menggunakan kolom lain, sebagai contoh anda dapat menampilkan tanggal sekarang dengan memilih sysdate dari table dummy yang disebut DUAL.

#### **DUAL**

Dual adalah dummy table yang dimiliki oleh SYS dan dapat diakses oleh semua user. Dual berisi satu kolom dan satu row yang berisi nilai x.

```
SQL> SELECT SYSDATE
2  FROM DUAL;
```

### Aritmatika dalam Date

Karena tipe date disimpan oleh oracle berupa numeric data, maka anda bisa melakukan operasi aritmatika padanya seperti penjumlahan atau pengurangan.

Operasi yang bisa dilakukan adalah sebagai berikut:

Operasi	Hasil	Keterangan
Date + number	Date	Menambahkan sejumlah hari ke date
Date - number	Date	Mengurangi date dengan sejumlah hari
Date-date	Day	Mengurai date dengan date
Date+number/24	Date	Menambahkan jam ke date

### Fungsi –fungsi dalam tipe data Date

Fungsi	Keterangan
MONTHS_BETWEEN	Jumlah bulan antara dua tanggal
ADD_MONTHS	Menambahkan bulan ke tanggal
NEXT_DAY	Hari berikutnya dari tanggal yang disebutkan.
LAST_DAY	Hari terakhir dari bulan
ROUND	Round tanggal
TRUNC	Truncate tanggal

Keterangan :

- ☞ MONTHS\_BETWEEN(*date1*, *date2*) : mencari jumlah bulan antara *date1* dan *date2*. hasil bisa positif atau negatif.
- ☞ ADD\_MONTHS(*date*,*n*) : menambahkan *n* bulan ke *date*.
- ☞ NEXT\_DAY(*date*,*'char'*) : mencari tanggal berikutnya sesuai dengan hari yang diberikan.
- ☞ LAST\_DAY(*date*) : Mencari tanggal terakhir dari bulan dalam *date*.
- ☞ ROUND(*date*[,*'fmt'*]) : mengembalikan tanggal yang diround berdasarkan format yang diberikan. Jika format tidak diberikan maka tanggal akan di round ke tanggal terdekat.
- ☞ TRUNC(*date*[,*'fmt'*]) : truncate tanggal dan menghilangkan format waktu.

Contoh :

```
MONTHS_BETWEEN('01-SEP-95','11-JAN-94')      = 19.6774194
ADD_MONTHS('11-JAN-94',6)                   = 11-JUL-94
NEXT_DAY('01-SEP-95','FRIDAY')              = 08-SEP-95
LAST_DAY('01-SEP-95')                       = 30-SEP-95
ROUND('25-JUL-95','MONTH')                   = 01-AUG-95
ROUND('25-JUL-95','YEAR')                   = 01-JAN-96
TRUNC('25-JUL-95','MONTH')                  = 01-JUL-95
```

TRUNC('25-JUL-95','YEAR') = 01-JAN-95

```
SQL> SELECT empno, hiredate,
2         ROUND(hiredate, 'MONTH'), TRUNC(hiredate, 'MONTH')
3 FROM emp;

EMPNO HIREDATE   ROUND(HIR TRUNC(HIR
-----
7654 28-SEP-81 01-OCT-81 01-SEP-81
7782 09-JUN-81 01-JUN-81 01-JUN-81
7839 17-NOV-81 01-DEC-81 01-NOV-81
7902 03-DEC-81 01-DEC-81 01-DEC-81
7934 23-JAN-82 01-FEB-82 01-JAN-82
```

### A.4 Conversion Function

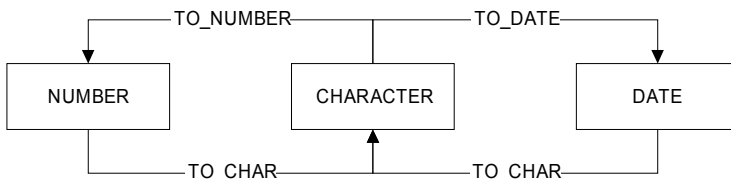
Fungsi konversi di oracle terbagi menjadi dua yaitu:

- ▢ Implicit Datatype Conversion
- ▢ Expicit Datatype Conversion.

