

**Database Management System Lab (CS-2094)**

# **KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**School of Computer Engineering**



Strictly for internal circulation (within KIIT) and reference only. Not for outside circulation without permission

***1 Credit***

**Dr. Jayanta  
Mondal**

# Dual - Revisited



2

## DUAL Table

DUAL table is a small worktable, which consists of only one column **DUMMY** and a single row with value **X** of VARCHAR2 type

This table is owned by user SYS and is available to all users

It is used for Arithmetic calculations and Date retrieval

```
SELECT 2*5 FROM DUAL;
```

```
SELECT SYSDATE FROM DUAL;
```

# An example Employee table



3

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-NOV-81	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-SEP-81	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

## Built-in Functions

The built-in functions provide a powerful tool for the enhancement of a basic query. They serve the purpose of manipulating data items and returning a result. Functions are of two types:

- ***Single-row or Scalar functions:***
  - They work on columns from each row and return one result per row
- ***Group functions or Aggregate functions:***
  - They manipulate data in a group of rows and return single result



## Scalar Functions

### Scalar Functions

These functions act on one value at a time. There are various types of scalar functions:

- **Date functions:** These functions take a date value or date-type column as argument and return date-type data
- **Numeric functions:** These functions take a number or number-type column as argument and return a numeric value
- **Character functions:** These functions take a character string or character-type column as argument and return a character or numeric value
- **Conversion functions:** These functions are used to convert value from one data type to another
- **Misc. functions:** These functions perform some specific tasks

# Built-in-Functions



6

## Date Functions

The date values are stored internally with day, month, year, hour, minute and second information. The different date functions are:

### **SYSDATE**

It is the pseudo column that returns the system's current date

```
SELECT SYSDATE FROM DUAL;
```

SYSDATE
21-JAN-13

### **ADD\_MONTHS(date, n)**

It adds calendar months to a date

```
SELECT ADD_MONTHS(HIREDATE, 4) FROM EMP WHERE  
EMP_NO=7369;
```

ADD_MONTHS(HIREDATE,4)
17-APR-81

# Built-in-Functions



7

## Date Functions...

### **LAST\_DAY(date)**

It returns the last day of the month

```
SELECT LAST_DAY(SYSDATE) FROM DUAL;
```

LAST_DAY(SYSDATE)
31-JAN-13

### **MONTHS\_BETWEEN(date1, date2)**

It finds the number of months between two dates

```
SELECT MONTHS_BETWEEN(SYSDATE,'23-JAN-89') FROM DUAL;
```

MONTHS_BETWEEN(SYSDATE,'23-JAN-89')
287.90

# Built-in-Functions



8

## Date Functions...

### **NEXT\_DAY(date, 'day')**

It finds the next occurrence of a day from the given date

```
SELECT NEXT_DAY(SYSDATE, 'MONDAY') FROM DUAL;
```

NEXT_DAY(SYSDATE, 'MONDAY')
28-JAN-13

### **EXTRACT(YEAR/MONTH/DAY FROM date)**

This extracts the year, month, or day from a date value

```
SELECT EXTRACT(MONTH FROM SYSDATE) FROM DUAL;
```

EXTRACT(MONTH FROM SYSDATE)
1

```
SELECT EXTRACT(YEAR FROM SYSDATE) FROM DUAL;
```

EXTRACT(YEAR FROM SYSDATE)
2013



# Built-in-Functions



9

## Numeric Functions

These functions take numeric values and return a numeric value. The different functions in this category are:

### **ABS(n)**

It returns the absolute value of n

```
SELECT ABS(5), ABS(-100) FROM DUAL;
```

ABS(5)	ABS(-100)
5	100

### **CEIL(n)**

This returns the smallest integer greater than or equal to the given value

```
SELECT CEIL(-5.2), CEIL(5.7) FROM DUAL;
```

CEIL(-5.2)	CEIL(5.7)
-5	6

# Built-in-Functions



10

## Numeric Functions...

### FLOOR(n)

This returns the largest integer less than or equal to the given value

```
SELECT FLOOR(-5.2), FLOOR(5.7) FROM DUAL;
```

FLOOR(-5.2)	FLOOR(5.7)
-6	5

### EXP(n)

It returns the exponent e raised to power n

```
SELECT EXP(5) FROM DUAL;
```

EXP(5)
148.413159

# Built-in-Functions



11

## Numeric Functions...

### LN(n)

It returns the natural logarithm of n

```
SELECT LN(2) FROM DUAL;
```

LN(2)
0.693147181

### LOG(b, n)

It returns  $\log_b n$  value

```
SELECT LOG(4,10) FROM DUAL;
```

LOG(4,10)
1.66096405

### MOD(n, m)

It returns the integer remainder of n/m

```
SELECT MOD(15,4) FROM DUAL;
```

MOD(15,4)
3

# Built-in-Functions



12

## Numeric Functions...

### POWER(m, n)

It returns m raised to power n

```
SELECT POWER(4,3) FROM DUAL;
```

POWER(4,3)
64

### SQRT(n)

It returns the square root of the number n

```
SELECT SQRT(25) FROM DUAL;
```

SQRT(25)
5

```
SELECT SQRT(-25) FROM DUAL;
```

ORA-01428: argument '-25' is out of range
---



# Built-in-Functions



13

## Numeric Functions...

### **ROUND(m, [n])**

It returns m, rounded to n places to the right of a decimal point

```
SELECT ROUND(15.19,1), ROUND(15.19) FROM DUAL;
```

ROUND(15.19,1)	ROUND(15.19)
15.2	15

### **TRUNC(m, n)**

It returns the truncated value of m up to n positions

```
SELECT TRUNC(15.19,1) FROM DUAL;
```

TRUNC(15.19,1)
15.1

## Numeric Functions...

### **SIGN(n)**

It returns the sign of number n: -1 for negative, 0 for zero, 1 for positive

```
SELECT SIGN(-8.5) FROM DUAL;
```

SIGN(-8.5)
-1

### **SIN(n)**

It returns sine of n, where n is in radian

```
SELECT SIN(60), SIN(1.047167) FROM DUAL;
```

SIN(60)	SIN(1.047167)
-0.3048106	0.8660

Other trigonometric functions are: *COS(n)*, *TAN(n)*, *SINH(n)*, *COSH(n)*, *TANH(n)*

# Built-in-Functions



15

## Character Functions

These functions work on character values. The different types of character functions are:

### CHR(n)

It returns the ASCII character corresponding to the integer n

```
SELECT CHR(70) FROM DUAL;
```

CHR(70)
F

### CONCAT(s1, s2)

It joins the first string to the second string. It is similar to the || operator

```
SELECT CONCAT('RAM','KRISHNA'), 'RAM' || 'KRISHNA'  
FROM DUAL;
```

CONCAT('RAM','KRISHNA')	'RAM'    'KRISHNA'
RAMKRISHNA	RAMKRISHNA

# Built-in-Functions



16

## Character Functions...

### LPAD(s, n, c)

It pads the string s with the character c to the left to a total width of n

```
SELECT LPAD('ORACLE',10,'*') FROM DUAL;
```

LPAD('ORACLE',10,'*')
****ORACLE

### RPAD(s, n, c)

It pads the string s with the character c to the right to a total width of n

```
SELECT RPAD('ORACLE',10,'*') FROM DUAL;
```

RPAD('ORACLE',10,'*')
ORACLE****



# Built-in-Functions



17

## Character Functions...

### INITCAP(s)

It returns the string with capitalization of the first letter in each word

```
SELECT INITCAP('HELLO') FROM DUAL;
```

INITCAP('HELLO')
Hello

```
SELECT INITCAP(E_NAME) FROM EMP;
```

### LOWER(s)

It converts each letter to lowercase

```
SELECT LOWER('HELLO') FROM DUAL;
```

LOWER('HELLO')
hello

# Built-in-Functions



18

## Character Functions...

### UPPER(s)

It converts each letter to uppercase

```
SELECT UPPER('HeLLo') FROM DUAL;
```

UPPER('HeLLo')
HELLO

### LTRIM(s, c)

It trims the string s from the left when the characters specified, c, is present in s

```
SELECT LTRIM(E_NAME,'S') FROM EMP;
```

### RTRIM(s, c)

It trims the string s from the right when the characters specified, c, is present in s

```
SELECT RTRIM(E_NAME,'I') FROM EMP;
```

## Character Functions...

### **REPLACE(s, s1, s2)**

It returns the string s with the replacement of s2 in place of s1

```
SELECT REPLACE('ORACLE','RAC','V') FROM DUAL;
```

REPLACE('ORACLE','RAC','V')
OVLE

### **SUBSTR(s, n, m)**

It returns a substring, starting at character position n, and returns m number of characters

```
SELECT SUBSTR('DATABASE',3,2) FROM DUAL;
```

SUBSTR('DATABASE',3,2)
TA

# Built-in-Functions



20

## Character Functions...

### LENGTH(s)

It returns the number of characters present in the string s

```
SELECT LENGTH('ORACLE') FROM DUAL;
```

LENGTH('ORACLE')
6

### SOUNDEX(s)

It compares words that are spell differently, but sound alike

```
SELECT E_NAME FROM EMP WHERE  
SOUNDEX(E_NAME)=SOUNDEX('KEEING');
```

E_NAME
KING



# Decode Function



21

DECODE function has the functionality of an IF-THEN-ELSE statement.

```
SELECT supplier_name, DECODE(supplier_id, 10000, 'IBM', 10001, 'Microsoft', 10002,  
'Hewlett Packard', 'Gateway') result FROM suppliers;
```

The above DECODE statement is equivalent to the following IF-THEN-ELSE statement:

```
IF supplier_id = 10000 THEN  
    result := 'IBM';  
ELSIF supplier_id = 10001 THEN  
    result := 'Microsoft';  
ELSIF supplier_id = 10002 THEN  
    result := 'Hewlett Packard';  
ELSE  
    result := 'Gateway';  
END IF;
```