Secara umum oracle server menggunakan rule ekspresi, ketika konversi tipe data diperlukan.

Sebagai catatan CHAR ke NUMBER hanya berhasil jika karakter adalah valid number, dan CHAR ke DATE berhasil jika format dari char adalah DD-MON-YY.

Explicit Datatype Conversion adalah konversi tipe data menggunakan fungsi yang telah disediakan oleh oracle.



Oracle meyediakan tiga fungsi utama untuk melakukan konversi tipe data :

FUNCTION	KEGUNAAN
TO_CHAR(number/date,[fmt],[nlsparams])	Mengkonversi number atau Date ke tipe data VARCHAR2 dengan format <i>fmt</i> . <b>Number Conversion :</b> <i>nlsparams</i> dapat berupa : <ul style="list-style-type: none"> <li>- decimal character</li> <li>- Group separator</li> <li>- Local currency</li> <li>- International currency</li> </ul>
TO_CHAR(number/date,[fmt],[nlsparams])	<b>Date Conversion :</b> <i>nlsparams</i> merupakan

	format dari tanggal, hari, bulan dan tahun.
TO_NUMBER(char,[fmt],[nlparams])	Mnegkonversi charater uang berisi digit ke number.
TO_DATE(char,[fmt],[nlparams])	Mengkonversi character yang bersesuaian dengan tanggal ke tipe data date.

Format Element tanggal yang valid di oracle.

Element	Keterangan
SCC atau CC	Abad : S adalah prefix BC
YYYY atau SYYYY	Year : S prefix BC
YYY , YY, Y	Tahun 3, 2 atau 1 angka terakhir
MM	Bulan, dua digit
MON	Nama bulan, tiga kata
MONTH	Nama bulan lengkap
DDD, DD, D	Hari dari tahun, bulan atau minggu
DAY	Nama hari
DY	Nama hari disingkat 3 huruf.
J	Julian date
WW, W	Minggu dari tahun atau bulan.
HH, HH12, HH24	Jam dari hari, jam (1-12), jam (0-23)
MI	Menit (0-59)
SS	Detik (0-59)
TH	Ordinal number (contoh DDTH hasilnya 4th)
AM, PM	Meridian indicator.

```
SQL> SELECT ename, TO_CHAR(hiredate,'fmDD Month YYYY') hiredate
2 FROM emp;
```

```
ENAME          HIREDATE
-----
ALLEN          20 February 1981
WARD           22 February 1981
JONES          2 April 1981
MARTIN         28 September 1981
```

TO\_CHAR untuk konversi Number

Element	Deskripsi
9	Merepresentasikan number
0	Menuliskan angka 0
\$	Menempatkan tanda dolar
L	Menggunakan local currency simbol
.	Nilai desimal
,	Nilai ribuan

```
SQL> SELECT ename, TO_CHAR(sal, '$99,999') salary
2 FROM emp;
```

ENAME	SALARY
ALLEN	\$1,600
WARD	\$1,250
JONES	\$2,975

## A.5 NVL Function

NVL Function berfungsi untuk mengkonversi nilai null ke nilai yang sebenarnya

Sebagai contoh :

- NVL(comm,0)
- NVL(hiredate,'01-JAN-81')
- NVL(job,'Job Not Yet')

```
SQL> SELECT ename, sal, comm, (sal*10)+NVL(comm,0) total
2 FROM emp;
```