## Conversion Functions

These functions convert data from one data type to another.  
The different conversion functions are:

### **TO\_NUMBER(char [,format])**

It converts a character value with valid digits to a number using the given format

```
SELECT SUM(SAL) FROM EMP;  
SELECT SUM(TO_NUMBER(SAL)) FROM EMP;
```

### **TO\_DATE(char [,format])**

It converts a character value to date value based on the format provided

```
SELECT TO_DATE('January 7, 1988','month dd, yyyy') FROM  
DUAL;
```

TO_DATE('January 7, 1988','month dd, yyyy')
07-JAN-88

## Conversion Functions...

### TO\_CHAR(number [,format])

It converts a number to a VARCHAR value based on the format provided. *0* is used for compulsory purpose and *9* is used for optional purpose

```
SELECT TO_CHAR(17145,'$999,999') FROM DUAL;
```

TO_CHAR(17145,'\$999,999')
\$17,145

```
SELECT TO_CHAR(17145,'$000,000') FROM DUAL;
```

TO_CHAR(17145,'\$000,000')
\$017,145

# Built-in-Functions



24

## Conversion Functions...

### TO\_CHAR(date [,format])

It converts a date to a VARCHAR value based on the format provided

```
SELECT TO_CHAR(HIREDATE,'MONTH DD, YYYY') FROM  
EMP WHERE EMP_NO=7566;
```

TO_CHAR(HIREDATE,'MONTH DD, YYYY')
APRIL 02, 1981

### Use of TH in Date formatting

It converts a date to a VARCHAR value based on TH format

```
SELECT HIREDATE, TO_CHAR(HIREDATE,'DDTH-MON-YY')  
FROM EMP WHERE DEPT_NO=10;
```

HIREDATE	TO_CHAR(HIREDATE,'DDTH-MON-YY')
09-JUN-81	09TH-JUN-81
17-NOV-81	17TH-NOV-81
23-JAN-82	23RD-JAN-82



# Built-in-Functions



25

## Conversion Functions...

### Use of *SP* in Date formatting

It converts a date to a VARCHAR value with the spelling

```
SELECT TO_CHAR(HIREDATE,'DDSP-MON-YY') FROM EMP  
WHERE DEPT_NO=10;
```

<i>TO_CHAR(HIREDATE,'DDSP-MON-YY')</i>
NINE-JUN-81
SEVENTEEN-NOV-81
TWENTY-THREE-JAN-82

### Use of *SPTH* in Date formatting

It converts a date to a VARCHAR value with the spelling and *TH* format

```
SELECT TO_CHAR(HIREDATE,'DDSPTH-MON-YY') FROM  
EMP WHERE DEPT_NO=10;
```

<i>TO_CHAR(HIREDATE,'DDSPTH-MON-YY')</i>
NINTH-JUN-81
SEVENTEENTH-NOV-81
TWENTY-THIRD-JAN-82

# Built-in-Functions



26

## Misc. Functions

Two important functions to deal with NULL value are:

### **NVL(column, value)**

It converts a NULL value to an actual value supplied as an argument. For numerical values, it accepts 0; whereas for character values, it accepts a fixed string

```
SELECT E_NAME, NVL(COMMISSION, 0)
COMMISSION FROM EMP;
SELECT E_NAME, SALARY+NVL(COMMISSION, 0) Total
Salary FROM EMP;
```

### **NVL2(column, notnullvalue, nullvalue)**

It checks for NULL as well as not NULL values. If the column has a not NULL value, the second parameter is displayed. If the column has a NULL value, the third parameter is displayed

```
SELECT E_NAME, NVL2(COMMISSION, 'YES', 'NO') FROM
EMP;
```

# Built-in-Functions



27

## Group Functions

The group or aggregate functions perform an operation on a group of rows and return one result. The different aggregate functions are:

### **COUNT([DISTINCT] column)**

This function counts the number of rows without considering NULL values

```
SELECT COUNT(MGR) FROM EMP;  
SELECT COUNT(DISTINCT MGR) FROM EMP;
```

COUNT(MGR)	COUNT(DISTINCT MGR)
13	6

### **COUNT(\*)**

It counts the number of rows including NULL values

```
SELECT COUNT(*) FROM EMP;
```

COUNT(*)
14

# Built-in-Functions



28

## Group Functions...

### **SUM([DISTINCT] column)**

It finds the sum of all values in a column ignoring the NULL values

```
SELECT SUM(SAL) FROM EMP;
```

SUM(SAL)
29055

### **AVG([DISTINCT] column)**

It finds the average of all values in a column ignoring the NULL values

```
SELECT AVG(SAL) FROM EMP;
```

AVG(SAL)
2075.35

# Built-in-Functions



29

## Group Functions...

### MAX([DISTINCT] column)

It finds the maximum value in the column ignoring the NULL values

```
SELECT MAX(SAL) FROM EMP;
```

MAX(SAL)
5000

### MIN([DISTINCT] column)

It finds the minimum value in the column ignoring the NULL values

```
SELECT MIN(SAL) FROM EMP;
```

MIN(SAL)
800