ENAME	SAL	COMM	TOTAL
ALLEN	1600	300	16300
WARD	1250	500	13000
JONES	2975		29750
MARTIN	1250	1400	13900

## A.6 DECODE FUNCTION

Fungsi decode mirip dengan kondisi if then else yang dipakai pada SQL statement.

```
DECODE(col/expression, search1, result1
      [, search2, result2, . . . ]
      [, default])
```

contoh penggunaan :

kita ingin memberikan perubahan gaji pada karyawan sesuai kondisi dibawah ini

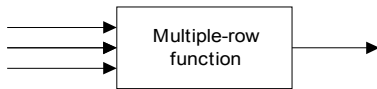
```
if job = 'ANALYST' then sal = sal * 1.1
if job = 'CLERK' then sal = sal * 1.15
if job = 'MANAGER' then sal = sal * 1.20
else sal = sal;
```

```
SQL> SELECT job, sal,
2 DECODE(job, 'ANALYST', sal*1.1,
3         'CLERK', sal*1.15,
4         'MANAGER', sal*1.20,
5         sal) NEW_SAL
6 FROM emp;
```

JOB	SAL	NEW_SAL
MANAGER	2975	3570
SALESMAN	1250	1250
MANAGER	2850	3420
ANALYST	3000	3300
PRESIDENT	5000	5000
SALESMAN	1500	1500
CLERK	1100	1265
CLERK	950	1092.5

## B. Multiple Row Function

Pada multiple Row function fungsi menerima input lebih dari satu dan menghasilkan satu output.



Contoh dari multiple row function adalah :

- AVG
- SUM
- COUNT
- MAX
- MIN
- STDDEV

Untuk penjelasan dari fungsi diatas akan ada pada bab selanjutnya.

## C. SQL Workshop

1. Buatlah query untuk menampilkan tanggal hari ini dan beri nama kolom Tanggal Sekarang.
2. Tampilkan nomor karyawan, nama, gaji dan gaji yang sudah ditambah 15% dan simpan dengan nama b2p2.sql.
3. Load file b2p2.sql dan edit nama kolom untuk gaji yang ditambahkan dengan Gaji Baru.
4. Modifikasi script b2p2.sql tambahkan kolom yang isinya nilai penambahan gaji dan beri label penambahan.
5. Tampilkan nama karyawan, tanggal masuk kerja dan tanggal gaji pertama. Gaji pertama didapat setelah bekerja 6 bulan minggu pertama, beri nama kolom review date. Format review date 'Sunday, the seventh of September, 1985'.
6. Tampilkan nama karyaan dan lama mereka bekerja dalam bulan. Beri label kolom dengan MONTHS\_WORKED, urutkan berdasarkan lama bekerja.

7. Tulis sebuah query untuk menampilkan data dari tabel emp dengan format <Nama karyawan> mendapatkan <Gaji> setiap bulan, tetapi ingin mendapatkan gaji <3 \* gaji> dan beri nama kolom Gaji Impian.
8. Buat query untuk menampilkan nama dan gaji karyawan, format gaji karyawan menjadi 15 karakter rata kiri padded dengan 0.
9. Tulis sebuah query untuk menampilkan nama dan panjang nama dari semua karyawan yang nama awalnya J, A atau M. Untuk Nama huruf pertama dicetak kapital.
10. Tampilkan nama, hiredate dan hari dimana karyawan pertama kali bekerja beri nama kolom DAY.
11. Buat sebuah query untuk menampilkan nama dan komisi karyawan, jika karyawan tidak mempunyai komisi tuliskan No Commision dan beri label COMM.
12. Buat sebuah query untuk menampilkan nama dan gaji karyawan yang ditulis dengan tanda asterik(\*). Setiap \* mewakili gaji 100. urutkan berdasarkan gaji secara descending.
13. Buat sebuah query untuk menunjukkan grade setiap karyawan berdasarkan pekerjaan. Daftar grade berdasarkan pekerjaan ditunjukkan oleh tabel berikut

JOB	Grade
PRESIDENT	A
MANAGER	B
ANALYST	C
SALESMAN	D
CLERK	E
None	O

## BAB III

### JOIN, SUB-QUERY

Terkadang anda pasti ingin menampilkan data dari beberapa table, sebagai contoh anda ingin menampilkan nama, nomor departemen dan lokasi kantor. Untuk mendapatkan itu anda harus menggabungkan table EMP dan DEPT untuk menampilkan data tersebut.

Syntax:

```
SELECT    table1.column, table2.column
FROM      table1, table2
WHERE     table1.column1 = table2.column2
```

Ketika data dari lebih satu table dalam database maka kondisi join diperlukan untuk menampilkannya. Row dari satu table dapat dijoin dengan row pada table lain berdasarkan kolom tertentu yang berhubungan, biasanya primary key dan foreign key.

Untuk menampilkan data dari dua tabel atau lebih digunakan kondisi join pada WHERE clause. Jika ada n table yang dijoin minimal ada n-1 kondisi join.

#### A. Kartesian Produk

Kartesian produk terjadi ketika:

- kondisi join tidak diberikan.
- Kondisi join tidak valid.
- Semua row pada table pertama dijoin dengan semua row pada tabel kedua.

Ketika kondisi join tidak valid atau tidak diberikan dengan benar maka akan dihasilkan cartesian product, dimana semua kombinasi row akan ditampilkan. Kartesian produk akan mengenerate jumlah record yang banyak, dan hasilnya tidak terlalu perlu, kecuali memang anda ingin menampilkan semua kombinasi yang mungkin dari dua table yang dijoin.

#### A. Tipe Join

Ada dua tipe utama dalam join yaitu :

- Equijoin
- Non-Equijoin

Selain tipe join diatas ada beberapa tipe join yang lain yaitu outer join, Self join dan Set Operator.

#### A.1 Equijoin

Untuk menunjukkan data employee dan nama departemen anda harus membandingkan nilai pada kolom DEPTNO pada table EMP dengan kolom DEPTNO pada table DEPT. hubungan antara table EMP dan DEPT disebut equijoins. Biasanya tipe join ini menggunakan primary key dan foreign key untuk proses joinnya. Equijoin lebih dikenal sebagai inner join atau simple join.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
2      dept.deptno, dept.dname
3 FROM emp, dept
4 WHERE emp.deptno = dept.deptno;
```

EMPNO	ENAME	DEPTNO	DEPTNO	DNAME
7934	MILLER	10	10	ACCOUNTING
7782	CLARK	10	10	ACCOUNTING
7788	SCOTT	20	20	RESEARCH
7369	SMITH	20	20	RESEARCH

Pada contoh diatas perintah SELECT menampilkan empno, ename dari table EMP, dan deptno, dname dari table DEPT. Klausula FROM menunjukkan ada dua table yang diakses untuk menampilkan data tersebut. Dan kondisi WHERE bagaimana kedua table harus dijoin. Pada kondisi diatas kedua table akan dijoin berdasarkan kolom DEPTNO.

Anda harus menggunakan nama table disertai kolom untuk menghindari ambiguitas.

Menggunakan table Alias.

Menggunakan nama table untuk mengambil kolom akan membutuhkan waktu penulisan jika nama table sangat panjang. Untuk memudahkan oracle menyediakan table alias untuk menggantikan nama table.

```
SQL> SELECT a.empno, a.ename, a.deptno,
2      b.deptno, b.dname
3 FROM emp a,
4      dept b
5 WHERE a.deptno = b.deptno;
```

EMPNO	ENAME	DEPTNO	DEPTNO	DNAME
7934	MILLER	10	10	ACCOUNTING
7782	CLARK	10	10	ACCOUNTING
7788	SCOTT	20	20	RESEARCH
7369	SMITH	20	20	RESEARCH

## A2. Non-Equijoin

Kita lihat table EMP dan SALGRADE, kedua table tersebut adalah non-equijoins, karena tidak ada hubungan secara langsung antara table EMP dengan Table SALGRADE. Hubungan antara kedua table tersebut adalah pada nilai kolom SAL pada table EMP adalah antara kolom LOSAL dan HISAL pada table SALGRADE. Relasi pada tipe ini adalah relasi yang menggunakan tanda selain sama dengan (=).

```
SQL> SELECT a.ename, a.sal, b.grade
2 FROM emp a,
3 salgrade b
4 WHERE a.sal BETWEEN b.losal AND b.hisal;
```

ENAME	SAL	GRADE
KING	5000	5
SCOTT	3000	4
CLARK	2450	4
MILLER	1300	2

## A3. Outer Join

Jika ada suatu row tidak memenuhi kondisi join maka row itu tidak akan dimunculkan. Sebagai contoh jika kita menambah satu departemen pada table DEPT misal OPERATIONS, maka departemen OPERATIONS tidak akan muncul dengan query join diatas karena tidak ada pegawai yang kerja didepartemen itu.

Departemen OPERATIONS dapat dimunculkan dengan menggunakan outer join. Anda dapat melakukan outer join dengan operator (+).

Select contoh

### A.4 Self Join

Kadang kita perlu melakukan join dengan table itu sendiri, sebagai contoh untuk mencari manajer dari setiap pegawai anda membutuhkan join table EMP dengan dirinya sendiri. Pada kasus ini kondisi join adalah kolom MGR dan kolom EMPNO.

Contoh:

## B. Group Function

Sebuah group function menerima input sekumpulan nilai dan menghasilkan satu nilai hasil setiap group.

Beberapa contoh Group Function yang sering dipakai adalah:

Function	Deskripsi
----------	-----------

AVG([DISTINCT ALL],n)	Nilai rata-rata dari n
COUNT({* [DISTINCT ALL]expr})	Jumlah dari row, atau expr
MAX([DISTINCT ALL]expr)	Nilai maksimum dari expr
MIN([DISTINCT ALL]expr)	Nilai minimum dari expr
STDDEV([DISTINCT ALL]x)	Standar deviasi dari n
SUM([DISTINCT ALL]n)	Jumlah dari n
VARIANCE([DISTINCT ALL]x)	Nilai vaiance dari n

Keterangan:

- Option DISTINCT membuat fungsi hanya menggunakan nilai yang tidak duplikat, sedangkan option ALL akan menggunakan semua nilai. Default nilainya adalah ALL.
- Semua fungsi kecuali COUNT(\*) mengabaikan nilai NULL, untuk memperhitungkan nilai NULL digunakan fungsi NVL.
- Secara implisit oracle akan mengurutkan hasil secara ascending ketika menggunakan klausa GROUP BY.
- Tipe data dari argumen dapat berupa CHAR, VARCHAR2, NUMBER atau DATE.

## Menggunakan klausa GROUP BY

Ketika menggunakan GROUP BY pada SELECT statemen anda harus menyakinkan bahwa semua kolom yang tidak termasuk dalam fungsi group, harus disertakan dalam klausa GROUP BY. Sebagai contoh

```
SELECT deptno, AVG(sal)
FROM emp
GROUP BY deptno
```

Pada perintah SELECT diatas, kita ingin menampilkan rata-rata gaji dari tiap departemen.

## Klausa GROUP BY pada banyak Kolom

Pada suatu saat kita terkadang memerlukan menampilkan data hasil group dari proses grouping. Sebagai contoh kita ingin menampilkan jumlah gaji yang harus dibayar untuk tiap pekerjaan dari tiap departemen. Langkah yang harus dilakukan adalah:  
Table EMP pertama digroup berdasarkan departemen, kemudian hasil ini digroup berdasarkan pekerjaan.

```
SELECT deptno, job, SUM(sal)
FROM emp
GROUP BY deptno, job
```

Kesalahan-kesalahan yang sering terjadi :

Kolom yang tidak termasuk dalam fungsi group dan tidak ditempatkan pada klausa GROUP BY.

Yang berikutnya adalah menggunakan fungsi group dalam klausa WHERE.

## Membatasi hasil GROUP By dengan klausa HAVING

Kita menggunakan klausa HAVING untuk membatasi record mana yang akan dimunculkan.

Syntax:

```
SELECT      column, group_function
FROM        table
[WHERE      conditions ]
[GROUP BY   group_by_expression ]
[HAVING     group_condition ]
[ORDER BY   column ]
```

contoh kita ingin menampilkan gaji rata-rata dari tiap departemen dimana maksimum gaji dari tiap departemen lebih dari 2900.

```
SELECT      deptno, AVG(sal)
FROM        emp
GROUP BY    deptno
HAVING      MAX(sal) > 2900
```

## C. Sub-Queries

Jika anda ditanya siapa yang mempunyai gaji lebih besar dari Jones? Untuk menyelesaikan permasalahan ini kita memerlukan dua query. Pertama kita mencari berapa gaji Jones, yang kedua kita mencari gaji karyawan yang gajinya lebih besar dari gaji Jones. Kita bisa menggabungkan dua query diatas menjadi satu query dengan cara menempatkan query pertama didalam query kedua.

```
SELECT      select_list
FROM        table
WHERE       expr operator
              (SELECT select list FROM table)
```

contoh penggunaan sub-queries

```
SELECT      ename
FROM        emp
WHERE       sal >
              (SELECT sal FROM emp WHERE empno=7566)
```

### Tipe Sub-Queries

- Single-row subqueries
- Multiple-row subqueries
- Multiple column subqueries

### Single-Row subqueries

Adalah query yang menghasilkan hanya satu baris dari inner SELECT, dan menggunakan operator perbandingan satu baris (=, >, <, <>, <=, =>).

```
SQL> SELECT  ename, job
  2  FROM emp
  3  WHERE job =
  4          ( SELECT job
  5            FROM emp
  6            WHERE empno = 7369)
  7  AND    sal >
  8          (SELECT sal
  9            FROM emp
 10           WHERE empno = 7369);
```

ENAME	JOB
ADAMS	CLERK
JAMES	CLERK
MILLER	CLERK

Pada query diatas, ada tiga blok query. Inner query dijalankan terlebih dahulu dan menghasilkan job=CLERK dan sal=1100, hasil ini digunakan oleh outer query untuk mencari data sesuai dengan yang diinginkan.

Dalam sub-queries bisa juga menggunakan group function dan klausa having.

### Sub-queries yang salah

```
SQL> SELECT empno, ename
  2  FROM emp
  3  WHERE sal = ( SELECT MIN(sal)
  4                FROM emp
  5                GROUP BY deptno);
WHERE sal = ( SELECT MIN(sal)
              *
ERROR at line 3:
ORA-01427: single-row subquery returns more than one row
```

### Multiple Row Sub-Queries

Adalah sub-queries yang menghasilkan data lebih dari satu baris, dan menggunakan operator perbandingan *multiple-row* (IN, ANY, ALL). Sebagai contoh kita ingin menampilkan pegawai yang mempunyai gaji terendah untuk masing-masing departemen.

```
SQL> SELECT empno, ename
  2  FROM emp
  3  WHERE sal IN (SELECT MIN(sal)
  4                FROM emp
```

```
5          GROUP BY deptno);  
  
      EMPNO ENAME  
-----  
      7934 MILLER  
      7369 SMITH  
      7900 JAMES
```

## multiple-column subqueries

jika kita ingin membandingkan lebih dari satu kolom pada query kita bisa menggunakan multiple column subqueries.

Sintaks :

```
SELECT  column, column, ...  
FROM    table  
WHERE   (column, column, ...) IN  
        (SELECT column, column, ...  
         FROM table  
         WHERE condition);
```